بسم الله الرحمن الرحیم

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته علوم تصمیم و مهندسی دانش

ماشین‌های یادگیر نهایی

استاد راهنما

دکترمحمود امین طوسی

استاد مشاور

دکترمهدی زعفرانیه

پژوهشگر:

سکینه خورسندی

بهمن ۱۳۹۵

فرم ۱۰۵

دانشگاه حکیم سبزواری

بسمه تعالی

فرم ارزشیابی و صورتجلسه دفاع از پایان نامه کارشناسی ارشد *

جلسه دفاع از پایان نامه خانم سکینه خورسندی دانشجوی رشته علوم تصمیم و مهندسی دانش به شماره دانشجویی ۹۳۱۳۱۳۷۰۵۳ با عنوان **ماشین‌های یادگیر نهایی** در تاریخ ۱۳۹۵/۱۱/۱۳ در دانشکده ریاضی و علوم کامپیوتر برگزار و توسط هیات داوران مورد ارزشیابی قرار گرفت و نمره.....۱۹/۸..... برابر درجه.....عالی..... برای آن تعیین گردید.

به این ترتیب از این تاریخ نامبرده به عنوان کارشناس ارشد در رشته مذکور شناخته می شود.

موارد ارزشیابی	موارد	حداکثر نمره	نمره کسب شده
۱-کیفیت نگارش	رعایت اصول نگارش انسجام در تنظیم بخشهای مختلف ،کیفیت تصاویر ، جداول و اشکال ،تنظیم فهرست ها ، منابع و ماخذ	۴	۳/۷۸
۲-کیفیت علمی	بررسی تاریخچه و سابقه تجربی و نظری موضوع انسجام منطقی دربخش های مختلف پایان نامه ،ابتکار و نوآوری ،اهمیت و ارزش علمی پایان نامه ،استفاده از منابع معتبر و جدید ،کیفیت تجزیه و تحلیل یافته ها و نتیجه گیری ،روشن بودن روش کار ،هدف ها و فرضیه های تحقیق ،جدید بودن روش تحقیق	۱۰	۱۰
۳-کیفیت ارایه در جلسه دفاع	تسلط بر موضوع و بیان واضح وتفهیم آن ،توانایی در پاسخگویی به سوالات مطرح شده در جلسه ، رعایت زمان ارایه ، روش ارائه	۴	۴
۴-ارزشیابی گزارشات	گزارش های دوره ای پیشرفت کار(حداقل ۴مورد)	۱	۱
۵- خروجی پایان نامه	مقاله مستخرج از پایان نامه: این نمره به صورت زیر اختصاص می یابد ۱) چکیده کنفرانسی هر مورد ۰/۲۵نمره تا سقف۰/۵ نمره ۲) مقاله کامل در مجموع مقالات همایشهای معتبر یا مقاله در مجلات عـی-ترویجی معتبر پذیرفته شده یا چاپ شده هر مورد ۰/۵ نمره تا سقف ۱نمره ۳)مقاله پذیرفته شده یا چاپ شده در مجلات علمی پژوهشی معتبر ۱ نمره ۴)مقاله ارسال شدهبه مجلات علمی پژوهشی معتبرهر مورد ۰/۲۵ نمره تا سقف ۰/۵ نمره ۵) دستگاه ساخته شده دارای گواهی ثبت اختراع یا به سفارش سازمان ها تا سقف ۱ نمره ۶) دستگاه ساخته شده کاربردی که به تایید رئیس دانشکده رسیده باشد تا سقف ۰/۵ نمره	۱	۰/۷۸
جمع			۱۹/۸

درجه معادل کسب شده: (از ۱۹تا ۲۰ عالی☑ از۱۸تا ۱۸/۹۹بسیار خوب ☐ از ۱۶تا ۱۷/۹۹خوب ☐ از ۱۴ تا ۱۵/۹۹قابل قبول☐ کمتر از ۱۴ غیر قابل قبول☐)

مشخصات هیات دوران

ردیف	نام ونام خانوادگی	سمت	مرتبه علمی	محل کار	امضاء
۱	دکتر محمود امین طوسی	استاد راهنما	استادیار	دانشگاه حکیم سبزواری	
۲	دکتر مهدی زعفرانیه	استاد مشاور	استادیار	دانشگاه حکیم سبزواری	
۳	دکتر قدیر صادقی	استاد داور	دانشیار	دانشگاه حکیم سبزواری	
۴	دکتر علیرضا قدسی	نماینده تحصیلات تکمیلی	استادیار	دانشگاه حکیم سبزواری	

امضاء
مدیر گروه
۹۵/۱۱/۱۴

امضاء
رئیس دانشکده

* این فرم الزاما باید به صورت تایپ شده تهیه، ارسال و در پایان نامه درج شود

سوگند نامه دانش آموختگان دانشگاه حکیم سبزواری

به نام خداوند جان و خرد کزین برتر اندیشه بر نگذرد

اینک که به خواست آفریدگار پاک، کوشش خویش و بهره گیری از دانش استادان و سرمایه‌های مادی و معنوی این مرز و بوم، توشه‌ای از دانش و خرد گردآورده‌ام، در پیشگاه خداوند بزرگ سوگند یاد می‌کنم که در به کارگیری دانش خویش، همواره بر راه راست ودرست گام بردارم. خداوند بزرگ، شما شاهدان، دانشجویان و دیگر حاضران را به عنوان داورانی امین گواه می گیرم که ازهمه دانش و توان خود برای گسترش مرزهای دانش بهره گیرم و از هیچ کوششی برای تبدیل جهان به جایی بهتر برای زیستن، دریغ نورزم. پیمان می‌بندم که همواره کرامت انسانی را در نظر داشته باشم و همنوعان خود را در هر زمان و مکان تا سر حد امکان یاری دهم. سوگند می خورم که در به کارگیری دانش خویش به کاری که با راه و رسم انسانی، آیین پرهیزگاری، شرافت و اصول اخلاقی برخاسته از ادیان بزرگ الهی، به ویژه دین مبین اسلام، مباینت دارد دست نیازم. همچنین در سایه اصول جهان شمول انسانی و اسلامی، پیمان می‌بندم از هیچ کوششی برای آبادانی و سرافرازی میهن و هم میهنانم فروگذاری نکنم و خداوند بزرگ را به یاری طلبم تا همواره در پیشگاه او و در برابر وجدان بیدار خویش و ملت سرافراز، بر این پیمان تا ابد استوار بمانم.

نام و نام خانوادگی: سکینه خورسندی

تاریخ و امضا:

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب سکینه خورسندی به شماره دانشجویی ۹۳۱۳۱۳۷۰۵۳ دانشجوی رشته علوم تصمیم و مهندسی دانش مقطع تحصیلی کارشناسی ارشد تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده ی اینجانب خواهد بود و دانشگاه هیچ گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: سکینه خورسندی

تاریخ و امضا:

تقدیم به:

پدر و مادرم

و

برادرانم

خدای را سپاس که اندیشه را آفرید و آن را سفیر عقل و عقل را مرکب روح ساخت تا به اتفاق، آدمی را در مسیر کمال سر منزل مقصود رهنمون سازند و کیست که قابلیت آینه‌ها را منکر شود آن گاه که به فراخور جلای، تجلی‌گر حقیقتی آسمانی خواهند شد و تو ای انسان مخاطب آینه‌هایی!

اما بعد به مصداق حدیث «من علمنی حرفا فقد صیرنی عبدا» بر خود لازم می‌دانم که از استاد گرانقدرم جناب آقای دکتر محمود امین طوسی که در لحظه لحظه این پژوهش راهنمای حقیر بوده‌اند و آقای دکتر مهدی زعفرانیه که مطالعه و مشاوره این مجموعه را تقبل فرمودند خالصانه سپاس‌گزاری کنم.

در پایان، بوسه می زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس‌شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

سکینه خورسندی

بهمن ۱۳۹۵

فهرست مطالب

فهرست جداول

فهرست تصاویر

فهرست الگوریتم‌ها

فرم چکیده ی پایان‌نامه ی دوره ی تحصیلات تکمیلی

مدیریت تحصیلات تکمیلی

دانشگاه حکیم سبزواری

نام خانوادگی دانشجو:خورسندی	نام:سکینه	ش. دانشجویی:۹۳۱۳۱۳۷۰۵۳
استاد راهنما: دکترمحمود امین طوسی		
استاد مشاور: دکترمهدی زعفرانیه		
دانشکده ریاضی و علوم کامپیوتر	رشته: علوم تصمیم و مهندسی دانش	
مقطع: کارشناسی ارشد	تاریخ دفاع:بهمن ۱۳۹۵	تعداد صفحات: ۱۳۱
عنوان پایان‌نامه: ماشین‌های یادگیر نهایی		

کلید واژه‌ها:شبکه‌های پیشخور تک لایه پنهان، ماشین یادگیر نهایی، قابلیت تقریب سراسری، قابلیت طبقه‌بندی، رگرسیون

چکیده: سرعت آهسته یادگیری شبکه‌های عصبی پیشخور و تنظیم به طور تکراری پارامترها در دهه‌های گذشته تنگنایی برای کاربردهای شبکه‌های عصبی بوده است. ماشین یادگیر نهایی برای شبکه‌های عصبی پیشخور تک لایه پنهان تعمیم یافته ارائه شد که در آن پارامترهای نورون‌های پنهان به طور تصادفی انتخاب و وزن‌های خروجی از راه تجزیه تعیین می‌شوند. زمان یادگیری ماشین یادگیر نهایی در محاسبه معکوس تعمیم‌یافته مور–پنروز ماتریس خروجی لایه پنهان صرف می‌شود. ماشین یادگیر نهایی یک الگوی یادگیری مؤثر برای کاربردهای طبقه‌بندی و رگرسیون است. ماشین یادگیر نهایی در سال‌های اخیر تبدیل به یک حوزه‌ی پژوهشی داغی شده است، که به پژوهشگران رو به افزایش زمینه‌های پژوهشی سراسر دنیا نسبت داده می‌شود. نظریه‌ها و آزمایشات ثابت کرده است که ماشین یادگیر نهایی دارای ساختاری ساده و قابلیت تقریب سراسری است. هدف از این پایان‌نامه معرفی این الگوریتم جدید، قابلیت‌ها، ویژگی‌ها و روش‌های آموزش آن و هم چنین بررسی ماشین یادگیر نهایی برای کارهای نیمه ناظر و بدون ناظر مبتنی بر چارچوب تنظیم منیفلد است. هم چنین نشان داده می‌شود که تمامی آن‌ها می‌توانند در یک چارچوب واحد قرار گیرند.

پیش‌گفتار

ماشین یادگیر نهایی[۱](ELM) ، ارائه شده توسط گوانگ بین هوانگ[۲] و همکاران [۱] ، یک الگوریتم یادگیری سریع نه تنها برای شبکه‌های عصبی پیشخور تک لایه پنهان [۳] (SLFNs) "تعمیم یافته"بلکه برای شبکه‌های عصبی پیشخور چندلایه پنهان[۴] (MLFNs) " تعمیم یافته" است. از اهداف ELM شکستن موانع بین روش‌های یادگیری سنتی و یادگیری بیولوژیکی است و نیز ارائه مجموعه‌ای از روش‌های یادگیری ماشین است که در آن نورون‌ها در لایه پنهان نیاز نیست تنظیم شوند. در مقایسه با شبکه‌های عصبی سنتی و ماشین بردار پشتیبان، ELM مزایای قابل توجهی هم چون سرعت یادگیری، سهولت پیاده‌سازی و حداقل مداخله انسانی را ارائه می‌کند. با توجه به عملکرد تعمیم قابل توجه و کارآیی پیاده‌سازی، ELM می‌تواند در کاربردهای مختلف هم چون مهندسی پزشکی، بینایی کامپیوتر، شناسایی سیستم، کنترل و رباتیک به کار برده شود. فشرده‌سازی، یادگیری ویژگی، خوشه‌بندی، رگرسیون و طبقه‌بندی پایه‌ای برای یادگیری ماشین هستند و اهداف ELM نیز پیاده‌سازی این پنج عملیات بنیادی یادگیری در معماری‌های ELM است که در این پایان‌نامه خوشه‌بندی، رگرسیون و طبقه‌بندی بررسی می‌شوند [۲] .

این پایان‌نامه شامل ۴ فصل است:

در فصل ۱ تعاریف اولیه، مفاهیم مورد نیاز، علت پیدایش ماشین یادگیر نهایی و ویژگی‌های آن مطرح خواهند شد.

در فصل ۲ انواع ماشین یادگیر نهایی را با توجه به روش آموزش بیان کرده و به الگوریتم‌ها و برنامه‌های متلب آن پرداخته می‌شود.

در فصل ۳ ماشین یادگیر نهایی نیمه ناظر و ماشین یادگیر نهایی بدون ناظر به همراه الگوریتم و برنامه متلب هریک بیان خواهند شد.

در فصل ۴ الگوریتم‌های ماشین یادگیر نهایی (ELM) ، ماشین یادگیر نهایی دو لایه پنهان [۵](TELM) و

---

[۱] Extreme Learning Machine(ELM) [۲] Guang-Bin Huang [۳] Single-Layer Feedforward neural Networks(SLFNs) [۴] Multi-Layer Feedforward neural Networks (MLFNs) [۵] Two-hidden-layer Extreme Learning Machine(TELM)

شبکه عصبی پرسپترون چندلایه [۱](MLP) در کاربرد قطعه‌بندی تصویر بر روی پایگاه داده BSD[۲] اجرا و با هم از نظر معیارهای صحت، حساسیت، نرخ مثبت کاذب، نرخ منفی کاذب و زمان آموزش مقایسه می‌شوند.

مطالب این پایان‌نامه برگرفته از مقالات زیر می‌باشد:

1. Qu, B.Y., Lang, B.F., Liang, J.J., Qin, A.K., and Crisalle, O.D. Two-hidden-layer extreme learning machine for regression and classification. *Neurocomputing.*, 175(PA):826–834, January 2016.

2. Li, Junpeng, Hua, Changchun, Tang, Yinggan, and Guan, Xinping. A fast training algorithm for extreme learning machine based on matrix decomposition. *Neurocomputing*, 173, Part 3:1951 – 1958, 2016.

3. Huang, Gao, Huang, Guang-Bin, Song, Shiji, and You, Keyou. Trends in extreme learning machines: A review. *Neural Networks*, 61:32 – 48, 2015.

4. Ye, Yibin and Qin, Yang. Qr factorization based incremental extreme learning machine with growth of hidden nodes. *Pattern Recognition Letters*, 65(C):177–183, November 2015.

5. Huang, G., Song, S., Gupta, J. N. D., and Wu, C. Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics*, 44(12):2405–2417, Dec 2014.

[۱]Multi Layer Perceptron(MLP) [۲]Berkeley Segmentation Dataset(BSD)

نمادگذاری ریاضی

در این پایان‌نامه سعی شده است برای تعریف متغیرها و پارامترها از یک نمادگذاری واحد استفاده شود. به این منظور شیوه عمل مرجع [۳] ملاک عمل قرار گرفته است.

ماتریس . با حروف بزرگ برجسته[۱] و ایستاده[۲] هم چون $\mathbf{X}$ نشان داده می‌شود.

بردار . با حروف کوچک برجسته و ایستاده هم چون $\mathbf{x}$ نشان داده می‌شود. تمام بردارها به طور پیش‌فرض ستونی هستند. علامت T ترانهاده بردار یا ماتریس است، به طوری که $\mathbf{x}^T$ یک بردار سطری خواهد بود. عناصر یک بردار با x مشخص می‌شوند. بنابراین x نشان‌دهنده اسکالر است.

N . تعداد نمونه‌های آموزشی

D . تعداد نورون‌های لایه ورودی یا به تعبیری دیگر بعد نمونه‌های آموزشی

M . تعداد نورون‌های لایه پنهان

K . تعداد نورون‌های لایه خروجی

نمادهایی هستند که در ماشین یادگیر نهایی مورد بررسی در این پایان‌نامه استفاده می‌شوند.

به طور کلی بردار و ماتریس‌های استفاده شده در پایان‌نامه به شرح زیر هستند

ماتریس نمونه‌های آموزشی(نمونه‌های ورودی شبکه)

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times D}$$

[۱] bold [۲] Roman

ماتریس خروجی مطلوب (هدف) نمونه‌های آموزشی

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1K} \\ t_{21} & t_{22} & \dots & t_{2K} \\ \vdots & \vdots & \vdots & \vdots \\ t_{N1} & t_{N2} & \dots & t_{NK} \end{bmatrix} \in \mathbb{R}^{N \times K}$$

ماتریس وزن‌های ورودی(بین لایه ورودی و لایه پنهان در شبکه)

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_M^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1D} \\ w_{21} & w_{22} & \dots & w_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M1} & w_{M2} & \dots & w_{MD} \end{bmatrix} \in \mathbb{R}^{M \times D}$$

بردار بایاس

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix}$$

ماتریس وزن‌های خروجی(بین لایه پنهان و لایه خروجی)

از آن جایی که نمی‌توان β (وزن‌های خروجی) را به صورت برجسته و ایستاده نوشت لذا از $\boldsymbol{\beta}$ برای نمایش بردار β و از $\boldsymbol{\beta}$ برای ماتریس β استفاده می‌شود.

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \boldsymbol{\beta}_2^T \\ \vdots \\ \boldsymbol{\beta}_M^T \end{bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1K} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2K} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{M1} & \beta_{M2} & \dots & \beta_{MK} \end{bmatrix} \in \mathbb{R}^{M \times K}$$

ماتریس خروجی واقعی(شبکه)

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1K} \\ y_{21} & y_{22} & \cdots & y_{2K} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NK} \end{bmatrix} \in \mathbb{R}^{N \times K}$$

علاوه بر این، در این پایان‌نامه

- منظور از نرم یک ماتریس $\mathbf{A}_{m \times n}$ ($\|\mathbf{A}\|$)، نرم فروبینیوس یا همان نرم اقلیدسی است که با $\|\mathbf{A}\|_F$ نشان داده می‌شود و برابر است با ریشه دوم مجموع توان دوم تمام درایه‌های ماتریس $\mathbf{A}$ و به صورت زیر نوشته می‌شود

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^2}$$

- نماد $< \ . \ >$ نشان‌دهنده ضرب نقطه‌ای(داخلی) است، برای ضرب داخلی دو بردار هم چون $\mathbf{w}$ و $\mathbf{x}$ به جای $\mathbf{w}^T \cdot \mathbf{x}$ از رابطه $\mathbf{w}^T \mathbf{x}$ استفاده می‌شود.

فصل ۱

ماشین یادگیر نهایی

این فصل دربرگیرنده مفاهیم، لم‌ها و قضایای مقدماتی مورد نیاز پایان‌نامه است. بخش اول به تعریف پرسپترون چندلایه ، آموزش آن با قاعده پس انتشار و اجرای برنامه متلب آن بر روی یک مجموعه داده پرداخته است. بخش دوم تعاریف، لم‌ها و قضایای مورد نیاز را به ترتیب آورده‌ایم. بخش سوم را به تعریف ماشین یادگیر نهایی، ویژگی‌ها، نظریه‌ها و کاربرد طبقه‌بندی آن اختصاص داده‌ایم. در انتهای این فصل هم مثالی را برای مقایسه روش پس انتشار و ماشین یادگیر نهایی مورد بررسی قرار داده‌ایم.

۱-۱ مقدمه

شبکه‌های عصبی مصنوعی[۲] (ANNs) یا به زبان ساده‌تر شبکه‌های عصبی، روش‌های محاسباتی جدید برای یادگیری ماشین، نمایش دانش و اعمال دانش به دست آمده در بیش‌بینی پاسخ‌های خروجی سامانه‌های پیچیده می‌باشند. در حقیقت، هدف از ایجاد یک شبکه عصبی نرم افزاری، بیش از آن که شبیه‌سازی مغز انسان باشد، ایجاد مکانیسمی برای حل مسائل مهندسی با الهام از الگوی رفتاری شبکه‌های بیولوژیک است. تاکنون مدل‌های مختلف با ساختار و الگوریتم‌های متنوعی از شبکه‌های عصبی ارائه شده است و هر چند این مدل‌ها با یکدیگر تفاوت دارند، اما تمام این مدل‌ها یک هدف مشترک را دنبال می‌کنند. یک شبکه عصبی از اجزای ساده (نورون) و زنجیره‌ای تشکیل می‌شود و دارای ویژگی‌های زیر است [۴]:

- قابلیت یادگیری و تطبیق پذیری
- قابلیت تعمیم پذیری

[۲] Artificial Neural Networks (ANNs)

- پردازش موازی
- مقاوم بودن
- قابلیت تقریب سراسری

قابلیت یادگیری و تطبیق پذیری قابلیت یادگیری یعنی توانایی تنظیم پارامترهای شبکه عصبی. برای این منظور نمونه‌های اولیه را به شبکه اعمال می‌کنند، شبکه پارامترها را براساس این نمونه‌ها تنظیم می‌کند. اگر نمونه‌های جدید به این شبکه که به این طریق آموزش دیده اعمال شوند، خروجی مناسب را با درصد خطای کوچک می‌توان به دست آورد. به این ترتیب شبکه‌های عصبی می‌توانند با تغییر شرایط به صورت هوشمندانه خود را تطبیق یا اصلاح نمایند.

قابلیت تعمیم پذیری: پس از آن که نمونه‌های اولیه به شبکه آموزش داده شد، شبکه می‌تواند در مقابل ورودی‌های آموزش داده نشده (ورودی‌های جدید) قرار گیرد و یک خروجی مناسب تولید نماید. این خروجی براساس مکانیسم تعمیم، که چیزی جز فرآیند درون‌یابی نیست به دست می‌آید.

پردازش موازی: هنگامی که شبکه عصبی در قالب سخت افزار پیاده‌سازی می‌شود نورون‌هایی که در یک لایه قرار می‌گیرند می‌توانند به طور همزمان به ورودی‌های آن لایه پاسخ دهند. این ویژگی باعث افزایش سرعت پردازش می‌شود. در واقع در چنین سیستمی وظیفه کلی پردازش، بین پردازنده‌های کوچکتر مستقل از یکدیگر توزیع می‌گردد.

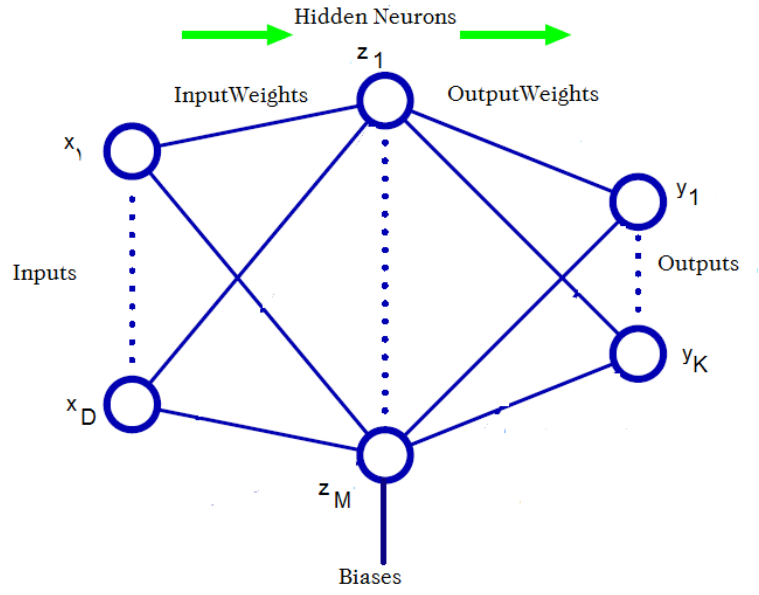
مقاوم بودن: در یک شبکه عصبی هر نورون به طور مستقل عمل می‌کند و رفتار کلی شبکه برآیند رفتارهای محلی نورون‌های متعدد است. این ویژگی باعث می‌شود تا خطاهای محلی از چشم خروجی نهایی به دور بماند. به عبارت دیگر نورون‌ها در یک روند همکاری خطاهای محلی یکدیگر را تصحیح می‌کنند. این خصوصیت باعث افزایش مقاوم بودن در سیستم می‌گردد.

قابلیت تقریب سراسری[۱]**:** شبکه‌های عصبی با یک یا چندلایه پنهان به شرط آن که لایه‌ها تعداد کافی نورون‌های پنهان داشته باشند، می‌توانند هر تابع غیرخطی قطعه به قطعه پیوسته [۲] را تخمین بزنند.

اجزای شبکه عصبی : ورودی و خروجی، نورون، وزن و توابع فعالیت[۳] (توابع انتقال) هستند.

---

[۲] nonlinear piecewise continuous function [۳] activation functions



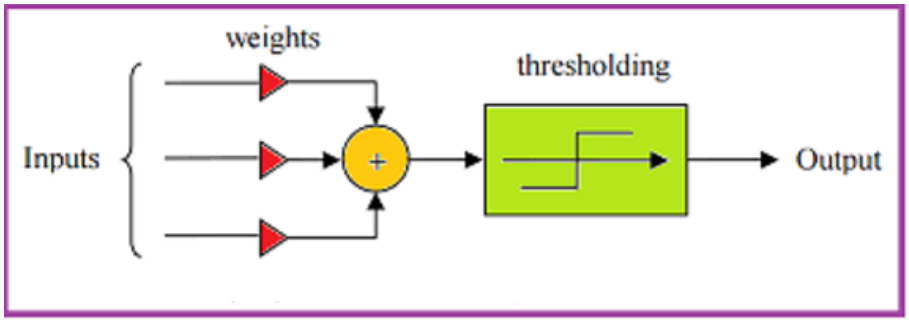
شکل ۱-۱: شمای کلی یک شبکه عصبی پیشخور تک لایه پنهان [۳]

سبک‌های معماری شبکه عصبی

طرح اتصالات بین نورون‌ها در یک شبکه عصبی به سبک معماری شبکه عصبی معروف است. از حیث سبک معماری، انواع مختلفی از شبکه‌های عصبی وجود دارند که در یک طبقه‌بندی کلی به مدل‌های ایستا و پویا تقسیم می‌شوند. در مدل‌های ایستا مسیر پردازش اطلاعات از لایه ورودی به لایه خروجی است، بدون این که بازگشتی در سیستم ارتباطی لایه‌ها وجود داشته باشد؛ در حالی که در مدل‌های پویا مسیرهای بازگشتی از لایه خروجی یا لایه‌های میانی به لایه ورودی نیز وجود دارد. شبکه‌های ایستا را شبکه‌های «پیشخور» و شبکه‌های پویا را شبکه‌های «برگشتی» یا «پسخور» نیز می‌گویند. بنابراین تفاوت شبکه‌های پسخور با شبکه‌های پیشخور در این است که در شبکه‌های پسخور حداقل یک سیگنال برگشتی از یک نورون به همان نورون یا نورون‌های همان لایه یا لایه قبل وجود دارد [۴].

شبکه‌های عصبی پیشخور در طول چند دهه گذشته بسیار مورد مطالعه قرار گرفته‌اند. شبکه‌های عصبی پیشخور دارای یک یا چندلایه پنهان‌اند. شبکه‌های عصبی پیشخور تک لایه پنهان (SLFNs) یکی از مدل‌های شبکه عصبی بسیار محبوب دارای ساختاری ساده شامل یک لایه ورودی، یک لایه پنهان و یک لایه خروجی هستند. در شکل ۱-۱ یک شبکه عصبی پیشخور تک لایه پنهان دیده می‌شود.

در یک شبکه عصبی چندلایه پنهان، هر لایه ماتریس وزن ورودی (W) ، بردار بایاس (b) خود را دارد. لایه‌های مختلف می‌توانند تعداد نورون‌های متفاوتی داشته باشند. نورون‌های لایه ورودی صرفاً وظیفه توزیع



شکل ۱-۲: نمای مدل اصلی نورون

مقادیر ورودی به لایه بعدی را برعهده دارند و بنابراین هیچ محاسبه‌ای را انجام نمی‌دهند. لایه‌ای که خروجی آن خروجی شبکه است، لایه خروجی نامیده می‌شود. لایه‌های دیگر لایه‌های پنهان نام دارند. شبکه‌های چندلایه قدرتمندتر از شبکه‌های تک لایه هستند. برای نمونه، یک شبکه با دو لایه پنهان که شامل لایه اول سیگموئید و لایه دوم خطی می‌باشد، می‌تواند به منظور تقریب اکثر توابع اختیاری آموزش داده شود. بیشتر شبکه‌های کاربردی تنها دو یا سه لایه دارند.

۱-۱-۱ پرسپترون تک لایه و چندلایه

پرسپترون [۵] یک نوع شبکه عصبی است که در سال ۱۹۵۷به وسیله فرانک روزن بلات[۱] ابداع شد. از آن جایی که در پرسپترون نورون‌های پایه (مطابق شکل ۱-۲) به طریقی ساده به یکدیگر متصل‌اند می‌توان آن را ساده‌ترین نوع شبکه عصبی در نظر گرفت که برخی از ویژگی‌های سیستم‌های عصبی واقعی در آن به کار رفته و برخی دیگر از آن‌ها نادیده گرفته شده است. پرسپترون یک نوع تفکیک کننده دودویی است که مجموع وزنی ورودی‌ها را محاسبه می‌کند و با مقدار آستانه نورون موردنظر مقایسه می‌کند، اگر از میزان آستانه بیشتر بود خروجی نورون یک و در غیر این صورت صفر می‌شود (این فرآیند در شکل ۱-۲ مشاهده می‌شود).

برای بیان واضح‌تر موضوع، تابع $f : \mathbb{R}^D \to \mathbb{R}$ ورودی خود ($\mathbf{x}$ ، یک بردار از اعداد حقیقی) را به مقدار خروجی ($f(\mathbf{x})$ ، یک اسکالر از نوع اعداد حقیقی) که به صورت زیر حساب می‌شود، متناظر می‌کند

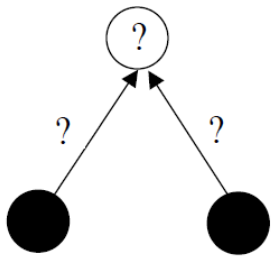
$$f(\mathbf{x}) = \begin{cases} ۱ & \text{if} \quad \mathbf{w}^T \cdot \mathbf{x} + b > ۰ \\ ۰ & \text{otherwise} \end{cases}$$

که در آن $\mathbf{w}$ یک بردار از وزن‌های ورودی با مقادیر حقیقی است، $< \ . \ >$ ضرب نقطه‌ای(داخلی) است (که مجموع وزن‌دار را محاسبه می‌کند)، مقدار آستانه برابر صفر و b بایاس نورون که یک جمله ثابت است و به ورودی

[۱] Frank Rosen Blot

XOR

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	0



شکل ۱-۳: جدول تابع یای حذفی [۵]

بستگی ندارد.

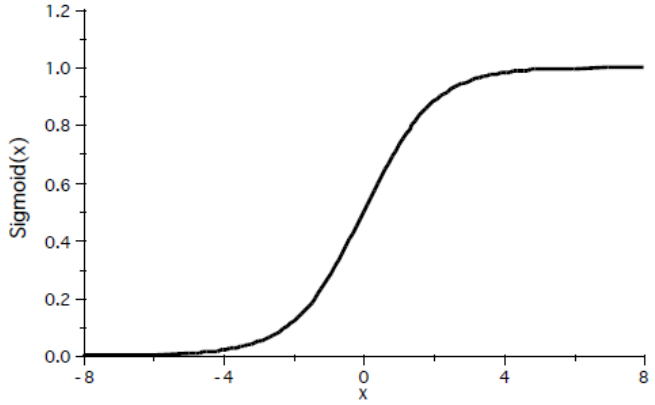
در نظر بگیرید که پرسپترون به دنبال خطی است که طبقه‌ها را تفکیک کند. پرسپترون به راحتی می‌تواند طبقاتی را که در دوسوی خط قرار دارند تفکیک کند، لیکن حالت‌های فراوانی است که جدایی طبقات بسیار پیچیده‌تر است مثلاً مسئله یای حذفی[۱](XOR) همان طور که در شکل ۱-۳ دیده می‌شود پرسپترون باید درنهایت یاد بگیرد که اگر ورودی اول $in_۱$ فعال و ورودی دوم $in_۲$ خاموش و یا به عکس $in_۲$ فعال و $in_۱$ خاموش بود جواب یک بدهد و اگر هر دو فعال یا خاموش بودند جواب صفر بدهد.

پرسپترون تک لایه‌ای با وجود سادگی مدل هیچ مسئله جدایی ناپذیر خطی را نمی‌تواند حل کند. رفع این نیاز برای نخستین بار در سال ۱۹۸۶ توجه محافل علمی را به خود جلب کرد، زمانی که رومل هارت[۲] و مک کللند[۳] صورت جدیدی از مدل پرسپترون را به نام پرسپترون چندلایه‌ای (MLP) که به صورت لایه‌ای منظم شده بود و سه لایه (ورودی، پنهان و خروجی) داشت، کشف کردند. هر نورون در لایه پنهان و لایه خروجی مانند یک پرسپترون عمل می‌کند. لازم است که از تابع آستانه‌ای غیرخطی استفاده شود زیرا اگر لایه‌های پرسپترون همه از توابع خطی استفاده کنند توان آن‌ها از یک مدل پرسپترون تک لایه‌ای فراتر نمی‌رود. دلیل آن این است که هر لایه پرسپترون صرفاً عملیاتی خطی بر ورودی‌های خود انجام می‌دهد و عملیات خطی نهایتاً می‌توانند با هم به صورت یک عملیات واحد ترکیب شوند. با تغییر تابع غیرخطی از صورت پلکانی به سیگموئید و اضافه کردن یک لایه پنهان به ناچار باید قاعده فراگیری مدل تغییر کند. ضابطه و شکل تابع سیگموئید در ادامه آمده است

$$Sigmoid(\mathbf{x}) = \frac{۱}{۱ + exp(-\mathbf{x})}$$

در بخش بعدی برای پرسپترون چندلایه قاعده فراگیری جدید با جزئیات بیشتر بررسی می‌شود.

[۱] exclusive OR (XOR) [۲] Rommel Hart [۳] McClelland



شکل ۱-۴: تابع فعالیت سیگموئید

۱-۱-۲ قاعده پس انتشار

قاعده فراگیری پرسپترون چندلایه "قاعده پس انتشار" یا "قاعده کلی دلتا" نامگذاری شده و در سال ۱۹۸۶ توسط رومل هارت، مک کللند و ویلیامز[۱] پیشنهاد شده است و می‌تواند در آموزش شبکه‌های عصبی پرسپترون چندلایه (MLP) که قادر به حل مسائل غیرخطی هستند به کار رود. شبکه‌های عصبی پرسپترون چندلایه که با الگوریتم پس انتشار[۲](BP) آموزش دیده باشد در حال حاضر پرکاربردترین شبکه عصبی است که به طور گسترده استفاده می‌شود [۵].

نحوه عمل پرسپترون چندلایه‌ای مشابه پرسپترون تک لایه‌ای است. بدین صورت که نمونه‌ای به شبکه عرضه می‌شود و خروجی آن محاسبه می‌گردد، مقایسه خروجی واقعی و خروجی مطلوب باعث می‌گردد که ضرایب وزنی شبکه تغییر یابد به طوری که در دفعات بعد خروجی درست‌تری حاصل شود. قاعده فراگیری، روش تنظیم ضرایب وزنی شبکه را بیان می‌کند. برای موفق شدن در آموزش شبکه باید خروجی آن به تدریج به خروجی مطلوب نزدیک شود. به عبارت دیگر باید میزان تابع خطا را به طور دائم کاهش داد. برای این منظور ضرایب وزنی خطوط ارتباطی نورون‌ها با استفاده از قاعده کلی دلتا تنظیم می‌شود. قاعده دلتا[۳] مقدار تابع خطا را محاسبه کرده و آن را به عقب از یک لایه به لایه پیشین آن انتشار می‌دهد. عبارت "پس انتشار" به این علت است. ضرایب وزنی هر نورون جداگانه تنظیم می‌شود و بدین صورت میزان خطا کاهش می‌یابد. این عمل در مورد نورون‌های لایه خروجی ساده است زیرا خروجی واقعی و مطلوب آن‌ها مشخص است و در بخش ۱-۱-۳ بیان می‌شود، ولی در مورد لایه پنهان چندان روشن نیست. این گمان می‌رود که ضرایب وزنی نورون‌های پنهان که با میزان خطایی بزرگ به نورون‌های خروجی مرتبط هستند باید بیش‌تر از نورون‌های پنهان که به نورون‌های مرتبط آن‌ها خروجی تقریباً صحیحی دارند تغییر یابد. در واقع ریاضیات نشان می‌دهد که ضرایب نورون‌ها باید به تناسب میزان خطای

---

[۱] Williams [۲] Back Propagation (BP) [۳] delta-rule

نورونی که به آن متصل‌اند تغییر کند. بنابراین می‌توان با انتشار خطا به عقب ضرایب وزنی خطوط ارتباطی تمام لایه‌ها را به درستی تنظیم کرد. به این طریق تابع خطا کاهش و شبکه آموزش می‌یابد.

۳-۱-۱ به روز رسانی ضرایب وزنی با قاعده پس انتشار

در این بخش هدف به دست آوردن وزن‌های ورودی و خروجی در الگوریتم پس انتشار است و نحوه به روز رسانی این وزن‌ها بیشتر توضیح داده می‌شود. ابتدا فرضیات زیر را درنظر بگیرید [۵]

E_p: تابع خطای نمونه آموزشی p که تفاوت خروجی واقعی و خروجی مطلوب را نشان می‌دهد.

t_{pj}: خروجی مطلوب نمونه p در نورون j

o_{pj}: خروجی واقعی نمونه p در نورون j

w_{ij}: ضریب وزنی خط ارتباطی نورون i به نورون j

فرض کنید تابع خطا متناسب با مجذور تفاوت خروجی واقعی و خروجی مطلوب برای تمامی نمونه‌های عرضه شده به شبکه به صورت زیر باشد

$$E_p = \frac{۱}{۲} \sum_j (t_{pj} - o_{pj})^۲ \qquad (۱-۱)$$

میزان انگیزش هر نورون یا به عبارتی ورودی خالص نورون j برای نمونه p که با net_{pj} نشان داده می‌شود، می‌تواند به صورت زیر نوشته شود که در آن o_{pi} خروجی هر لایه است که به عنوان ورودی لایه بعد استفاده می‌شود.

$$net_{pj} = \sum_i w_{ij} o_{pi} \qquad (۲-۱)$$

خروجی هر نورون j برابر با میزان تابع آستانه f_j است که بر جمع وزنی ورودی‌های آن نورون عمل می‌کند. این تابع در مورد پرسپترون تابع پلکانی بود در حالی که در مورد پرسپترون چندلایه‌ای تابع سیگموئید است.

$$o_{pj} = f_j(net_{pj}) \qquad (۳-۱)$$

برای شبکه چندلایه، خطا تابع صریحی از وزن‌ها در لایه‌های پنهان نمی‌باشد، بنابراین مشتقات به آسانی قابل

محاسبه نیستند. با استفاده از قاعده زنجیره‌ای مشتق‌گیری می‌توان رابطه زیر را نوشت

$$-\frac{\partial E_p}{\partial w_{ij}} = -\frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \qquad (۴-۱)$$

بادر نظر گرفتن رابطه (۱-۲) و جایگزینی آن در عبارت دوم رابطه (۱-۴) نتیجه می‌شود

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial \sum_k w_{kj} o_{pk}}{\partial w_{ij}} = \sum_k \frac{\partial w_{kj}}{\partial w_{ij}} o_{pk} = o_{pi}$$

زیرا در همه موارد $\frac{\partial w_{kj}}{\partial w_{ij}}$ برابر صفر است مگر هنگامی که $k = i$ باشد که در آن صورت برابر یک است. مقدار تغییر خطا به صورت تابعی از ورودی خالص هر نورون می‌تواند به شکل زیر تعریف شود

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj} \qquad (۵-۱)$$

بنابراین رابطه (۱-۴) به صورت زیر در می‌آید

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} o_{pi}$$

درنتیجه کاهش مقدار خطا E_p به معنی تغییر ضرایب وزنی متناسب با $\delta_{pj} o_{pi}$ خواهد بود. به عبارت دیگر

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi}$$

بعد از یافتن مقدار δ_{pj} برای هر نورون می‌توان E را کاهش داد. با استفاده از رابطه (۱-۵) و قاعده زنجیره‌ای می‌توان نوشت

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \qquad (۶-۱)$$

عبارت دوم رابطه فوق را در نظر بگیرید. از رابطه (۱-۳) نتیجه می‌شود

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f_j'(net_{pj})$$

اکنون عبارت اول در رابطه (۱-۶) را در نظر بگیرید. با بهره گیری از رابطه (۱-۱) می‌توان مشتق E_p را نسبت به

o_{pj} به دست آورد

$$-\frac{\partial E_p}{\partial o_{pj}} = -(-(t_{pj} - o_{pj}))$$

عبارت فوق برای نورون‌های لایه خروجی مفید است، زیرا خروجی‌های مطلوب و خروجی‌های واقعی هر دو معلوم‌اند، لیکن برای نورون‌های لایه پنهان مناسب نیست، زیرا مقدار خروجی‌های مطلوب آن‌ها در دست نیست. بنابراین در مورد نورون‌های لایه خروجی به رابطه زیر خواهیم رسید

$$\delta_{pj} = f'_j(net_{pj})(t_{pj} - o_{pj}) \qquad (۷-۱)$$

اگر نورون j جزء نورون‌های **لایه پنهان** باشد، با استفاده از قاعده زنجیره‌ای مشتق به طریق زیر عمل می‌شود

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}}$$

$$= (\sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}}) \frac{\partial o_{pj}}{\partial net_{pj}}$$

$$= (\sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial \sum_i w_{ik} o_{pi}}{\partial o_{pj}}) \frac{\partial o_{pj}}{\partial net_{pj}}$$

$$= (\sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\sum_i \partial o_{pi} w_{ik}}{\partial o_{pj}}) \frac{\partial o_{pj}}{\partial net_{pj}}$$

$$= (\sum_k \delta_{pk}\, w_{jk}) \frac{\partial o_{pj}}{\partial net_{pj}}$$

بنابراین در مورد نورون‌های لایه پنهان رابطه زیر برقرار خواهد بود

$$\delta_{pj} = (\sum_k \delta_{pk} w_{jk})\, f'_j(net_{pj})$$

رابطه فوق نشان‌دهنده مقدار تغییرات تابع خطا نسبت به ضرایب وزنی در شبکه بوده و روشی را برای کاهش تابع خطا فراهم می‌کند. مقدار تابع متناسب با مقادیر خطای δ نورون‌های بعدی است. بنابراین مقادیر خطا باید ابتدا در نورون‌های لایه خروجی محاسبه شوند (با استفاده از رابطه (۷-۱)) و آن گاه به لایه‌های قبلی شبکه برای تغییر ضرایب وزنی نورون‌های پیشین پس رانده شوند. عمل جمع در مورد k نورون واقع در لایه بعد از نورون j صورت می‌گیرد.

یکی از مزایای استفاده از تابع سیگموئید به عنوان تابع غیرخطی شباهت آن به تابع پلکانی است و هم چنین

یکی از دلایل عمده استفاده از تابع سیگموئید سادگی مشتق آن است که استفاده از روش پس انتشار خطا را بسیار ساده‌تر می‌کند. در تابع سیگموئید مقدار خروجی o_{pj} توسط رابطه زیر به دست می‌آید

$$o_{pj} = f(net) = \frac{۱}{۱ + e^{-\mathcal{K}net}}$$

ورودی این تابع هر مقداری بین منفی بی‌نهایت تا مثبت بی‌نهایت است و مقدار تابع دارای برد $۰ < f(net) < ۱$ می‌باشد. $\mathcal{K}$ عدد ثابت مثبتی است که گستره تابع را تنظیم می‌کند. مقادیر بزرگ $\mathcal{K}$ تابع را فشرده‌تر می‌کند به طوری که اگر $\mathcal{K}$ به سمت بی‌نهایت میل کند $f(net)$ منطبق بر تابع پلکانی خواهد شد. چنان چه از آن نسبت به (net) مشتق گرفته شود نتیجه زیر حاصل می‌شود

$$f'(net) = \frac{\mathcal{K}e^{-\mathcal{K}net}}{(۱ + e^{-\mathcal{K}net})^{۲}} = \mathcal{K}\, f(net)(۱ - f(net)) = \mathcal{K}o_{pj}(۱ - o_{pj})$$

بنابراین مشتق تابع صرفاً تابع ساده‌ای از خروجی است. در نتیجه به روز رسانی ضرایب وزنی مطابق رابطه زیر خواهد شد

$$w_{ij}(t + ۱) = w_{ij}(t) + \eta\delta_{pj}o_{pi} \qquad (۸-۱)$$

که $w_{ij}(t)$ نشان‌دهنده ضرایب وزنی از نورون i به نورون j در زمان t، η طول گام یادگیری و δ_{pj} نمایانگر خطای مربوط به نمونه p در نورون j است. هر قدر طول گام یادگیری کوچک‌تر انتخاب گردد تغییرات ایجاد شده در پارامترهای شبکه پس از هر مرحله تکرار الگوریتم BP کوچک‌تر خواهد بود که این خود منجر به هموار گشتن مسیر حرکت پارامترها به سمت مقادیر بهینه در فضای پارامترها می‌گردد. این مسئله موجب کندتر گشتن الگوریتم BP می‌گردد. بر عکس با افزایش طول گام یادگیری ، اگرچه نرخ یادگیری و سرعت یادگیری الگوریتم BP افزایش می‌یابد لیکن تغییرات زیادی در پارامترهای شبکه از هر تکراربه تکرار بعد ایجاد می‌گردد که گاهی اوقات موجب ناپایداری و نوسانی شدن شبکه می‌شود که به اصطلاح می‌گویند پارامترهای شبکه واگرا شده‌اند. در نهایت

به روز رسانی ضرایب وزنی بین لایه پنهان و لایه خروجی (وزن‌های خروجی) به صورت زیر خواهد بود

$$w_{ij}(t + ۱) = w_{ij}(t) + \eta\left(\mathcal{K}o_{pj}(۱ - o_{pj})(t_{pj} - o_{pj})\right)o_{pi}$$

به روز رسانی ضرایب وزنی بین لایه ورودی و لایه پنهان (وزن‌های ورودی) به صورت زیر خواهد بود

$$w_{ij}(t+۱) = w_{ij}(t) + \eta \left(\mathcal{K} o_{pj}(۱ - o_{pj}) \sum_k \delta_{pk} w_{jk} \right) o_{pi}$$

با استفاده از این فرمول‌ها در الگوریتم پس انتشار مقدار خطا کاهش می‌یابد.

۱-۱-۴ آموزش پرسپترون چندلایه با قاعده پس انتشار

چارچوب کلی قاعده پس انتشار در الگوریتم ۱-۱ آمده است. در این بخش قاعده پس‌انتشار روی مجموعه داده گل‌های زنبق[۱] اعمال می‌شود تا با دیدن خروجی، نحوه کار این قاعده بهتر درک شود (برنامه متلب صفحه ۹۷) . همان طور که در برنامه دیده می‌شود وزن‌های ورودی بین لایه ورودی و لایه پنهان و وزن‌های خروجی بین لایه پنهان و لایه خروجی به طور تصادفی تولید شده‌اند و به روز رسانی می‌شوند.

۱-۱-۵ مشکلات

از قاعده یادگیری پس انتشار خطا (BP) ، برای آموزش شبکه‌های عصبی پرسپترون چندلایه استفاده می‌شود و علی‌رغم موفقیت‌های آن در این زمینه، هنوز چندین مشکل اصلی وجود دارد:

- الگوریتم پس انتشار خطا ممکن است به نقاط مینیمم محلی در فضای پارامتر همگرا شود. بنابراین زمانی که الگوریتم پس انتشار خطا همگرا می‌شود، نمی‌توان مطمئن شد که به یک جواب بهینه رسیده است. بهتر است که چندین وضعیت اولیه متفاوت را به منظور اطمینان از این که جواب بهینه به دست آمده است، امتحان کرد.
- سرعت همگرایی الگوریتم پس انتشار خطا خیلی آهسته است.
- همگرایی الگوریتم پس انتشار خطا به انتخاب مقادیر اولیه وزن‌های شبکه، بردارهای بایاس و پارامترهای موجود در الگوریتم مانند طول گام یادگیری وابسته است.

برخی از این مشکلات در ELM حل شده است که در بخش بعد بیان خواهد شد.

[۱] Iris

الگوریتم ۱-۱ الگوریتم آموزش پرسپترون چندلایه با قاعده پس انتشار [۵]

ورودی: بردار p امین نمونه آموزشی $\mathbf{x}_p = \left[\begin{array}{cccc} x_{p0} & x_{p1} & x_{p2} & \dots & x_{p,D-1} \end{array} \right]^T$ و بردار خروجی مطلوب متعلق به p امین نمونه آموزشی $\mathbf{t}_p = \left[\begin{array}{cccc} t_{p0} & t_{p1} & t_{p2} & \dots & t_{p,K-1} \end{array} \right]^T$ و مقدار η و $\mathcal{K}$

خروجی: ضرایب وزنی شبکه

۱: مقادیر اولیه ضرایب وزنی (وزن‌های ورودی بین لایه ورودی و لایه پنهان، وزن‌های خروجی بین لایه پنهان و لایه خروجی) و بایاس‌ها را به طور تصادفی انتخاب کنید (تمام وزن‌ها و بایاس‌ها را برابر با اعداد تصادفی کوچک قرار دهید).

۲: ورودی و خروجی را به شبکه عرضه کنید. D تعداد عناصر بردارهای نمونه آموزشی و K تعداد عناصر بردارهای خروجی هدف است. در مسئله طبقه بندی تمام عناصر $\mathbf{t}_p$ برابر با صفر قرار داده می‌شود مگر یکی از عناصر که برابر با یک است و آن طبقه‌ای را نشان می‌دهد که $\mathbf{x}_p$ در آن قرار دارد.

۳: هر لایه جمع وزنی ورودی‌هایش را به صورت زیر محاسبه کرده و به عنوان ورودی لایه بعدی انتقال می‌دهد.

$$y_{pj} = f[\sum_{i=0}^{D-1} \mathbf{w}_i^T \mathbf{x}_i]$$

منظور از y_{pj} در لایه آخر خروجی واقعی و همان o_{pj} ذکرشده در بخش ۱-۱-۳ است.

۴: مطابق رابطه (۱-۸) ضرایب وزنی را تنظیم کنید. در این رابطه منظور از o_{pi} ورودی هر لایه است، به تعبیر دیگر در روند تنظیم وزن‌های بین لایه پنهان و لایه خروجی $f[\sum_{i=0}^{D-1} \mathbf{w}_i^T \mathbf{x}_i]$ (خروجی لایه پنهان) و تنظیم وزن‌های بین لایه ورودی و لایه پنهان نمونه آموزشی $(\mathbf{x}_i)$ همان o_{pi} است. ابتدا از لایه خروجی شروع کنید و به عقب برگردید.

مقدار دلتا نورون‌های لایه خروجی

$$\delta_{pj} = k o_{pj}(1 - o_{pj})(t_{pj} - o_{pj})$$

مقدار دلتا نورون‌های لایه پنهان

$$\delta_{pj} = k o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{jk}$$

عمل جمع در مورد k نورون در لایه بعد از نورون j صورت می‌گیرد.

ماشین یادگیر نهایی

باورها و فلسفه انسان در مورد یادگیری ماشین و یادگیری بیولوژیکی در نهایت منجر به روش‌های جدید با نام ماشین یادگیر نهایی (ELM) شده است. در سال ۲۰۰۴ گوانگ بین هوانگ و همکاران، با هدف اجتناب از یک روش آموزش تکراری وقت‌گیر، کاهش هزینه‌های محاسباتی و بهبود عملکرد تعمیم یک الگوریتم یادگیری ماشین مفید - ماشین یادگیر نهایی (ELM) - را برای آموزش شبکه‌های عصبی پیشخور تک لایه پنهان تعمیم یافته پیشنهاد کرده‌اند.

'Extreme' به معنی حرکت روش‌های یادگیری ماشین مصنوعی به فراتر و حرکت به سمت یادگیری مشابه مغز است. اهداف ELM شکستن موانع بین روش‌های یادگیری ماشین و یادگیری بیولوژیکی است. ماشین یادگیر نهایی (ELM) نشان‌دهنده مجموعه‌ای از روش‌های یادگیری ماشین است که در آن نورون‌های پنهان با توجه به نظریه تعمیم شبکه عصبی، نظریه کنترل، نظریه ماتریس و نظریه سیستم خطی نیاز ندارند تنظیم شوند [۶].

۱-۲ پیش‌نیازها

برای بیان و اثبات قضایای مربوطه نیاز به مفاهیم و تعاریفی است که در ابتدا ذکر خواهد شد.

۱-۲-۱ تعاریف

تعریف ۱-۲-۱ (ماتریس منفرد). ماتریس مربعی که دترمینان آن برابر صفر باشد، یعنی $|A|= ۰$. ماتریس نامنفرد[۱] یا وارون پذیر. اگر در یک ماتریس مربعی دترمینال آن صفر نباشد به آن ماتریس نامنفرد می‌گویند، یعنی $|A|\neq ۰$ [۷].

تعریف ۱-۲-۲ (ماتریس هرمیتی). ماتریس مربعی که ترانهاده مزدوج مختلط آن با خودش برابر باشد. به تعبیر ریاضی، یعنی $\mathbf{A}^H = \mathbf{A}$ که $\mathbf{A}$ ماتریس مختلط و H نماد ترانهاده مزدوج است. اگر $\mathbf{A}$ ماتریس حقیقی باشد، آن گاه $\mathbf{A}$ را هرمیتی گوییم در صورتی که $\mathbf{A}^T = \mathbf{A}$. به عنوان نمونه ماتریس زیر یک ماتریس هرمیتی است.

$$\begin{bmatrix} ۲ & ۲-i & ۱ \\ ۲+i & ۳ & i \\ ۱ & -i & ۴ \end{bmatrix}$$

[۱] nonsingular matrix

تعریف ۱-۲-۳. ماتریس $\mathbf{A}$ را قطری شدنی گوییم اگر ماتریس وارون پذیر $\mathbf{P}$ وجود داشته باشد به طوری که در رابطه زیر صدق کند

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{B}$$

که در آن $\mathbf{B}$ یک ماتریس قطری است.

تعریف ۱-۲-۴ (نیمه معین مثبت). ماتریس حقیقی و متقارن $\mathbf{A}$ نیمه معین مثبت نامیده می‌شود هرگاه به ازای هر بردار غیرصفر $\mathbf{x}$ داشته باشیم [۷]

$$\mathbf{x}^T\mathbf{A}\mathbf{x} \geqslant 0$$

تعریف ۱-۲-۵. ماتریس حقیقی و مربعی $\mathbf{A}$ را نرمال گویند اگر و تنها اگر $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A}$. به عنوان نمونه ماتریس زیر یک ماتریس نرمال است.

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

تعریف ۱-۲-۶. اگر رئوس گراف G را با $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$ نمایش دهیم، ماتریس مجاورت $\mathbf{W}_{n\times n}$ ماتریسی می‌باشد که درایه ij ام آن برابر است با تعداد یال‌هایی که $\mathbf{v}_i$ را به $\mathbf{v}_j$ متصل می‌کند. بدیهی است در گراف‌های بدون جهت

$$\forall i,j : \begin{cases} \mathbf{W}_{ij} = 1 \\ \text{یا} \\ \mathbf{W}_{ij} = 0 \end{cases}$$

و $\mathbf{W}_{ij} = \mathbf{W}_{ji}$، یعنی ماتریس مجاورت یک گراف بدون جهت و متقارن است.

اگر گراف بدون حلقه باشد $\forall i : \mathbf{W}_{ii} = 0$، یعنی درایه‌های روی قطر اصلی ماتریس مجاورت در این حالت همگی صفر می‌باشند. لذا به راحتی دیده می‌شود ماتریس مجاورت یک گراف ساده، یک ماتریس متقارن با درایه‌های صفر و یک می‌باشد که تمام درایه‌های قطر اصلی آن الزاماً صفر است.

تعریف ۱-۲-۷. ماتریس دلخواه $\mathbf{M}_{n\times n}$ را در نظر بگیرید. بردار غیرصفر $\mathbf{v}$ یک بردار ویژه با مقدار ویژه λ برای ماتریس $\mathbf{M}$ خواهد بود اگر رابطه زیربرقرار باشد

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

در مجموع ماتریس $n \times n$ بعدی $\mathbf{M}$ دارای n بردار ویژه و n مقدار ویژه خواهد بود [۷].

تعریف ۱-۲-۸. برای ماتریس $\mathbf{M}_{n\times n}$ با مقادیر ویژه $\lambda_1, \lambda_2, ..., \lambda_n$ اثر[۱] ماتریس به صورت زیر محاسبه می‌شود

$$trace(\mathbf{M}) = \lambda_1 + \lambda_2 + ... + \lambda_n$$

تعریف ۱-۲-۹ (ماتریس تنک[۲]). در آنالیز عددی به ماتریسی گفته می‌شود که اکثر درایه‌های آن صفر هستند. به عنوان نمونه ماتریس زیر یک ماتریس تنک است.

$$\begin{bmatrix} ۱ & ۰ & ۰ & ۰ & ۱ \\ ۰ & ۱ & ۰ & ۰ & ۱ \\ ۰ & ۰ & ۱ & ۰ & ۰ \\ ۰ & ۰ & ۰ & ۱ & ۰ \end{bmatrix}$$

تعریف ۱-۲-۱۰. پنروز[۳] در سال ۱۹۵۵ نشان داد که برای هر ماتریس با درایه‌های مختلط $\mathbf{A}$ با ابعاد $m \times n$ یک ماتریس $\mathbf{B}$ با ابعاد $n \times m$ وجود دارد که در چهار رابطه زیر صدق می‌کند:

- $\mathbf{ABA} = \mathbf{A}$
- $\mathbf{BAB} = \mathbf{B}$
- $\mathbf{AB}$ هرمیتی است.
- $\mathbf{BA}$ هرمیتی است.

چنین ماتریسی $(\mathbf{B})$ ، شبه معکوس مور - پنروز $\mathbf{A}$ نامیده می‌شود و با $\mathbf{A}^{\dagger}$ نمایش داده می‌شود.

تعریف ۱-۲-۱۱ (قطعه به قطعه پیوسته). به تابع $g : \mathbb{R} \to \mathbb{R}$ قطعه به قطعه پیوسته گفته می‌شود اگر در هر فاصله تنها دارای تعداد متناهی انفصال باشد و حد چپ و راست در هر انفصال تعریف شده باشد(لازم نیست برابر باشند).

تعریف ۱-۲-۱۲ (انتگرال لبگ[۴]). انتگرال لبگ تابع F در فضای اندازه X به صورت زیر نوشته می‌شود [۸]

$$\int_X F d\mu$$

[۱] Trace [۲] Sparse [۳] Penrose [۴] Lebesgue integral

که تاکیدی به این موضوع است که انتگرال نسبت به اندازه μ گرفته می‌شود.

اندازه μ تابعی است که بر سیگما جبر $\sum$ روی مجموعه X **تعریف** می‌شود و مقادیری بین $[0, +\infty)$ می‌پذیرد و دارای خصوصیات زیر است:

$$1)\mu(\varnothing) = 0$$

$$2)\mu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \mu(E_i)$$

که $E_1, E_2, \ldots$ غیرتهی و شمارش پذیر هستند و $E_i \cap E_j = \varnothing \quad \forall i \neq j$ در این حالت به $(X, \sum, \mu)$ فضای اندازه و به اعضای $\sum$ مجموعه‌های اندازه پذیر گفته می‌شود.

سیگما جبر. اگر $\sum$ جبر بر روی مجموعه X باشد آن گاه

۱) X عضوی از $\sum$ است.

۲) اگر E عضوی از $\sum$ باشد آن گاه متمم آن نیز عضوی از $\sum$ است.

۳) اجتماع تعداد شمارایی از اعضای $\sum$ مجدداً در $\sum$ است.

تعریف ۱-۲-۱۳ (فضای L^p). فرض کنید X فضای اندازه دلخواه با اندازه مثبت μ باشد. اگر $0 < p < \infty$ و f یک تابع اندازه پذیر مختلط بر X باشد، تعریف می‌کنیم

$$\|f\|_p = \left\{ \int_X |f|^p \, d\mu \right\}^{\frac{1}{p}}$$

و $L^p(\mu)$ از تمام f هایی تشکیل شده باشد که

$$\|f\|_p < \infty$$

$\|f\|_p$ را نرم L^p ی f می‌نامند [۹].

تعریف ۱-۲-۱۴. نزدیکی[1] دو تابع شبکه f_M و تابع هدف f در فضای L^p به صورت زیر تعریف می‌شود [۱۰]

$$\|f_M - f\|_p = \left[\int_X | f_M(x) - f(x) |^p \, d\mu \right]^{\frac{1}{p}}.$$

[1] closeness

تعریف ۱-۲-۱۵ (فضای تولید شده[۱]). فضای برداری V به روی میدان F را در نظر بگیرید. این فضا توسط بردارهای $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$ تولید شده است اگر کلیه این بردارها متعلق به V بوده و بتوان هر بردار $\mathbf{v} \in V$ را به صورت ترکیب خطی از این بردارها نوشت. هم چنین فضای تولید شده توسط بردارهای $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$ را به صورت $sp\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$ نمایش می‌دهند.

تعریف ۱-۲-۱۶ (ترتیب معکوس فرهنگ لغت[۲]). فرض کنید

$$\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D \quad , \quad \mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{iD}]^T \in \mathbb{R}^D (i = 1, 2)$$

باشد آن گاه $\mathbf{x}_1 <_D \mathbf{x}_2$ تعریف می‌شود اگر و تنها اگر $j_0 \in \{1, 2, ..., D\}$ ای وجود داشته باشد به طوری که $x_{1j_0} < x_{2j_0}$ و برای $j = j_0 + 1, ..., D$ شرط $x_{1j} = x_{2j}$ برقرار باشد. j_0 ویژگی تمایز[۳] نامیده می‌شود و با $da(\mathbf{x}_1, \mathbf{x}_2)$ یا $da(1, 2)$ نشان داده می‌شود [۱۱].

تعریف ۱-۲-۱۷ (بستار). بستار مجموعه $\mathbf{A}$ در $\mathbb{R}^n$ که با $\bar{\mathbf{A}}$ نشان داده می‌شود به صورت زیر به دست می‌آید

$$\bar{\mathbf{A}} = \mathbf{A} \cup \partial\mathbf{A}$$

که در آن $\partial\mathbf{A}$ نقاط حدی مجموعه $\mathbf{A}$ است.

$$a \in \partial\mathbf{A} \Leftrightarrow \begin{cases} \forall U_b \cap A \neq \varnothing \\ \forall U_b \cap A^c \neq \varnothing \end{cases}$$

A^c متمم مجموعه $\mathbf{A}$ است و

$$U_{\mathbf{x}} = \{\mathbf{y} \in \mathbb{R}^n \quad | \quad \|\mathbf{x} - \mathbf{y}\| < \delta_u\}$$

که δ_u شعاع همسایگی است [۸].

تعریف ۱-۲-۱۸ (چگال). مجموعه $\mathbf{A}$ در $\mathbb{R}^n$ چگال است اگر و تنها اگر $\bar{\mathbf{A}} = \mathbb{R}^n$, $\forall \mathbf{A} \subseteq \mathbb{R}^n$ [۸].

تعریف ۱-۲-۱۹ (ناحیه). یک مجموعه بسته صرف نظر از این که کران دار است یا نه را ناحیه می‌نامند [۱۲].

---

[۱] span [۲] inverse dictionary order [۳] different

تعریف ۱-۲-۲۰. با فرض m مجموعه k_i ، $i = ۱, ..., m$، مجموعه‌های k_i را مجزا از هم[۱] گویند اگر و تنها اگر بین هر دو مجموعه اشتراکی نباشد [۸] ، یعنی،

$$k_i \cap k_j = \varnothing \qquad \forall i \neq j$$

تعریف ۱-۲-۲۱ (کهاد). اگر $\mathbf{A}$ یک ماتریس مربعی باشد، دترمینان ماتریسی مربعی و کوچکتری که از حذف یک یا چند سطر و ستون $\mathbf{A}$ به دست می‌آید را کهاد ماتریس $\mathbf{A}$ می‌نامند. اگر فقط سطر i ام و ستون j ام از ماتریس $\mathbf{A}$ حذف شود و دترمینان گرفته شود، آن گاه کهاد مرتبه اول i ام و j ام به دست می‌آید. اگر دو سطر و دو ستون حذف گردد و دترمینان گرفته شود یکی از کهادهای مرتبه دوم حاصل خواهد شد. منظور از کهاد معمولاً کهاد مرتبه اول است.

تعریف ۱-۲-۲۲ (رتبه). رتبه ماتریس $\mathbf{A}_{n\times n}$ برابر با ماکزیمم تعداد ستون(سطر)های مستقل خطی در آن ماتریس است، که با نماد $\mathrm{Rank}(\mathbf{A})$ نشان داده می‌شود.

ازآن جایی که رتبه یک ماتریس به صورت بزرگترین درجه کلیه کهادهای غیرصفر آن ماتریس تعریف می‌شود، می‌توان نتیجه گرفت که رتبه یک ماتریس مربعی مانند $\mathbf{A}_{n\times n}$ حداکثر می‌تواند برابر n باشد و این زمانی است که تمامی ستون(سطر)های ماتریس مستقل خطی باشند و در این صورت $|\mathbf{A}| \neq ۰$ یعنی ماتریس $\mathbf{A}_{n\times n}$ نامنفرد است (ماتریس نامنفرد معکوس‌پذیر است). در چنین حالتی ماتریس $\mathbf{A}_{n\times n}$ را رتبه کامل[۲] می‌نامند و اگر $|\mathbf{A}| = ۰$ باشد ماتریس منفرد بوده و تعدادی از ستون‌های آن وابستگی خطی دارند، چنین ماتریسی نقص رتبه[۳] دارد.

برای ماتریس غیرمربعی $\mathbf{A}_{m\times n}$ ، $\mathrm{Rank}(\mathbf{A}) \leqslant \min(m, n)$ است، که در صورت مساوی بودن می‌گوییم ماتریس $\mathbf{A}$ رتبه کامل است و اگر کوچکتر باشد نقص رتبه دارد.

$$\mathrm{Rank}(\mathbf{A}_{m\times n}) = \min(m, n) = \begin{cases} m \rightarrow (\text{رتبه سطری کامل}^{۴}) & \text{if} \quad m \leqslant n \\ n \rightarrow (\text{رتبه ستونی کامل}^{۵}) & \text{if} \quad n \leqslant m \end{cases}$$

تعریف ۱-۲-۲۳ (دنباله کُشی[۶]). دنباله‌ای است که جملات آن با افزایش دنباله به هم نزدیک‌تر و نزدیک‌تر می‌شوند. دنباله $\{x_۱, x_۲, ...\}$ یک دنباله کُشی است اگر رابطه زیر برقرار باشد

$$\forall \varepsilon > ۰ \quad \exists N \in \mathbb{N} \quad \forall n, m \in \mathbb{N} : n, m \geqslant N : \|x_n - x_m\| < \varepsilon.$$

---

[۱] disjoint [۲] Full Rank [۳] Rank Deficiency [۶] Cauchy

اگر دنباله کُشی در فضایی کامل قرار داشته باشد، به عضوی از همان فضا همگرا می‌شود [۸].

تعریف ۱-۲-۲۴ (فضای متریک [۱]). فضای متری یا فضای متریک به مجموعه‌ای گفته می‌شود که مفهومی از نوع فاصله[۲] (موسوم به متری) مابین اعضای آن تعریف شده باشد. زوج مرتب $(\mathbf{X}, d)$ را که در آن $\mathbf{X}$ مجموعه‌ای از نقاط و d یک تابع حقیقی $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ می‌باشد یک فضای متریک گویند هرگاه ۴ شرط زیر را دارا باشد [۸]:

$$d(\mathbf{p}, \mathbf{q}) \geqslant ۰$$

$$d(\mathbf{p}, \mathbf{q}) = ۰ \iff \mathbf{p} = \mathbf{q}$$

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) \qquad \text{(خاصیت تقارنی)}$$

$$d(\mathbf{p}, \mathbf{q}) + d(\mathbf{q}, \mathbf{r}) \geqslant d(\mathbf{p}, \mathbf{r}) \qquad \text{(نامساوی مثلث)}$$

این خاصیت‌ها به طور شهودی مفهوم فاصله را بیان می‌کند. مثلاً فاصله بین دو نقطه همیشه مقداری مثبت است و یا فاصله بین دو نقطه $\mathbf{p}$ و $\mathbf{q}$ برابر با فاصله $\mathbf{q}$ تا $\mathbf{p}$ است. هم چنین براساس نامساوی مثلث، مسیر مستقیم $\mathbf{p}$ تا $\mathbf{q}$ کوتاهتر از مسیری است که از $\mathbf{p}$ به $\mathbf{r}$ و سپس از $\mathbf{r}$ به $\mathbf{q}$ طی می‌کنیم.

تعریف ۱-۲-۲۵ (فضای متریک کامل[۳]). در آنالیز ریاضی، یک فضای متریک $\mathbf{M}$ ، کامل یا (فضای کُشی) گفته می‌شود، اگر هر دنباله کُشی از نقاط موجود در $\mathbf{M}$ ، به عددی در $\mathbf{M}$ همگرا شود. مثلاً فضای اعداد گویا کامل نیست، زیرا اعداد گنگ مانند $\sqrt{۲}$ در آن یافت نمی‌شوند. اما اعداد حقیقی مجموعه‌ای کامل است [۸].

تعریف ۱-۲-۲۶ (خارج قسمت رایلی[۴]). دو ماتریس متقارن $\mathbf{A}$ و $\mathbf{B}$ به صورت تابعی از بردار $\mathbf{w}$ به صورت زیر تعریف می‌شود

$$R(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{A} \mathbf{w}}{\mathbf{w}^T \mathbf{B} \mathbf{w}}$$

تعریف ۱-۲-۲۷ (ماتریس لاپلاسین). فرض کنید $G = (V, E)$ یک گراف ساده باشد که N رأس دارد ، $\mathbf{W}$

[۱] Metric Space [۲] distance [۳] Complete Metric Space [۴] Rayleigh quotient

ماتریس مجاورت $N \times N$ با عناصر روی قطر اصلی صفر، مطابق رابطه زیر است [۱۳]

$$\mathbf{W}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}) & i \neq j \\ 0 & i = j \end{cases} \quad (۹-۱)$$

که در آن $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ بوده و مقادیر متناظر با رئوس v_i و v_j می‌باشند و پارامتر σ عرض همسایگی را کنترل می‌کند . $\mathbf{D}$ ماتریس درجه گراف $N \times N$ قطری هم چون زیر است

$$\mathbf{D}(i,i) = \sum_j w_{ij}$$

ماتریس لاپلاسین $\mathbf{L}(G)$ متناظر با گراف G یک ماتریس $N \times N$ متقارن است و به صورت زیر تعریف می‌شود

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

ماتریس لاپلاسین $\mathbf{L} = \mathbf{D} - \mathbf{W}$ غیرنرمال است.

۱-۲-۲ لم‌ها

در این بخش چند لم مورد نیاز برای ادامه کار بیان خواهد شد.

لم ۱-۲-۲۸. فضای L^2 فضایی کامل است[۱۰].

لم ۱-۲-۲۹. در فضای L^p به ازای هر $p \in [1, +\infty)$ ، اگر و تنها اگر تابع g یک چندجمله‌ای نباشد، آن گاه

$$sp\{g(\mathbf{w}^T\mathbf{x} + b) : (\mathbf{w}, b) \in \mathbb{R}^D \times \mathbb{R}\}$$

چگال است[۱۰].

لم ۱-۲-۳۰. فرض کنید تابع غیرخطی قطعه به قطعه پیوسته $g : \mathbb{R} \to \mathbb{R}$ ، هر مقدار مثبت $\hat{\theta} < \frac{\pi}{2}$ و g_0 ی را داریم، به ازای هر دنباله تابع به طور تصادفی تولید شده $\{g_M\}$ ، یک عدد صحیح مثبت m وجود دارد به طوری که برای هر قطعه پیوسته $G_{(M,m)} = \{g_M, g_{M+1}, ..., g_{M+m-1}\}(M = 1, 2, ...)$ با احتمالی نزدیک

به یک، آن گاه $g_i \in G_{(M,m)}$ وجود دارد که در رابطه $\theta_{(g_i,g_*)} < \hat{\theta}$ صدق کند ($\theta_{(g_i,g_*)}$ نشان‌دهنده زاویه بین بردارهای g_i و g_* است). اثبات این قضیه در مرجع [۱۴] است.

لم ۱-۲-۳۱. معکوس ماتریس بلوکی . فرض کنید $\mathbf{U}$ ماتریسی با ابعاد $(m+n)\times(m+n)$ است و به یک فرم بلوکی افراز می‌شود[۱۵]

$$\mathbf{U} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

به طوری که $\mathbf{A}$ ماتریس $m\times m$، $\mathbf{B}$ ماتریس $m\times n$، $\mathbf{C}$ ماتریس $n\times m$ و $\mathbf{D}$ ماتریس $n\times n$ و معکوس پذیر هستند. آن گاه

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \\ \mathbf{Q}_3 & \mathbf{Q}_4 \end{bmatrix}$$

$$\mathbf{Q}_1 = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$$

$$\mathbf{Q}_2 = -(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}$$

$$\mathbf{Q}_3 = -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$$

$$\mathbf{Q}_4 = \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}$$

با توجه به قضیه ۱-۲-۳۶ لم زیر به دست می‌آید

لم ۱-۲-۳۲. با فرض m ناحیه مجزای $k_1, k_2, ..., k_m$ در $\mathbb{R}^d$، m مقدار حقیقی متناظر دلخواه $c_1, c_2, ..., c_m$ و ناحیه دلخواه X مجزا از هر k_i، یک تابع پیوسته $f(\mathbf{x})$ وجود دارد به طوری که رابطه زیر برقرار باشد

$$f(\mathbf{x}) = \begin{cases} c_i & \textit{if} \quad \mathbf{x} \in k_i, i = 1, ..., m \\ c_* & \textit{if} \quad \mathbf{x} \in X \end{cases}$$

که در آن c_* یک مقدار حقیقی دلخواه متفاوت از $c_1, c_2, ..., c_m$ است[۱۲].

لم ۱-۲-۳۳. با فرض هر تابع کران دار $g(x)$ در $\mathbb{R}$ که دارای حدهای $\lim_{\mathbf{x}\to-\infty} g(x)$ و $\lim_{\mathbf{x}\to+\infty} g(x)$ است و $\lim_{\mathbf{x}\to-\infty} g(x) \neq \lim_{\mathbf{x}\to+\infty} g(x)$. آن گاه تمام ترکیب‌های خطی $\sum_{i=1}^{M} \beta_i g(\mathbf{w}_i^T\ \mathbf{x} + b_i)$ در

$C(\mathbb{R}^d)$ یا $C(\Omega)$ چگال است، که Ω یک مجموعه فشرده از $\mathbb{R}^d$، $\beta_i \in \mathbb{R}$ ، $\mathbf{w}_i \in \mathbb{R}^D$ و $\mathbf{w}_i^T \mathbf{x}$ ضرب داخلی $\mathbf{w}_i$ و $\mathbf{x}$ است [۱۲].

لم ۱-۲-۳۴. اگر تابع $g(x)$ غیرخطی، کران دار و پیوسته باشد آن گاه در هر مجموعه فشرده از $\mathbb{R}^d$ تمام ترکیب‌های خطی $\sum_{i=1}^{M} \beta_i g(\mathbf{w}_i^T \mathbf{x} + b_i)$ چگال است [۱۲].

لم ۱-۲-۳۵. فرض کنید چندجمله‌ای $P \in \mathcal{P}_s^d$ با d متغیر و حداکثر درجه s است. بنابراین مجموعه صفر آن به صورت زیر تعریف می‌شود

$$N(P) := \{u \in [0,1]^d : P(u) = 0\}$$

که دارای اندازه لبگ صفر است. برای مشاهده اثبات به مرجع [۱۶] رجوع شود.

۱-۲-۳ قضایا

در این بخش قضایای مربوط به قابلیت طبقه‌بندی در فصل اول و پس از تعریف ماتریس لاپلاسین در بخش تعاریف ویژگی‌های مورد نیاز آن در پایان‌نامه در قالب قضیه، قضیه ریلی ریتز و مسئله مقدار ویژه تعمیم یافته که در فصل سوم به کار برده می‌شوند بیان خواهند شد.

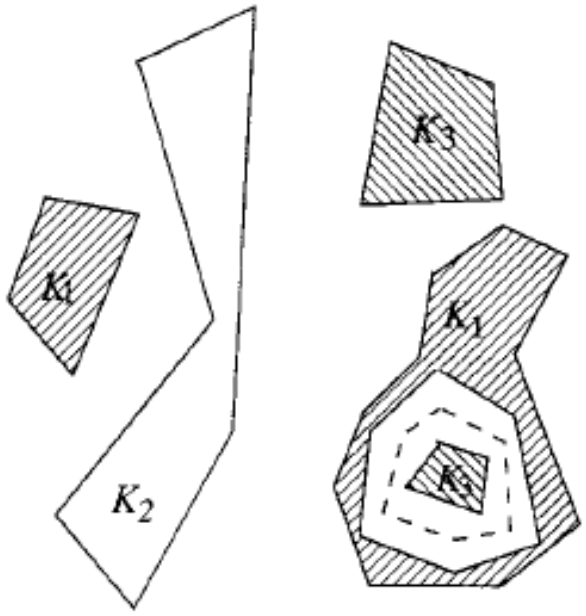
قضیه ۱-۲-۳۶ (تعمیم تیتز[1]). فرض کنید X یک فضای توپولوژی نرمال، A یک زیرمجموعه بسته از X و f یک تابع حقیقی پیوسته روی A باشند. آنگاه یک تابع حقیقی پیوسته g روی X وجود دارد به طوری که رابطه زیر برقرار باشد [۱۲]

$$\forall x \in A \qquad g(x) = f(x)$$

قضیه ۱-۲-۳۷. با فرض هر تابع $g : \mathbb{R} \to \mathbb{R}$، اگر تمام ترکیب‌های خطی $\sum_{i=1}^{M} \beta_i g(\mathbf{w}_i^T \mathbf{x} + b_i)$ در $C(\mathbb{R}^d)$ یا $C(\Omega)$ چگال باشد، آن گاه یک *SLFN* با تابع فعالیتی هم چون $g(x)$ و تعداد نورون‌های پنهان کافی می‌تواند نواحی مجزای دلخواه با هر شکلی را تفکیک کند [۱۲] (شکل ۱-۵ را ببینید).

قضیه ۱-۲-۳۸. با فرض هر تابع کران دار $g : \mathbb{R} \to \mathbb{R}$ که دارای حدهای $\lim_{\mathbf{x}\to-\infty} g(x)$ و $\lim_{\mathbf{x}\to+\infty} g(x)$ است و $\lim_{\mathbf{x}\to-\infty} g(x) \neq \lim_{\mathbf{x}\to+\infty} g(x)$. آن گاه یک *SLFN* با تابع فعالیتی هم چون $g(x)$ و تعداد نورون‌های پنهان کافی می‌تواند نواحی مجزای دلخواه با هر شکلی را تفکیک کند [۱۲].

[1] Tietze's Extension



شکل ۱-۵: نواحی تصمیم گیری با اشکال دلخواه

قضیه ۱-۲-۳۹ (ریلی ریتز[۱]). فرض کنید $\mathbf{A}$ یک ماتریس متقارن حقیقی و $\mathbf{x}$ یک بردار غیرصفر باشد که بر $j-1$ تا از کوچکترین بردارهای ویژه $\mathbf{A}$، یعنی $\mathbf{v}_1, ..., \mathbf{v}_{j-1}$ عمود است در این صورت کسر $\frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$ با استفاده از j امین کوچکترین بردار ویژه $\mathbf{A}$، یعنی $\mathbf{v}_j$ مینیمم می‌شود و مقدار این مینیمم متناظر با مقدار ویژه متناظر با $\mathbf{v}_j$ یعنی λ_j است *[۱۳]*.

برهان. اگر $\{\mathbf{v}_1, ..., \mathbf{v}_{j-1}, ..., \mathbf{v}_n\}$ بردارهای ویژه ماتریس $\mathbf{A}$ متناظر با مقادیر ویژه $\lambda_1 \leqslant \lambda_2 \leqslant ... \leqslant \lambda_n$ باشند، رابطه زیر برقرار است

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i. \qquad (۱-۱۰)$$

طبق فرض قضیه داریم

$$\mathbf{x} \perp \mathbf{v}_i \quad (< \mathbf{x}\,, \mathbf{v}_i >= 0) \quad \forall i = 1, ..., j-1$$

چون $\{\mathbf{v}_1, ..., \mathbf{v}_{j-1}, ..., \mathbf{v}_n\}$ بردارهای ویژه ماتریس $\mathbf{A}$ ، مستقل خطی می‌باشند بنابراین تشکیل یک پایه برای فضا می‌دهند پس $\mathbf{x}$ را می‌توان به صورت ترکیب خطی این بردارها نوشت، در نتیجه اسکالرهای $c_1, ..., c_n$ موجودند طوری که

[۱] Rayleigh-Ritz

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{v}_i \qquad (1\text{-}11)$$

لذا با توجه به روابط (۱-۱۰) و (۱-۱۱) می‌توان نوشت

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{A}(\sum_{i=1}^{n} c_i\mathbf{v}_i) = \mathbf{x}^T(\sum_{i=1}^{n} c_i\mathbf{A}\mathbf{v}_i) = \mathbf{x}^T(\sum_{i=1}^{n} c_i\lambda_i\mathbf{v}_i)$$
$$= \sum_{i=1}^{n} c_i\lambda_i\mathbf{x}^T\mathbf{v}_i = \sum_{i=j}^{n} c_i\lambda_i\mathbf{x}^T\mathbf{v}_i$$
$$= \sum_{i=j}^{n} c_i\lambda_i c_i\mathbf{v}_i^T\mathbf{v}_i = \sum_{i=j}^{n} c_i^2\lambda_i \left\|\mathbf{v}_i\right\|^2$$

هم چنین بنا به رابطه (۱-۱۱) داریم

$$\mathbf{x}^T\mathbf{x} = (\sum_{i=1}^{n} c_i\mathbf{v}_i^T)(\sum_{i=1}^{n} c_i\mathbf{v}_i) = \sum_{i=1}^{n} c_i^2\mathbf{v}_i^T\mathbf{v}_i = \sum_{i=1}^{n} c_i^2 \left\|\mathbf{v}_i\right\|^2$$

بنابراین می‌توان نوشت

$$\frac{\sum_{i=j}^{n} c_i^2\lambda_i\|\mathbf{v}_i\|^2}{\sum_{i=1}^{n} c_i^2\|\mathbf{v}_i\|^2} = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

چون

$$\lambda_j \leqslant \lambda_i \quad i = j, ..., n$$

نتیجه می‌گیریم

$$\frac{(\sum_{i=j}^{n} c_i^2\|\mathbf{v}_i\|^2)\lambda_j}{\sum_{i=1}^{n} c_i^2\|\mathbf{v}_i\|^2} \leqslant \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

با فرض رابطه زیر

$$\mathbf{x} = \mathbf{v}_j = \sum_{i=1}^{n} c_i\mathbf{v}_i \quad \exists\, c_j = 1, c_i = 0 \quad \forall i \neq j$$

داریم

$$\frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \frac{c_j^2\|\mathbf{v}_j\|^2\lambda_j}{c_j^2\|\mathbf{v}_j\|^2} = \lambda_j$$

بنابراین کسر $\frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$ در $\mathbf{x} = \mathbf{v}_j$ مینیمم می‌شود. □

مسئله مقدار ویژه تعمیم یافته

هدف مسئله مقدار ویژه تعمیم یافته[1] با دو ماتریس متقارن $\mathbf{A}$ و $\mathbf{B}$ ، یافتن اسکالر λ و بردار متناظر $\mathbf{\Phi}$ برای رابطه

$$\mathbf{A}\mathbf{\Phi}_i = \lambda_i\mathbf{B}\mathbf{\Phi}_i \quad (i = 1, ..., n)$$

[1] Generalized eigenvalue problem

یا ماتریس‌های مقدار ویژه $\mathbf{\Lambda}$ و بردار ویژه $\mathbf{\Phi}$ در فرم ماتریسی زیر است

$$\mathrm{A}\mathbf{\Phi} = \mathrm{B}\mathbf{\Phi}\mathbf{\Lambda}.$$

برای یافتن ماتریس قطری مقدار ویژه $\mathbf{\Lambda_B}$ و ماتریس متعامد بردار ویژه ($\boldsymbol{\phi}_\mathrm{B}^T = \boldsymbol{\phi}_\mathrm{B}^{-1}$)، مسئله مقدار ویژه $\mathbf{B}$ یعنی $\mathrm{B}\boldsymbol{\phi}_\mathrm{B} = \boldsymbol{\phi}_\mathrm{B}\mathbf{\Lambda}_\mathrm{B}$ حل می‌شود. اگر مسئله مقدار ویژه $\mathbf{B}$ در عبارت $\boldsymbol{\phi}_\mathrm{B}^{-1}$ ضرب شود به رابطه زیر خواهیم رسید

$$\boldsymbol{\phi}_\mathrm{B}^{-1}\mathrm{B}\boldsymbol{\phi}_\mathrm{B} = \boldsymbol{\phi}_\mathrm{B}^T\mathrm{B}\boldsymbol{\phi}_\mathrm{B} = \mathbf{\Lambda}_\mathrm{B} \qquad (۱-۱۲)$$

باشد. هر دو طرف رابطه (۱-۱۲) از سمت چپ و راست در عبارت $\mathbf{\Lambda}^{-1/2}$ ضرب می‌شوند

$$\begin{aligned}\mathbf{\Lambda}_\mathrm{B}^{-1/2}\left(\boldsymbol{\phi}_\mathrm{B}^T\mathrm{B}\boldsymbol{\phi}_\mathrm{B}\right)\mathbf{\Lambda}_\mathrm{B}^{-1/2} &= \mathbf{\Lambda}_\mathrm{B}^{-1/2}\mathbf{\Lambda}_\mathrm{B}\mathbf{\Lambda}_\mathrm{B}^{-1/2} \\ &= \mathbf{I} \qquad (۱-۱۳)\end{aligned}$$

و $\boldsymbol{\phi}'_\mathrm{B} = \boldsymbol{\phi}_\mathrm{B}\mathbf{\Lambda}_\mathrm{B}^{-1/2}$ تعریف می‌شود. پس رابطه (۱-۱۳) به صورت زیر نوشته می‌شود

$$(\boldsymbol{\phi}'_\mathrm{B})^T\mathrm{B}\boldsymbol{\phi}'_\mathrm{B} = \mathbf{I}$$

توجه داشته باشید $\boldsymbol{\phi}'_\mathrm{B}$ همان طور که در زیر اثبات شده متعامد نیست

$(\boldsymbol{\phi}'_\mathrm{B})^{-1} \neq (\boldsymbol{\phi}'_\mathrm{B})^T$

$(\boldsymbol{\phi}_\mathrm{B}\mathbf{\Lambda}_\mathrm{B}^{-1/2})^{-1} \neq (\boldsymbol{\phi}_\mathrm{B}\mathbf{\Lambda}_\mathrm{B}^{-1/2})^T$

$\mathbf{\Lambda}_\mathrm{B}^{1/2}\boldsymbol{\phi}_\mathrm{B}^{-1} \neq \mathbf{\Lambda}_\mathrm{B}^{-1/2}\boldsymbol{\phi}_\mathrm{B}^T, \quad \boldsymbol{\phi}_\mathrm{B}^{-1} = \boldsymbol{\phi}_\mathrm{B}^T$

$\mathbf{\Lambda}_\mathrm{B}^{1/2}\boldsymbol{\phi}_\mathrm{B}^T \neq \mathbf{\Lambda}_\mathrm{B}^{-1/2}\boldsymbol{\phi}_\mathrm{B}^T$

همان تبدیل برای $\mathbf{A}$ به کار برده می‌شود

$$(\boldsymbol{\phi}'_\mathrm{B})^T\mathrm{A}\boldsymbol{\phi}'_\mathrm{B} = (\mathbf{\Lambda}_\mathrm{B}^{-1/2}\boldsymbol{\phi}_\mathrm{B}^T)\mathrm{A}(\boldsymbol{\phi}_\mathrm{B}\mathbf{\Lambda}_\mathrm{B}^{-1/2}) = \mathrm{A}' \qquad (۱-۱۴)$$

لازم به ذکر است که $\mathbf{A}'$ هم چون $\mathbf{A}$ متقارن است

$(\mathbf{A}')^T = \mathbf{A}'$

$((\boldsymbol{\phi}'_{\mathbf{B}})^T \mathbf{A} \boldsymbol{\phi}'_{\mathbf{B}})^T = (\boldsymbol{\phi}'_{\mathbf{B}})^T \mathbf{A} \boldsymbol{\phi}'_{\mathbf{B}}$

$(\boldsymbol{\phi}'_{\mathbf{B}})^T \mathbf{A} \boldsymbol{\phi}'_{\mathbf{B}} = (\boldsymbol{\phi}'_{\mathbf{B}})^T \mathbf{A} \boldsymbol{\phi}'_{\mathbf{B}}$

از آن جایی که $\mathbf{A}'$ متقارن است، می‌تواند با ماتریس بردار ویژه متعامدش ($\boldsymbol{\phi}_{\mathbf{A}}$) قطری شود لذا خواهیم داشت

$$\boldsymbol{\phi}_{\mathbf{A}}^T \mathbf{A}' \boldsymbol{\phi}_{\mathbf{A}} = \boldsymbol{\Lambda}$$

یعنی با جایگذاری رابطه (۱-۱۴) در فرمول بالا خواهیم داشت

$$\boldsymbol{\phi}_{\mathbf{A}}^T (\boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{B}}^T \mathbf{A} \boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2}) \boldsymbol{\phi}_{\mathbf{A}} = (\boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{B}}^T) \mathbf{A} (\boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}}) = \boxed{\boldsymbol{\phi}^T \mathbf{A} \boldsymbol{\phi} = \boldsymbol{\Lambda}}$$

که $\boldsymbol{\phi} = \boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}}$ تعریف می‌شود و همان طور که در زیر می‌بینیم متعامد نیست

$\boldsymbol{\phi}^{-1} \neq \boldsymbol{\phi}^T$

$(\boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}})^{-1} \neq (\boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}})^T$

$\boldsymbol{\phi}_{\mathbf{A}}^{-1} \boldsymbol{\Lambda}_{\mathbf{B}}^{1/2} \boldsymbol{\phi}_{\mathbf{B}}^{-1} \neq \boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{B}}^T, \quad \boldsymbol{\phi}_{\mathbf{A}}^{-1} = \boldsymbol{\phi}_{\mathbf{A}}^T, \quad \boldsymbol{\phi}_{\mathbf{B}}^{-1} = \boldsymbol{\phi}_{\mathbf{B}}^T$

$\boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\Lambda}_{\mathbf{B}}^{1/2} \boldsymbol{\phi}_{\mathbf{B}}^T \neq \boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{B}}^T$

$\boldsymbol{\phi}$ هم چنین $\mathbf{B}$ را قطری می‌کند

$$\boldsymbol{\phi}^T \mathbf{B} \boldsymbol{\phi} = (\boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}})^T \mathbf{B} (\boldsymbol{\phi}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}}) = \boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} (\boldsymbol{\phi}_{\mathbf{B}}^T \mathbf{B} \boldsymbol{\phi}_{\mathbf{B}}) \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}}$$

باجایگذاری رابطه (۱-۱۲)، رابطه زیر به دست می‌آید

$$\boldsymbol{\phi}^T \mathbf{B} \boldsymbol{\phi} = \boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\Lambda}_{\mathbf{B}} \boldsymbol{\Lambda}_{\mathbf{B}}^{-1/2} \boldsymbol{\phi}_{\mathbf{A}} = \boldsymbol{\phi}_{\mathbf{A}}^T \boldsymbol{\phi}_{\mathbf{A}}$$

$$\boxed{\boldsymbol{\phi}^T \mathbf{B} \boldsymbol{\phi} = \mathbf{I}}$$

پس از طی مراحل بالا به دو رابطه زیر رسیدیم

$$\begin{cases} \boldsymbol{\phi}^T \mathbf{A} \boldsymbol{\phi} = \boldsymbol{\Lambda} \\ \boldsymbol{\phi}^T \mathbf{B} \boldsymbol{\phi} = \mathbf{I} \end{cases}$$

هر دو طرف رابطه دوم از سمت راست در $\boldsymbol{\Lambda}$ ضرب می‌شود و سمت چپ دو رابطه برابر هم قرار داده می‌شود و رابطه زیر نتیجه می‌شود

$$\mathbf{A} \boldsymbol{\phi} = \mathbf{B} \boldsymbol{\phi} \boldsymbol{\Lambda}$$

یعنی، $\mathbf{\Lambda}$ و $\boldsymbol{\phi}$ ماتریس‌های مقادیر ویژه و بردار ویژه مسئله مقدار ویژه تعمیم یافته‌اند. همان طور که در بالا نشان داده شد $\boldsymbol{\phi}$ متعامد نیست.

مطابق با تعریف ۱-۲-۲۶ برای یافتن $\mathbf{w}$ بهینه متناظر با اکسترمم(ماکسیمم یا مینیمم) $R(\mathbf{w})$ ،نسبت به $\mathbf{w}$ مشتق گرفته می‌شود که در زیر آمده است

$$\frac{d}{d\mathbf{w}}R(\mathbf{w}) = \frac{۲\mathbf{A}\mathbf{w}(\mathbf{w}^T\mathbf{B}\mathbf{w}) - ۲\mathbf{B}\mathbf{w}(\mathbf{w}^T\mathbf{A}\mathbf{w})}{(\mathbf{w}^T\mathbf{B}\mathbf{w})^۲}$$

با قرار دادن مشتق برابر صفر رابطه زیر حاصل می‌شود

$$\mathbf{A}\mathbf{w}(\mathbf{w}^T\mathbf{B}\mathbf{w}) = \mathbf{B}\mathbf{w}(\mathbf{w}^T\mathbf{A}\mathbf{w})$$

و با ساده سازی رابطه فوق به جواب نهایی خواهیم رسید

$\mathbf{A}\mathbf{w} = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}\mathbf{B}\mathbf{w}$

$\mathbf{A}\mathbf{w} = R(\mathbf{w})\mathbf{B}\mathbf{w}$

$$\boxed{\mathbf{A}\mathbf{w} = \lambda\mathbf{B}\mathbf{w}}$$

معادله فوق به عنوان مسئله مقدار ویژه تعمیم یافته با مقدار ویژه $\lambda = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}$ و بردار ویژه متناظر $\mathbf{w}$ است. با حل آن بردار $\mathbf{w} = \boldsymbol{\phi}$ متناظر با مقدار ویژه ماکسیمم/مینیمم $\lambda = R(\mathbf{w})$ است، که خارج قسمت رایلی را ماکسیمم/مینیمم می‌کند.

قضیه ۱-۲-۴۰. ویژگی‌های ماتریس لاپلاسین عبارتند از *[۱۷]*:

۱ . به ازای هر بردار $\mathbf{y} \in \mathbb{R}^N$، داریم

$$\mathbf{y}^T\mathbf{L}\mathbf{y} = \frac{۱}{۲}\sum_{i=۱}^{N}\sum_{j=۱}^{N} w_{ij}(y_i - y_j)^۲$$

۲ . ماتریس $\mathbf{L}$ متقارن و نیمه معین مثبت است.

۳ . کوچکترین مقدار ویژه ماتریس $\mathbf{L}$ برابر است با صفر و متناظر است با بردار ویژه $\vec{j}$.

برهان. ۱ . با توجه به تعریف $\mathbf{L}$ داریم

$\mathbf{y}^T\mathbf{L}\mathbf{y} = \mathbf{y}^T\mathbf{D}\mathbf{y} - \mathbf{y}^T\mathbf{W}\mathbf{y}$

$= \sum_{i=۱}^{N} \mathbf{d}_i y_i^۲ - \sum_{i=۱}^{N}\sum_{j=۱}^{N} y_i y_j w_{ij}$

$$= \frac{۱}{۲}(\sum_{i=۱}^{N} \mathbf{d}_i y_i^۲ - ۲\sum_{i=۱}^{N}\sum_{j=۱}^{N} y_i y_j w_{ij} + \sum_{j=۱}^{N} \mathbf{d}_j y_j^۲)$$
$$= \frac{۱}{۲}(\sum_{i=۱}^{N}\sum_{j=۱}^{N} w_{i,j} y_i^۲ - ۲\sum_{i=۱}^{N}\sum_{j=۱}^{N} y_i y_j w_{ij} + \sum_{i=۱}^{N}\sum_{j=۱}^{N} w_{i,j} y_j^۲)$$
$$= \frac{۱}{۲}\sum_{i=۱}^{N}\sum_{j=۱}^{N} w_{i,j}(y_i - y_j)^۲$$

۲ . متقارن بودن ماتریس $\mathbf{L}$ ، مستقیما از متقارن بودن ماتریس‌های $\mathbf{W}$ و $\mathbf{D}$ نتیجه می‌شود. برای نیمه معین مثبت بودن یک ماتریس باید به ازای هر بردار دلخواه $\mathbf{y} \in \mathbb{R}^N$ شرط $\mathbf{y}^T\mathbf{L}\mathbf{y} \geqslant ۰$ برقرار باشد. با توجه به اثبات قسمت (۱) و نامنفی بودن $\mathbf{W}_{i,j}$ ها، به وضوح این شرط برای ماتریس $\mathbf{L}$ برقرار است.

۳ . با توجه به نیمه معین مثبت بودن ماتریس $\mathbf{L}$ واضح است که $\forall \mathbf{y} \in \mathbb{R}^N : \mathbf{y}^T\mathbf{L}\mathbf{y} \geqslant ۰$

اگر فرض کنیم $\mathbf{x}$ یک بردار ویژه با مقدار ویژه λ برای ماتریس $\mathbf{L}$ باشد، داریم

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{x} \Longrightarrow \mathbf{x}^T\mathbf{L}\mathbf{x} = \lambda\mathbf{x}^T\mathbf{x}$$

بنابراین با توجه به این که $\mathbf{x}^T\mathbf{L}\mathbf{x} \geqslant ۰$ و $\mathbf{x}^T\mathbf{x} > ۰$ نتیجه می‌گیریم که $\lambda \geqslant ۰$ و $\lambda = ۰$ یک مقدار ویژه متناظر با بردار ویژه $\vec{j} = [۱, ۱, ..., ۱]^T$ است زیرا

$$\mathbf{L}\vec{j} = (\mathbf{D} - \mathbf{W})\vec{j} = \mathbf{D}\vec{j} - \mathbf{W}\vec{j} = \mathbf{D} - \mathbf{D} = ۰$$

بنابراین کوچکترین مقدار ویژه ماتریس لاپلاسین، همیشه صفر است و بردار ویژه متناظر با آن، بردار $\vec{j}$ است.

□

روش حداقل مربعات

با توجه به بردار ورودی $\mathbf{x}^T = [x_۱, x_۲, ..., x_D]$ هدف پیش بینی خروجی Y با استفاده از مدل خطی هم چون زیر است

$$\hat{Y} = \hat{\beta}_۰ + \sum_{j=۱}^{D} x_j\hat{\beta}_j$$

عبارت $\hat{\beta}_0$ عرض از مبدا است که در یادگیری ماشین بایاس[1] نامیده می‌شود. اغلب مناسب است تا متغیر ثابت ۱ در $\mathbf{x}$، $\hat{\beta}_0$ در بردار ضرایب $\hat{\boldsymbol{\beta}}$ قرار گیرند، $\mathbf{x} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$ و $\hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\boldsymbol{\beta}} \end{bmatrix}$ و بنابراین مدل خطی در فرم برداری به صورت ضرب داخلی مطابق رابطه زیر نوشته می‌شود

$$\hat{Y} = \mathbf{x}^T \hat{\boldsymbol{\beta}}$$

که در آن $\mathbf{x}^T$ نشان‌دهنده ترانهاده بردار است . این جا فقط یک خروجی مدلسازی می‌شود، بنابراین $\hat{Y}$ یک اسکالر است؛ به طور کلی $\hat{Y}$ می‌تواند برداری K تایی باشد که در آن $\boldsymbol{\beta}$ ماتریس ضرایب با ابعاد $D \times K$ است. روش‌های بسیاری برای برازش مدل خطی با یک مجموعه داده آموزشی وجود دارد. یکی از محبوب‌ترین روش‌ها، روش حداقل مربعات است [۱۸]. در این روش ضرایب $\boldsymbol{\beta}$ برای مینیمم‌سازی مجموع مربعات خطا[2] (RSS) انتخاب می‌شوند

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2$$

$RSS(\boldsymbol{\beta})$ تابعی درجه دوم از پارامترهاست و از این رو مینیمم آن وجود دارد، اما ممکن است منحصر به فرد نباشد. رابطه بالا را می‌توان به فرم ماتریسی نوشت

$$RSS(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

که $\mathbf{X}$ ماتریسی با ابعاد $N \times D$ و $\mathbf{y}$ یک بردار خروجی N تایی در مجموعه آموزشی است. از عبارت بالا نسبت به $\boldsymbol{\beta}$ مشتق گرفته و مساوی صفر قرار داده می‌شود

$$-2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = 0$$

اگر $\mathbf{X}^T\mathbf{X}$ نامنفرد باشد آن گاه جواب منحصر به فرد به دست می‌آید

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

---

[1] bias [2] Residual Sum of Squares(RSS)

مقدار برازش شده ورودی i ام $(\mathbf{x}_i)$ برابر است با :

$$\hat{y}_i = \hat{y}(\mathbf{x}_i) = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}$$

۳-۱ ماشین یادگیر نهایی (ELM)

ماشین یادگیر نهایی برای آموزش شبکه‌های عصبی پیشخور تک لایه پنهان (SLFNs) پیشنهاد شده است و سپس برای شبکه‌های عصبی پیشخور تک لایه پنهان تعمیم یافته توسعه یافته است. در ELM، پارامترهای نورون‌های پنهان بدون تنظیم به صورت تکراری، به طور تصادفی مقداردهی اولیه می‌شوند. تنها پارامتری که باید یاد گرفته شود اتصالات(وزن‌های) بین لایه پنهان و لایه خروجی است. در مقایسه با روش‌های یادگیری سنتی شبکه‌های عصبی پیشخور از جمله پس انتشار، ELM کارایی قابل ملاحظه و تمایل رسیدن به بهینه سراسری را دارد، علاوه برابن ELM فقط وزن‌های خروجی بین لایه پنهان و لایه خروجی را با روش حداقل مربعات منظم به روز رسانی می‌کند و پارامترهای لایه پنهان، یعنی وزن‌های ورودی (وزن‌های بین لایه ورودی و لایه پنهان) و بایاس‌ها، به جای این که هم چون SLFNs معمولی به صورت تکراری یاد گرفته شوند، همزمان به طور تصادفی تولید می‌شوند و در طول فاز آموزش ثابت می‌مانند [۱۹].

با توجه به مجموعه‌ای از N نمونه مجزای دلخواه بابرچسب $\{(\mathbf{x}_i, \mathbf{t}_i) | \, i = ۱, ۲, ..., N\}$ که در آن

$$\mathbf{x}_i^T = [x_{i۱}, x_{i۲}, ..., x_{iD}] \in \mathbb{R}^D$$

و بردار هدف

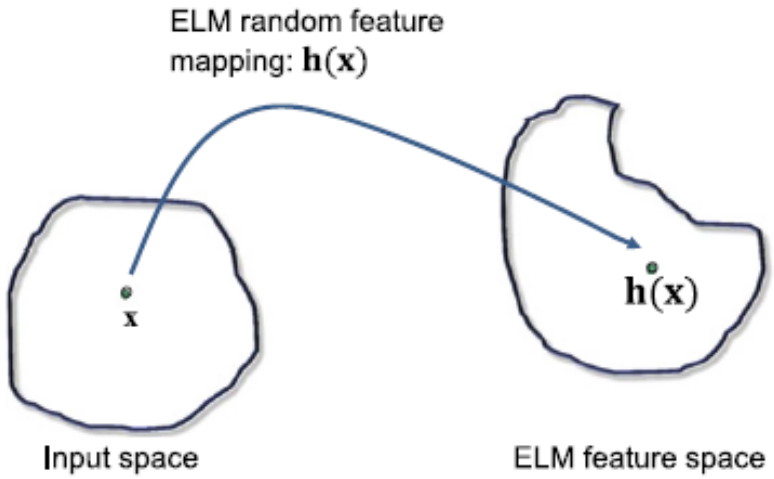
$$\mathbf{t}_i^T = [t_{i۱}, t_{i۲}, ..., t_{iK}] \in \mathbb{R}^K$$

، تابع خروجی ELM برای SLFN تعمیم یافته هم چون زیر فرموله‌بندی می‌شود

$$f_M(\mathbf{x}) = \sum_{i=۱}^{M} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\, \boldsymbol{\beta}$$

که در آن

$$\boldsymbol{\beta} = [\beta_۱, \beta_۲, ..., \beta_M]^T$$



شکل ۱-۶: نگاشت ویژگی تصادفی ELM

بردار وزن خروجی بین M نورون لایه پنهان و $K \geqslant 1$ نورون خروجی است

$$\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), ..., h_M(\mathbf{x})]$$

، نگاشت ویژگی غیرخطی ELM ، بردار خروجی لایه پنهان با توجه به نمونه آموزشی $\mathbf{x}$ است. مطابق با شکل ۱-۶، $\mathbf{h}(\mathbf{x})$ در واقع داده را از فضای ورودی D بعدی به فضای ویژگی M بعدی لایه پنهان $(\mathbf{H})$ نگاشت می‌کند، بنابراین $\mathbf{h}(\mathbf{x})$ یک نگاشت ویژگی است. $h_i(\mathbf{x})$ خروجی نورون پنهان i ام است. توابع خروجی نورون‌های پنهان ممکن است منحصر به فرد نباشند یعنی توابع خروجی متفاوت در نورون‌های پنهان متفاوت استفاده شوند. $h_i(\mathbf{x})$ می‌تواند به صورت زیر تعریف شود

$$h_i(\mathbf{x}) = G_i(\mathbf{w}_i, b_i, \mathbf{x}), \quad \mathbf{w}_i \in \mathbb{R}^D, b_i \in \mathbb{R}$$

$G(\mathbf{w}, b, \mathbf{x})$ یک تابع غیرخطی قطعه به قطعه پیوسته است، $\mathbf{w}_i^T = [w_{i1}, w_{i2}, ..., w_{iD}]$ بردار وزن ورودی است که نورون‌های ورودی را به iامین نورون پنهان متصل می‌کند و b_i بایاس iامین نورون پنهان است . ضرب داخلی بردارهای $\mathbf{w_i}$ و $\mathbf{x}$ در فضای $\mathbb{R}^D$ با $\mathbf{w_i}^T\mathbf{x}$ نشان داده می‌شود. هدف ELM مینیمم‌سازی خطای آموزش $\|\mathbf{T} - \mathbf{H}\boldsymbol{\beta}\|^2$ و نرم وزن‌های خروجی $\|\boldsymbol{\beta}\|^2$ است.

همان طور که هوانگ و همکاران نامیده اند، $\mathbf{H}$ ماتریس خروجی لایه پنهان و متشکل از $(\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_M})$،

جدول ۱-۱: توابع نگاشت غیرخطی [۶]

Activity Function Name	Activity Function Criterion
Sigmoid Function	$G(\mathbf{w}, b, \mathbf{x}) = \frac{1}{1+exp(-\mathbf{w}^T\mathbf{x}+b)}$
Sine Function	$G(\mathbf{w}, b, \mathbf{x}) = sin(\mathbf{w}^T\mathbf{x} + b)$
Hardlim Function	$G(\mathbf{w}, b, \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$
Triangular Basis Function	$G(\mathbf{w}, b, \mathbf{x}) = \begin{cases} 1 - \lvert\mathbf{w}^T\mathbf{x} + b\rvert & \text{if } \lvert\mathbf{w}^T\mathbf{x} + b\rvert < 1 \\ 0 & \text{otherwise} \end{cases}$
Multiquadric Function	$G(\mathbf{w}, b, \mathbf{x}) = exp(-b \lVert\mathbf{x} - \mathbf{w}\rVert^2)$

$(b_1, b_2, ..., b_M)$ و $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_N})$ به صورت زیر است

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & \dots & h_M(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \dots & h_M(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} g(\mathbf{w}_1^T \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_M^T \mathbf{x}_1 + b_M) \\ \vdots & \vdots & \vdots \\ g(\mathbf{w}_1^T \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_M^T \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M}$$

توجه داشته باشید که در این بخش تعداد نورون خروجی (K) برابر یک فرض شده است پس ماتریس خروجی مطلوب $\mathbf{T}_{N\times K}$ و ماتریس وزن‌های خروجی $\boldsymbol{\beta}_{M\times K}$ یک بردار محسوب می‌شوند.

درواقع، ELM یک SLFN را در دو مرحله آموزش می‌دهد: (۱) نگاشت ویژگی تصادفی و (۲) حل پارامترهای خطی. در مرحله اول، ELM لایه پنهان را به طور تصادفی مقداردهی اولیه می‌کند تا به وسیله برخی توابع نگاشت غیرخطی داده ورودی را به یک فضای ویژگی (فضای ویژگی ELM) نگاشت کند. مرحله نگاشت ویژگی تصادفی ELM را از بسیاری الگوریتم‌های یادگیری موجود هم چون SVM که توابع کرنلی را برای نگاشت استفاده می‌کند، متفاوت می‌سازد. برخی از توابع نگاشت غیرخطی در ELM که می‌توانند هر تابع غیرخطی قطعه به قطعه پیوسته باشند، در جدول ۱-۱ آورده شده است. در ELM، پارامترهای نورون پنهان $(\mathbf{w}, b)$ به جای این که آموزش داده شوند مطابق با هر توزیع احتمال پیوسته (مستقل از داده آموزشی) به طور تصادفی تولید می‌شوند که کارایی قابل ملاحظه‌ای را نسبت به شبکه‌های عصبی BP سنتی موجب می‌شود.

در مرحله دوم یادگیری ELM، وزن‌های اتصال دهنده لایه پنهان و لایه خروجی، با $\boldsymbol{\beta}$ نشان داده می‌شود، با به حداقل رساندن مربعات خطای تقریب و نرم وزن‌های خروجی حل می‌شوند

$$\min_{\boldsymbol{\beta}\in M\times K} \lVert\mathbf{T} - \mathbf{H}\boldsymbol{\beta}\rVert^2 + \lVert\boldsymbol{\beta}\rVert^2 \qquad (۱۵-۱)$$

جواب بهینه مسئله (۱-۱۵) برابر است با

الگوریتم ۱-۲ الگوریتم آموزش ELM [۱۱]

ورودی: مجموعه آموزشی $\aleph = \{(\mathbf{x}_i, t_i)|\mathbf{x}_i \in \mathbb{R}^D, t_i \in \mathbb{R}^K, i = ۱, ۲, ..., N\}$ ، تابع فعالیت g و M نورون پنهان.

خروجی: وزن خروجی (بین لایه پنهان و لایه خروجی)

۱: انتساب به طور تصادفی وزن ورودی ($\mathbf{w}_i$) و بایاس (b_i)، $(i = ۱, ۲, ..., M)$.

۲: محاسبه ماتریس خروجی لایه پنهان ($\mathbf{H}$) .

۳: محاسبه وزن خروجی ($\boldsymbol{\beta}$) با $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$ است، که $\mathbf{H}^{\dagger}$ معکوس تعمیم یافته مور-پنروز $\mathbf{H}$ و $\mathbf{T}$ ماتریس خروجی مطلوب است.

$$\boldsymbol{\beta}^* = \mathbf{H}^{\dagger}\mathbf{T}$$

که $\mathbf{H}^{\dagger}$ معکوس تعمیم یافته مور-پنروز ماتریس $\mathbf{H}$ است.

اگر $\mathbf{H}^T\mathbf{H}$ نامنفرد است $\mathbf{H}^{\dagger}$ به صورت $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-۱}\mathbf{H}^T$

اگر $\mathbf{H}\mathbf{H}^T$ نامنفرد است $\mathbf{H}^{\dagger}$ به صورت $\mathbf{H}^{\dagger} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-۱}$

محاسبه می‌شود که در ادامه بیشتر توضیح داده خواهد شد. در نتیجه تقریب نمونه‌ها با خطای صفر به وسیله انتخاب مناسب $\boldsymbol{\beta_i}$ ، $\mathbf{w_i}$ و b_i است به طوری که شرط زیر برقرار باشد

$$\|\mathbf{t_j} - f_M(\mathbf{x})\| = ۰ \qquad , (j = ۱, ۲, ..., N)$$

شرط فوق معادل رابطه زیر است

$$f_M(\mathbf{x_j}) = \mathbf{t_j} \qquad , (j = ۱, ۲, ..., N) \tag{۱۶-۱}$$

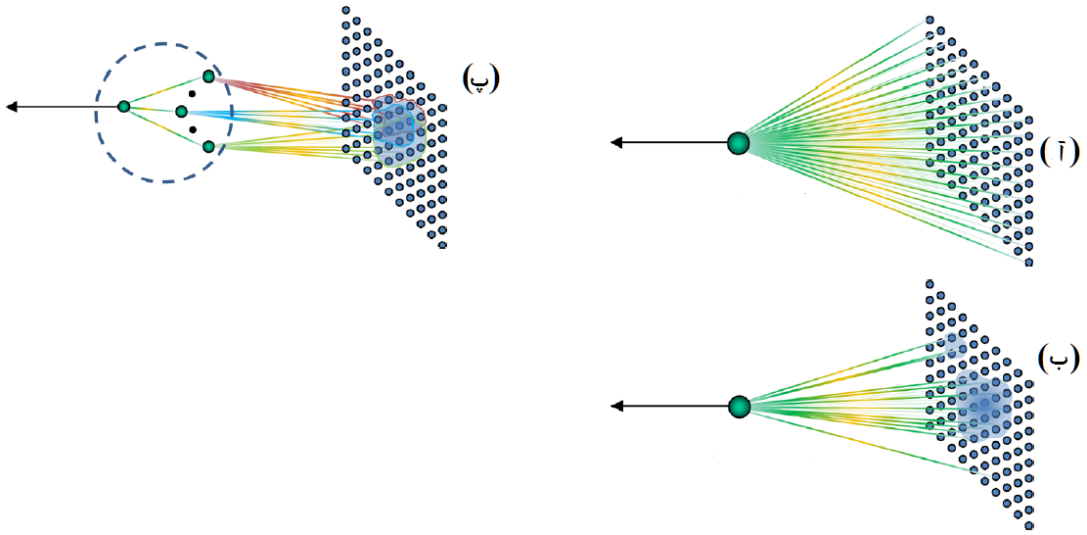
به تعبیری دیگر باید $\mathbf{H}\,\boldsymbol{\beta} = \mathbf{T}$ برقرار باشد. با توجه به مطالب فوق، الگوریتم ELM پیشنهادی هوانگ و همکاران می‌تواند به صورت الگوریتم ۱-۲ خلاصه شود (برنامه متلب در صفحه ۹۸ آمده است).

۱-۳-۱ ویژگی‌های ماشین یادگیر نهایی

یکی از ویژگی‌های برجسته ELM [۶] این است که پارامترهای نورون پنهان لازم نیست تنظیم شوند. یعنی

الف) پارامترهای نورون پنهان ممکن است به طور تصادفی تولید شوند.

ب) اگرچه نورون‌های پنهان لازم نیست به طور تصادفی تولید شوند، آن‌ها نیاز به تنظیم هم نخواهند داشت.



شکل ۱-۷: آ)نورون پنهان با اتصال کامل در ELM. ب)نورون پنهان با اتصال محلی/اتصال تصادفی در ELM. پ)نورون ترکیبی از چند نورون در ELM

به عنوان مثال، یک نورون پنهان در لایه بعدی می‌تواند یک جمع خطی یا تبدیل غیرخطی از برخی نورون‌های به طور تصادفی تولید شده در لایه قبلی باشد. در این مورد برخی از نورون‌ها به طور تصادفی تولید می‌شوند و برخی نمی‌شوند، اما هیچ کدام از آن‌ها تنظیم نمی‌شوند. "تنها پارامتری که نیاز به یادگیری دارد وزن بین لایه پنهان و لایه خروجی است".

در بالا گفته شد که پارامترهای نورون پنهان ممکن است به طور تصادفی تولید شوند، سه سطح تصادفی در ELM وجود دارد [۶] (برای جزئیات بیشتر شکل ۱-۷ را ببینید).

آ) پارامترهای نورون پنهان با اتصالات کامل، به طور تصادفی تولید می‌شوند.

ب) اتصال می‌تواند به طور تصادفی تولید شود، همه نورون‌های ورودی به یک نورون پنهان خاص متصل نمی‌شوند. احتمالاً برخی نورون‌های ورودی در برخی زمینه محلی به یک نورون پنهان متصل می‌شوند.

پ) هر نورون پنهان در ELM می‌تواند یک زیرشبکه از نورون‌های دیگر باشد که در آن یادگیری ویژگی می‌تواند به نحو احسن اجرا شود.

ELM الگوریتمی نسبتاً سریع است. این به دلیل انتخاب به طور تصادفی وزن‌های ورودی ($\mathbf{w_i}$) و بایاس‌ها (b_i) به جای انتخاب SLFNs است. به طور کلی، آموزش ELM شامل دو مرحله است:

مرحله اول. ساخت لایه پنهان با استفاده از تعداد ثابتی نورون نگاشت به طور تصادفی تولید شده است که (تابع نگاشت) می‌تواند هر تابع غیرخطی قطعه به قطعه پیوسته مانند تابع سیگموئید که در زیر آورده شده، باشد.

۱) تابع سیگموئید

$$g(\mathbf{x},\theta)=\frac{۱}{۱+exp(-(\mathbf{w}^T\mathbf{x}+b))}$$

که $\theta=\{\mathbf{w},b\}$ پارامترهای تابع نگاشت هستند. در این مرحله تعدادی نورون پنهان که داده‌ها را از فضای ورودی به فضای ویژگی M بعدی نگاشت می‌کنند، به طور تصادفی تولید می‌شوند.

مرحله دوم. هدف ELM حل وزن‌های خروجی از طریق به حداقل رساندن مجموع مربعات خطا و نرم وزن‌های خروجی، با استفاده از فرمول زیر است.

$\min_{\boldsymbol{\beta}\in R^{M\times K}} \frac{۱}{۲}\|\boldsymbol{\beta}\|^۲+\frac{C}{۲}\sum_{i=۱}^{N}\|\mathbf{e_i}\|^۲$

$s.t \quad h(\mathbf{x_i})\,\boldsymbol{\beta}=\mathbf{t_i}^T-\mathbf{e_i}^T, \qquad i=۱,...,N$

که در بخش ۱-۳-۳ حل این مسئله بیان خواهد شد. خروجی شبکه مطابق رابطه زیر حاصل خواهد شد.

$$f_M(\mathbf{x_i})=h(\mathbf{x_i})\,\boldsymbol{\beta} \qquad i=۱,...,N$$

۱-۳-۲ نظریه‌های ماشین یادگیر نهایی

هوانگ سه اصل یادگیری ELM را تحلیل کرده است. از نقطه نظر قابلیت یادگیری(به عنوان مثال، قابلیت تقریب سراسری، قابلیت طبقه بندی) نظریه‌های ELM تقریباً برای تمام انواع نورون‌های پنهان استفاده شده در کاربردهای واقعی و احتمالاً در مکانیسم یادگیری بیولوژیکی مناسب هستند [۲۰].

اصل یادگیری اول: نورون‌های پنهان SLFNs تقریباً با هر تابع فعالیت غیرخطی قطعه به قطعه پیوسته یا ترکیب خطی آن‌ها می‌توانند مطابق با هر نمونه‌گیری توزیع احتمال پیوسته به طور تصادفی تولید شوند و چنین نورون‌های پنهان می‌توانند مستقل از نمونه‌های آموزشی و محیط یادگیری باشند. علاوه بر این چارچوب یادگیری ELM ثبات یادگیری و عملکرد تعمیم را نیز در نظر می‌گیرد.

اصل یادگیری دوم: به منظور ثبات یادگیری و عملکرد تعمیم باید نرم وزن‌های خروجی SLFNs تعمیم یافته طبق برخی محدودیت‌های بهینه سازی کوچکتر باشند.

اصل یادگیری سوم: از نقطه نظر بهینه سازی، نورون‌های خروجی SLFNs باید بدون بایاس باشند(یا بایاس صفر تنظیم شود).

در این بخش کارهای نظری ازجمله نظریه درون‌یابی، قابلیت تقریب سراسری و قابلیت طبقه‌بندی ELM مرور می‌شوند.

نظریه درون‌یابی

قابلیت یادگیری ELM می‌تواند از نقطه نظر درون‌یابی تفسیر شود.

قضیه ۱-۳-۱. فرض کنید که توزیع حاشیه‌ای $\rho_{\mathbf{X}}$ با توجه به اندازه لبگ روی $\mathbf{X}$ به طور مطلق پیوسته است و تابع فعالیت به کار برده شده در *ELM* چندجمله‌ای جبری باشد، یعنی، $\phi(\boldsymbol{\beta}_i, \mathbf{x}) = (\mathbf{w}_i^T \mathbf{x} + b_i)^s$ با $\boldsymbol{\beta}_i = (\mathbf{w}_i, b_i) \in [۰, ۱]^{D+۱}$ به طور تصادفی تعیین شده است. اگر $M \leqslant N$ باشد، آن گاه ماتریس خروجی لایه پنهان $\mathbf{H} = (r_{ji})_{N \times M}$ که در آن $r_{ji} = (\mathbf{w}_i^T \mathbf{x}_j + b_i)^s$ است قریب به یقین[۱] از مرتبه ستونی کامل است. علاوه بر این، $\mathbf{H}^T\mathbf{H}$ قریب به یقین معکوس پذیر است *[۱۶]*.

برهان. چون $M \leqslant N$ است، برای اثبات قضیه کافی است ثابت کنیم که زیرماتریس معکوس پذیر $\mathbf{R}_{M \times M}$ از $\mathbf{H}$ با درایه‌های $r_{ji} = (\mathbf{w}_i^T \mathbf{x}_j + b_i)^s$ وجود دارد که $j = ۱, ..., N$ و $i = ۱, ..., M$. به این منظور

$$\mathbf{B}_M := \{(\mathbf{x}_۱, ..., \mathbf{x}_M) \in \mathbf{X}^M : det\, \mathbf{R}_M = ۰\}$$

به عبارتی دیگر ثابت می‌کنیم $\forall M \leqslant N \;\; \mathcal{L}(\mathbf{B}_M) = ۰$. این با استقرا روی M ثابت می‌شود. بدیهی است که برای $M = ۱$ درست است. فرض کنید برای $M = n$ $(n \leqslant N - ۱)$ درست است، یعنی $\mathcal{L}(\mathbf{B}_n) = ۰$. بنابراین ثابت می‌شود که $\mathcal{L}(\mathbf{B}_{n+۱}) = ۰$.

اولین n سطر زیرماتریس $\mathbf{R}_n$ معکوس پذیر است، بنابراین به ازای هر $\boldsymbol{\alpha}_{n+۱} := (r_{n+۱,۱}, ..., r_{n+۱,n})$ دلخواه، ضرایب $c_i := c_i(\mathbf{x}_۱, ..., \mathbf{x}_n) \in \mathbb{R}$ وجود دارد، همه صفر نیستند، به طوری که

$$\boldsymbol{\alpha}_{n+۱} = c_۱\boldsymbol{\alpha}_۱ + c_۲\boldsymbol{\alpha}_۲ + ... + c_n\boldsymbol{\alpha}_n.$$

$\boldsymbol{\alpha}_j = (r_{j۱}, ..., r_{jn})$ ، j امین سطر از $\mathbf{H}$ است. اکنون از این ادعا استفاده می‌شود تا ثابت کنیم که یک زیرماتریس $\mathbf{R}_{n+۱}$ وجود دارد به طوری که $\mathcal{L}(\mathbf{B}_{n+۱}) = ۰$. با نگاه به $n + ۱$ امین ستون از $\mathbf{H}$، می‌توان فهمید که $\mathbf{R}_{n+۱}$ معکوس پذیر است اگر و تنها اگر

$$r_{n+۱,n+۱} \neq c_۱ r_{۱,n+۱} + ... + c_n r_{n,n+۱}$$

[۱] almost surely

یا به طور معادل

$$(\mathbf{w}_{n+1}^T \mathbf{x}_{n+1} + b_{n+1})^s \neq c_1(\mathbf{w}_{n+1}^T \mathbf{x}_1 + b_{n+1})^s + ... + c_M(\mathbf{w}_{n+1}^T \mathbf{x}_n + b_{n+1})^s.$$

به عبارت دیگر، $\mathbf{R}_{n+1}$ معکوس پذیر است اگر و تنها اگر $\mathbf{x}_{n+1}$ در مجموعه زیر وجود نداشته باشد

$$\mathbf{D}_n(\mathbf{x}_1, ..., \mathbf{x}_n) := \{\mathbf{x} \in \mathbf{X} : (\mathbf{w}_{n+1}^T \mathbf{x} + b_{n+1})^s = c_1(\mathbf{w}_{n+1}^T \mathbf{x} + b_{n+1})^s + ... + c_n(\mathbf{w}_{n+1}^T \mathbf{x} + b_{n+1})^s\}.$$

برای هر $(\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathbf{X}^n$، $\mathbf{D}_n$ به طور واضح یک مجموعه صفر از برخی چندجمله‌ای جبری است، بنابراین با لم ۱-۲-۳۵، $\mathbf{D}_n$ دارای اندازه لبگ صفر در $\mathbf{X}$ است. از آن جایی که

$$\mathbf{B}_{n+1} \subset \{(\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x}_{n+1}) \in \mathbf{X}^{n+1} : \mathbf{x}_{n+1} \in \mathbf{D}_n(\mathbf{x}_1, ..., \mathbf{x}_n)\}$$

با توجه به قضیه فوبینی[1] [۲۱] می‌بینید که

$$\begin{aligned}
&\mathcal{L}(\mathbf{B}_{n+1}) \\
&= \int_{\mathbf{x}^{n+1}} \chi_{\mathbf{B}_{n+1}}(\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x}_{n+1}) d\mathbf{x}_1 ... d\mathbf{x}_n d\mathbf{x}_{n+1} \\
&= \int_{\mathbf{x}^n} \left(\int_{\mathbf{x}} \chi_{\mathbf{B}_{n+1}}(\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x}_{n+1}) d\mathbf{x}_{n+1} \right) d\mathbf{x}_1 ... d\mathbf{x}_n \\
&\leqslant \int_{\mathbf{x}^n} \mathcal{L}\left(\mathbf{D}_n(\mathbf{x}_1, ..., \mathbf{x}_n)\right) d\mathbf{x}_1 ... d\mathbf{x}_n = 0
\end{aligned}$$

یعنی، $\mathcal{L}(\mathbf{B}_{n+1}) = 0$، استقرا تمام می‌شود. بنابراین، اثبات شد که $\mathcal{L}(\mathbf{B}_M) = 0 \quad \forall M \leqslant N$. با توجه به $M \leqslant N$، معادله فوق نشان می‌دهد که ماتریس مربع $(\mathbf{H}^T\mathbf{H})_{M \times M}$ تقریباً به ازای هر انتخاب $\mathbf{x}_1, ..., \mathbf{x}_M$ معکوس پذیر است. از آن جایی که توزیع $\rho_{\mathbf{X}}$ با توجه به اندازه لبگ $\mathcal{L}$ به طور مطلق پیوسته است، مجموعه $\mathbf{B}_M$ نیز با توجه به $\rho_{\mathbf{X}}$ دارای اندازه صفر است.

□

این قضیه ثابت می‌کند که برای هر مجموعه آموزشی داده شده، یک شبکه ELM وجود دارد که با احتمال یک خطای آموزش کوچکی دارد و تعداد نورون‌های پنهان در آن بیشتر از نمونه‌های آموزشی نیست. در واقع، اگر تعداد نورون‌های پنهان برابر تعداد نمونه‌های آموزشی باشد، آن گاه با احتمال یک، خطای آموزشی به صفر کاهش می‌یابد. بنابراین، ELM می‌تواند هر مجموعه آموزشی با تعداد نورون‌های به اندازه کافی بزرگ را برازش کند.

[1] Fubini

قابلیت تقریب سراسری با نورون‌های تصادفی

قابلیت تقریب سراسری SLFNs در دهه‌های گذشته به خوبی مورد مطالعه قرار گرفته است. با این حال، معمولاً فرض می‌شود که تابع فعالیت نورون‌های پنهان پیوسته و مشتق پذیر است و پارامترهای نورون‌های پنهان در طول آموزش باید تنظیم شوند. در الگوی یادگیری ELM، پارامترهای لایه پنهان به جای این که آموزش داده شوند به طور تصادفی تولید می‌شوند. هوانگ و چن[1] در [۱۰] ثابت کردند که حتی با تولید به طور تصادفی لایه پنهان، ELM یک یادگیرنده سراسری است. یک SLFN با بردار وزن ورودی ($\mathbf{w}$) و بایاس (b) به طور تصادفی تولید شده دارای قابلیت تقریب سراسری است. تحقیقات درباره قابلیت تقریب SLFN روی دو جنبه تمرکز یافته است: تقریب سراسری روی مجموعه‌های ورودی فشرده و تقریب در مجموعه‌ای شامل تعداد متناهی نمونه‌های آموزشی [۱۰]. فرض کنید

$\mathbf{f}$: تابع هدف پیوسته

$\mathbf{f}_M$: تابع شبکه با M نورون پنهان

$\mathbf{e}_M$: تابع باقیمانده خطا با M نورون پنهان، $\mathbf{e_M} \equiv \mathbf{f} - \mathbf{f}_M$

$\mathbf{g}$: تابع فعالیت غیرخطی قطعه به قطعه پیوسته

$\mathbf{g}$ و $\mathbf{f}$ متعلق به فضای L^2 هستند (چون $\mathbf{X}$ مجموعه‌ای فشرده و هر دو تابع پیوسته‌اند بنابراین هر دو انتگرال پذیرند) $\mathbf{e}_M$ هم متعلق به فضای L^2 است. در برهان مسئله فرض شده که شبکه دارای یک نورون خروجی است .

ملاحظه ۱-۳-۲. دنباله تابع $\{g_M = g(\mathbf{w}_M^T \mathbf{x} + b_M)\}$ به طور تصادفی تولید می‌شوند اگر پارامترهای متناظر $(\mathbf{w}_M, b_M)$ مبتنی بر نمونه گیری توزیع احتمال پیوسته از فضای $\mathbb{R}^D \times \mathbb{R}$ به طور تصادفی تولید شوند.

قضیه ۱-۳-۳. با فرض هر نوع تابع غیرخطی قطعه به قطعه پیوسته $\mathbf{g} : \mathbb{R} \to \mathbb{R}$ برای نورون‌های پنهان افزایشی، به ازای هر تابع هدف پیوسته $\mathbf{f}$ و هر دنباله تابع به طور تصادفی تولید شده $\{\mathbf{g}_M\}$ ، اگر رابطه زیر برقرار باشد

$$\beta_M = \frac{< \mathbf{e}_{M-1}, \mathbf{g}_M >}{\|\mathbf{g}_M\|^2}$$

با احتمال یک داریم

$$\lim_{M \to \infty} \|\mathbf{f} - \mathbf{f}_M\| = 0$$

[1] Chen

برهان. ابتدا ثابت می‌شود عبارت $\|\mathbf{e}_M\| = \|\mathbf{f} - (\mathbf{f}_{M-1} + \mathbf{g}_M\beta_M)\|$ به مینیمم مقدار خود دست می‌یابد اگر و تنها اگر $\beta = \beta_M = \frac{<\mathbf{e}_{M-1}, \mathbf{g}_M>}{\|\mathbf{g}_M\|^2}$ و دنباله $\{\|\mathbf{e}_M\|\}$ همگرا می‌شود. سپس $\lim_{M\to\infty} \mathbf{e}_M = 0$ ثابت می‌شود.

الف) فرض کنید

$$\Delta = \|\mathbf{e}_{M-1}\|^2 - \|\mathbf{e}_M\|^2 \qquad (۱-۱۷)$$

اکنون رابطه (۱-۱۷) به صورت زیر بسط داده می‌شود

$$\begin{aligned}
\Delta &= \|\mathbf{e}_{M-1}\|^2 - \|\mathbf{e}_M\|^2 \\
&= \|\mathbf{e}_{M-1}\|^2 - \|\mathbf{f} - (\mathbf{f}_{M-1} + \mathbf{g}_M\beta_M)\|^2 \\
&= \|\mathbf{e}_{M-1}\|^2 - \|(\mathbf{f} - \mathbf{f}_{M-1}) - \mathbf{g}_M\beta_M\|^2 \\
&= \|\mathbf{e}_{M-1}\|^2 - \|\mathbf{e}_{M-1} - \mathbf{g}_M\beta_M\|^2 \\
&= \|\mathbf{e}_{M-1}\|^2 - (\|\mathbf{e}_{M-1}\|^2 + \beta_M^2\|\mathbf{g}_M\|^2 - 2\beta_M <\mathbf{e}_{M-1}, \mathbf{g}_M>) \\
&= \|\mathbf{e}_{M-1}\|^2 - \|\mathbf{e}_{M-1}\|^2 - \beta_M^2\|\mathbf{g}_M\|^2 + 2\beta_M <\mathbf{e}_{M-1}, \mathbf{g}_M>
\end{aligned}$$

در نتیجه عبارت زیر به دست می‌آید

$$\Delta = 2\beta_M <\mathbf{e}_{M-1}, \mathbf{g}_M> -\beta_M^2\|\mathbf{g}_M\|^2 \qquad (۱-۱۸)$$

اگر در رابطه (۱-۱۸) عبارت $\beta_M = \frac{<\mathbf{e}_{M-1}, \mathbf{g}_M>}{\|\mathbf{g}_M\|^2}$ جایگزین شود، Δ طبق رابطه زیر به دست می‌آید

$$\Delta = \frac{<\mathbf{e}_{M-1}, \mathbf{g}_M>^2}{\|\mathbf{g}_M\|^2}$$

بنابراین Δ به بیشترین مقدار خود رسیده است، چون از رابطه (۱-۱۸) نسبت به β_M مشتق گرفته و مساوی صفر قرار داده می‌شود

$$2 <\mathbf{e}_{M-1}, \mathbf{g}_M> -2\beta_M\|\mathbf{g}_M\|^2 = 0 \qquad (۱-۱۹)$$

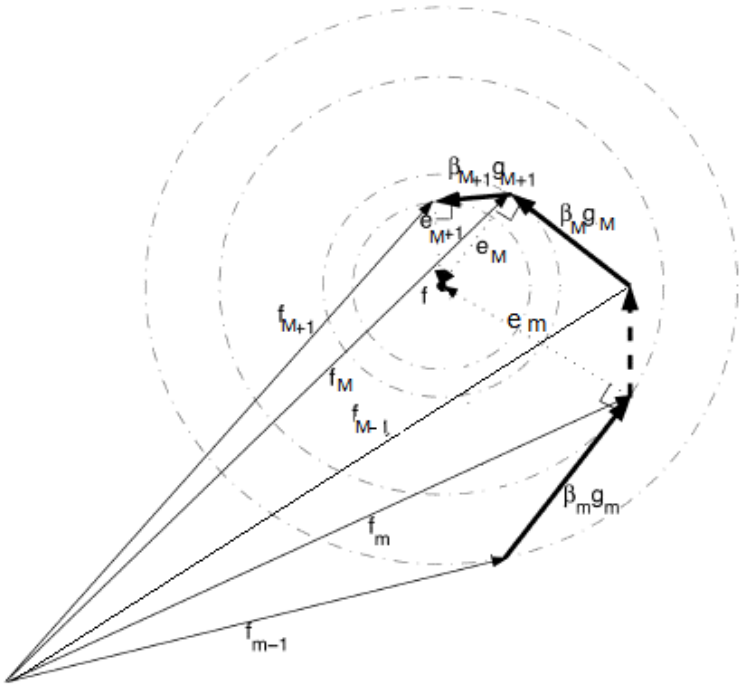
$$\beta_M = \frac{<\mathbf{e}_{M-1}, \mathbf{g}_M>}{\|\mathbf{g}_M\|^2}$$

سپس از رابطه (۱-۱۹) مشتق دوم می‌گیریم و به رابطه زیر می‌رسیم

$$-۲ \|g_M\|^۲$$

و جواب حاصل از مشتق اول در مشتق دوم قرار می‌گیرد و مقداری کوچکتر از صفر به دست می‌آید که نشان‌دهنده این است تابع مقعر بوده و دارای نقطه ماکزیمم است، پس Δ به بیشترین مقدار خود رسیده است. هم چنین طبق رابطه (۱-۱۷) با افزایش مقدار Δ، $\|\mathbf{e}_M\|$ کاهش می‌یابد.

از نقطه نظر فضای تابع، هنگامی که $\|\mathbf{e}_M\| = \|\mathbf{f} - (\mathbf{f}_{M-۱} + \mathbf{g}_M\beta_M)\|$ به کمترین مقدارش دست یابد $\mathbf{e}_M \perp \mathbf{g}_M$ (شکل ۱-۸ را ببینید). اکنون روابط جبری متعامد بودن این دو بردار به صورت زیر بررسی می‌شود.



شکل ۱-۸: $\mathbf{e}_M$ همیشه بر $\mathbf{g}_M$ عمود است: $\mathbf{e}_M \perp \mathbf{g}_M$. دنباله $\{\|e_M\|\}$ در حال کاهش است و از پایین به صفر محدود می‌شود [۱۰].

دو بردار زمانی برهم عمودند که ضرب داخلی آن دو بردار مساوی صفر شود، یعنی

$$< \mathbf{e}_M, \mathbf{g}_M > = < \mathbf{f} - (\mathbf{f}_{M-۱} + \mathbf{g}_M\beta_M), \mathbf{g}_M > = < (\mathbf{f} - \mathbf{f}_{M-۱}) - \mathbf{g}_M\beta_M, \mathbf{g}_M >$$

$$= < \mathbf{e}_{M-۱} - \mathbf{g}_M\beta_M, \mathbf{g}_M > = < \mathbf{e}_{M-۱}, \mathbf{g}_M > - \beta_M < \mathbf{g}_M, \mathbf{g}_M >$$

$$= < \mathbf{e}_{M-۱}, \mathbf{g}_M > - \frac{< \mathbf{e}_{M-۱}, \mathbf{g}_M >}{\|\mathbf{g}_M\|^۲} < \mathbf{g}_M, \mathbf{g}_M >$$

نرم در فضای ضرب داخلی با رابطه $\|g_M\| = \sqrt{< g_M, g_M >}$ $\left(\|g_M\|^۲ = < g_M, g_M >\right)$ بیان می‌شود.

اکنون طبق این تعریف داریم

$$< \mathbf{e}_M, \mathbf{g}_M > = < \mathbf{e}_{M-۱}, \mathbf{g}_M > - < \mathbf{e}_{M-۱}, \mathbf{g}_M > = ۰$$

درحقیقت، زمانی که $\beta_M = \frac{<\mathbf{e}_{M-1},\mathbf{g}_M>}{\|\mathbf{g}_M\|^2}$ و $\Delta = \Delta_{max} = \frac{<\mathbf{e}_{M-1},\mathbf{g}_M>^2}{\|\mathbf{g}_M\|^2}$ دنباله $\{\|\mathbf{e}_M\|\}$ کاهش می‌یابد و طبق این خاصیت از نرم که $\forall x \in X\ , \|x\| \geqslant 0$ پس دنباله $\{\|\mathbf{e}_M\|\}$ از پایین به صفر محدود می‌شود. طبق تعریف ۱-۲-۲۳ رابطه زیر را خواهیم داشت

$$\|\mathbf{e}_2 - \mathbf{e}_1\| > \|\mathbf{e}_3 - \mathbf{e}_2\| > ... > \|\mathbf{e}_M - \mathbf{e}_{M-1}\| > \rightarrow \lim_{M\to\infty} \|\mathbf{e}_M\| = r$$

پس می‌توان نتیجه گرفت که دنباله $\{\|\mathbf{e}_M\|\}$ دنباله‌ای کُشی و همگراست.

ب) اکنون با برهان خلف ثابت می‌شود که $\lim_{M\to\infty} \|\mathbf{e}_M\| = r = 0$.

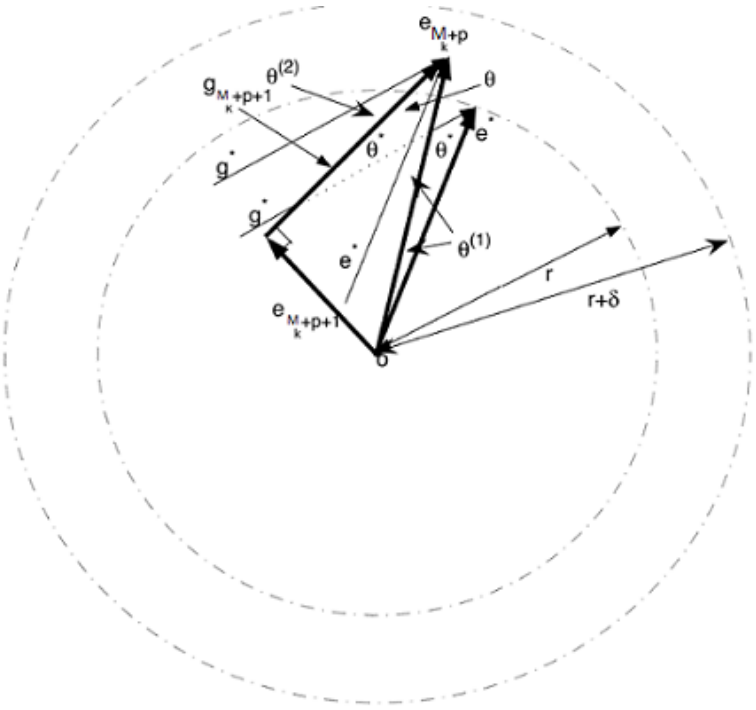
ب.۱) از آن جایی که دنباله $\{\|\mathbf{e}_M\|\}$ همگراست، $\exists\, r \geqslant 0 \quad s.t \quad \lim_{M\to\infty} \|\mathbf{e}_M\| = r$. فرض کنید $r > 0$ باشد: دنباله $\{\|\mathbf{e}_M\|\}$ نزولی و از پایین به r محدود می‌شود، یعنی برای تمامی مقادیر صحیح مثبت M ، $r \leqslant \|\mathbf{e}_M\|$. همچنین،

$$\forall \delta > 0, \exists \tilde{m}_1 > 0 \quad s.t \quad if\ M > \tilde{m}_1; \quad r \leqslant \|\mathbf{e}_M\| < r + \delta$$

که تعداد نامتناهی از $\mathbf{e}_M$ ها $(\forall M > \tilde{m}_1)$ به وسیله یک مجموعه فشرده پوشش داده می‌شوند. بنابراین زیردنباله $\{\mathbf{e}_{M_k}\}$ ای وجود دارد که به یک حد که با e^* نشان داده شده همگرا می‌شود و $\|\mathbf{e}^*\| = \lim_{k\to\infty} \|\mathbf{e}_{M_k}\| = r$. چون $\mathbf{e}_M$ متعلق به فضای L^2 است بنابراین زیردنباله آن $(\mathbf{e}_{M_k})$ نیز متعلق به این فضا است و طبق لم ۱-۲-۲۸، $\mathbf{e}^* \in L^2$.

علاوه بر این، باید $g^* = g((\mathbf{w}^*)^T\, \mathbf{x} + b^*)$ ای وجود داشته باشد طوری که بر $\mathbf{e}^*$ متعامد نباشد. در غیر این صورت، $\mathbf{e}^*$ بر فضای تولید شده $\{g(\mathbf{w}^T\, \mathbf{x} + b) : (\mathbf{w}, b) \in \mathbb{R}^D \times \mathbb{R}\}$ متعامد است و این با واقعیتی که فضای تولید شده $\{g(\mathbf{w}^T\, \mathbf{x} + b) : (\mathbf{w}, b) \in \mathbb{R}^D \times \mathbb{R}\}$ مطابق با لم ۱-۲-۲۹ در L^2 چگال است، متناقض است . در نظر بگیرید $\theta_{(\mathbf{g}^*,\mathbf{e}^*)}$ نشان‌دهنده زاویه بین $\mathbf{g}^*$ و $\mathbf{e}^*$ و $0 \leqslant \theta_{(\mathbf{g}^*,\mathbf{e}^*)} < \frac{\pi}{2}$ (شکل ۱-۹ را ببینید).

ب.۲) فرض کنید $\theta_{(\mathbf{e}_M,\mathbf{e}^*)}$ زاویه بین $\mathbf{e}_M$ و $\mathbf{e}^*$ باشد، با این شرط که $0 \leqslant \theta_{(\mathbf{e}_M,\mathbf{e}^*)} \leqslant \frac{\pi}{2}$. هر مقدار مثبت کوچک $\xi < 1 - \sin(\theta_{(\mathbf{g}^*,\mathbf{e}^*)})$ می‌تواند انتخاب شود. چون $\lim_{k\to\infty} \|\mathbf{e}_{M_k}\| = \|\mathbf{e}^*\| = r$ ، $\tilde{m}_2$



شکل ۱-۹: برای حد $\mathbf{e}^*$ از زیردنباله $\{\mathbf{e}_M\}$، اگر $\|\mathbf{e}^*\| = r \neq 0$ باشد $\mathbf{g}^*$ ای وجود دارد که بر $\mathbf{e}^*$ متعامد نیست، آن گاه $\mathbf{e}_{M_k+p+1}$ ای وجود دارد به طوری که $\|\mathbf{e}_{M_k+p+1}\| < \lim_{M\to\infty}\|\mathbf{e}_M\| = r$ ، که متناقض است با واقعیتی که $\|\mathbf{e}_{M_k+p+1}\| \geqslant r$ است. بنابراین r باید صفر باشد. برای سادگی، $\theta_{(\mathbf{g}^*,\mathbf{e}^*)}$، $\theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)}$، $\theta_{(\mathbf{g}_{M_k+p+1},\mathbf{g}^*)}$، و $\theta_{(\mathbf{e}_{M_k+p+1},\mathbf{g}_{M_k+p+1})}$ به ترتیب با θ^*، $\theta^{(1)}$، $\theta^{(2)}$ و θ نشان‌داده شده‌اند، $\theta \leqslant \theta^* + \theta^{(1)} + \theta^{(2)}$ [۱۰].

ای وجود دارد و زمانی که $k > \tilde{m}_2$ رابطه زیر را خواهیم داشت

$$\begin{aligned} &\|\mathbf{e}_{M_k}\| < \frac{r}{1-\xi} \\ &\theta_{(\mathbf{e}_{M_k},\mathbf{e}^*)} < \arcsin\left(\frac{1-\xi-\sin(\theta_{(\mathbf{g}^*,\mathbf{e}^*)})}{2}\right) \end{aligned} \qquad (۱-۲۰)$$

چون برای تمام اعداد صحیح مثبت M، $\mathbf{e}_M \perp \mathbf{g}_M$ ، پس بر ضریبی از $\mathbf{g}_M$ یعنی $(\mathbf{g}_M\beta_M)$ نیز عمود است، از طرفی هم داریم

$$\mathbf{g}_M\beta_M = \mathbf{e}_{M-1} - \mathbf{e}_M$$

بنابراین

$$\mathbf{e}_M \perp \mathbf{g}_M\beta_M \Rightarrow \mathbf{e}_M \perp (\mathbf{e}_{M-1} - \mathbf{e}_M)$$

مقدار عبارت $\|\mathbf{e}_{M-1} - \mathbf{e}_M\|^2$ به طریق زیر به دست می‌آید

$$\|\mathbf{e}_{M-1} - \mathbf{e}_M\|^2 = \|\mathbf{e}_{M-1}\|^2 + \|\mathbf{e}_M\|^2 - 2 < \mathbf{e}_{M-1}, \mathbf{e}_M >$$
$$= \|\mathbf{e}_{M-1}\|^2 + \|\mathbf{e}_M\|^2 - 2(< \mathbf{e}_{M-1}, \mathbf{e}_M > - < \mathbf{e}_{M-1} - \mathbf{e}_M, \mathbf{e}_M >)$$
$$= \|\mathbf{e}_{M-1}\|^2 + \|\mathbf{e}_M\|^2 - 2(< \mathbf{e}_{M-1}, \mathbf{e}_M > -(< \mathbf{e}_{M-1}, \mathbf{e}_M > - < \mathbf{e}_M, \mathbf{e}_M >))$$
$$= \|\mathbf{e}_{M-1}\|^2 + \|\mathbf{e}_M\|^2 - 2\|\mathbf{e}_M\|^2$$
$$= \|\mathbf{e}_{M-1}\|^2 - \|\mathbf{e}_M\|^2$$

درنتیجه داریم

$$\lim_{M\to\infty} \|\mathbf{g}_M \beta_M\|^2 = \lim_{M\to\infty} \|\mathbf{e}_{M-1} - \mathbf{e}_M\|^2 = \lim_{M\to\infty}(\|\mathbf{e}_{M-1}\|^2 - \|\mathbf{e}_M\|^2)$$
$$= \lim_{M\to\infty} \|e_{M-1}\|^2 - \lim_{M\to\infty} \|e_M\|^2 = r^2 - r^2 = 0$$

پس برای هر عدد صحیح مثبت m، $\lim_{M\to\infty} \sum_{p=M+1}^{M+m} \|\mathbf{g}_p \beta_p\| = 0$.

بنابر روابط $\|\mathbf{e}_{M_k} - \mathbf{e}_{M_k+m}\| = \left\|\sum_{p=M_k+1}^{M_k+m} \mathbf{g}_p \beta_p\right\|$ و نامساوی مثلثی $(\|x+y\| \leqslant \|x\| + \|y\|)$ داریم

$$\|\mathbf{e}_{M_k} - \mathbf{e}_{M_k+m}\| = \left\| \sum_{p=M_k+1}^{M_k+m} \mathbf{g}_p \beta_p \right\| \leqslant \sum_{p=M_k+1}^{M_k+m} \|\mathbf{g}_p \beta_p\|$$

پس برای هر عدد صحیح مثبت و ثابت m، $\lim_{k\to\infty} \|\mathbf{e}_{M_k} - \mathbf{e}_{M_k+m}\| = 0$.

حد عبارت $\|\mathbf{e}_{M_k+m} - \mathbf{e}^*\|$ به صورت زیر محاسبه می‌شود

$$\lim_{k\to\infty} \|\mathbf{e}_{M_k+m} - \mathbf{e}^*\| = \lim_{k\to\infty} \|\mathbf{e}_{M_k+m} - \mathbf{e}_{M_k} + \mathbf{e}_{M_k} - \mathbf{e}^*\|$$
$$\leqslant \lim_{k\to\infty} \|\mathbf{e}_{M_k+m} - \mathbf{e}_{M_k}\| + \lim_{k\to\infty} \|\mathbf{e}_{M_k} - \mathbf{e}^*\| = 0 \qquad (۲۱\text{-}۱)$$

مطابق با روابط (۲۰-۱) و (۲۱-۱)، $\tilde{m}_3 > \tilde{m}_2$ ای وجود دارد به طوری که $\forall k > \tilde{m}_3$ به رابطه زیر می‌رسیم (شکل ۱-۹ را ببینید)

$$\|\mathbf{e}_{M_k+p}\| < \frac{r}{1-\xi}$$
$$\theta_{(\mathbf{e}_{M_k+p}, \mathbf{e}^*)} < \arcsin\left(\frac{1 - \xi - \sin\left(\theta_{(\mathbf{g}^*, \mathbf{e}^*)}\right)}{2}\right) \qquad (۲۲\text{-}۱)$$

$$0 \leqslant p \leqslant L(M_k)$$

و $L(M_k)$ تابعی از k به صورت زیر است

$$L(M_k) = \max_m\{m : \|\mathbf{e}_{M_k+p}\| < \frac{r}{۱-\xi} \\ , \theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)} < \arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right), ۰ \leqslant p \leqslant m\} \quad (۲۳-۱)$$

بدیهی است که برای هر عدد صحیح مثبت q، $\exists k > \tilde{m}_۳ L(M_k) > q$ ،چون برای هر عدد صحیح مثبت m رابطه زیر برقرار است

$$\lim_{k\to\infty} \sum_{p=M_k+۱}^{M_k+m} \|\mathbf{g}_p \beta_p\| = ۰$$

ب.۳)در نظر بگیرید $\theta_{(\mathbf{g}_M,\mathbf{g}^*)}$ نشان‌دهنده زاویه بین $\mathbf{g}_M$ و $\mathbf{g}^*$ و $۰ \leqslant \theta_{(\mathbf{g}_M,\mathbf{g}^*)} \leqslant \frac{\pi}{۲}$. چون دنباله تابع $\mathbf{g}_M$ به طور تصادفی مبتنی بر نمونه‌گیری توزیع احتمال پیوسته تولید می‌شود، مطابق با لم ۱-۲-۳۰ ، یک عدد صحیح مثبت $\ddot{L}$ وجود دارد به طوری که برای هر بخش پیوسته دنباله تابع $\mathbf{g}_M$ هم چون زیر

$$G_{(M,\ddot{L})} = \{\mathbf{g}_M, \mathbf{g}_{M+۱}, ...\mathbf{g}_{M+\ddot{L}-۱}\}, (M = ۱, ۲, ...)$$

حداقل یک $\mathbf{g}_{M+p} \in G_{(M,\ddot{L})}$ وجود دارد که در رابطه زیر صدق کند

$$\theta_{(\mathbf{g}_{M+p},\mathbf{g}^*)} < \arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right).$$

هم چنین حداقل یک $k > \tilde{m}_۳$ ی وجود دارد به طوری که $\ddot{L} > L(M_k) + ۱$ ، مطابق با لم ۱-۲-۳۰ ، برای بخش پیوسته $G_{(M_k+۱,L(M_k)+۱)} = \{\mathbf{g}_{M_k+۱}, ...\mathbf{g}_{M_k+L(M_k)+۱}\}, (M = ۱, ۲, ...)$ حداقل یک p $(۰ \leqslant p \leqslant L(M_k))$ وجود دارد که در رابطه زیر صدق کند (شکل ۱-۹ را ببینید)

$$\theta_{(\mathbf{g}_{M_k+p+۱},\mathbf{g}^*)} < \arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right). \quad (۲۴-۱)$$

ب.۴) مطابق با روابط (۲۲-۱) و (۲۴-۱) داریم

$$\theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)} < \arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right)$$
$$\sin(\theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)}) < \sin\left(\arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right)\right)$$

$$\sin(\theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)}) < \left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right)$$

$$\theta_{(\mathbf{g}_{M_k+p+۱},\mathbf{g}^*)} < \arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right)$$
$$\sin(\theta_{(\mathbf{g}_{M_k+p+۱},\mathbf{g}^*)}) < \sin\left(\arcsin\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right)\right)$$
$$\sin(\theta_{(\mathbf{g}_{M_k+p+۱},\mathbf{g}^*)}) < \frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}$$

در نظر بگیرید $\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}$ نشان‌دهنده زاویه بین $\mathbf{g}_{M_k+p+۱}$ و $\mathbf{e}_{M_k+p}$. هم چنین روابط زیر برقرارند

$$\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})} \leqslant \theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)} + \theta_{(\mathbf{e}^*,\mathbf{g}_{M_k}+p+۱)}$$

$$\theta_{(\mathbf{e}^*,\mathbf{g}_{M_k+p+۱})} \leqslant \theta_{(\mathbf{g}^*,\mathbf{e}^*)} + \theta_{(\mathbf{g}_{M_k}+p+۱,\mathbf{g}^*)}$$

که $\theta_{(\mathbf{e}^*,\mathbf{g}_{M_k+p+۱})}$ نشان‌دهنده زاویه بین $\mathbf{e}^*$ و $\mathbf{g}_{M_k+p+۱}$ ، از تلفیق دو رابطه فوق خواهیم داشت

$$\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})} \leqslant \theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)} + \theta_{(\mathbf{e}^*,\mathbf{g}_{M_k}+p+۱)}$$
$$\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})} \leqslant \theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)} + \left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)} + \theta_{(\mathbf{g}_{M_k+p+۱},\mathbf{g}^*)}\right)$$

و

$$\sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) \leqslant \sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{e}^*)}\right) + \sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right) + \sin\left(\theta_{(\mathbf{g}_{M_k+p+۱},\mathbf{g}^*)}\right)$$
$$\sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) \leqslant \frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲} + \sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right) + \frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}$$
$$\sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) \leqslant ۲\left(\frac{۱-\xi-\sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)}{۲}\right) + \sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)$$
$$\sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) \leqslant ۱ - \xi - \sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right) + \sin\left(\theta_{(\mathbf{g}^*,\mathbf{e}^*)}\right)$$
$$\sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) \leqslant ۱ - \xi$$

با توجه به روابط $\|\mathbf{e}_{M_k+p}\| < \frac{r}{۱-\xi}$ و $\sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) = \frac{\|\mathbf{e}_{M_k+p+۱}\|}{\|\mathbf{g}_{M_k+p}\|}$ به دست آمده از شکل ۱-۹ در نهایت به رابطه زیر می‌رسیم

$$\|\mathbf{e}_{M_k+p+۱}\| = \|\mathbf{e}_{M_k+p}\| \sin\left(\theta_{(\mathbf{e}_{M_k+p},\mathbf{g}_{M_k+p+۱})}\right) < (\frac{r}{۱-\xi}) * (۱-\xi) = r$$

و رابطه فوق با این واقعیت که هر یک از $\|\mathbf{e}_M\| \geqslant r$ ، متناقض است. بنابراین $r = 0$، یعنی

$$\lim_{M\to\infty} \|\mathbf{e}_M\| = r = 0$$

اثبات کامل شد [۱۰]. □

قابلیت طبقه‌بندی ماشین یادگیر نهایی

شبکه‌های عصبی پیشخور برای ساخت طبقه‌بندهای زیادی به کار برده می‌شوند. تحقیقات زیادی روی قابلیت طبقه‌بندی شبکه‌های عصبی پیشخور چندلایه انجام گرفته است. در این بخش بیان می‌شود که SLFN با هر تابع فعالیت غیرخطی کران دار پیوسته یا هر تابع فعالیت کران دار دلخواه با حدهای نابرابر در بی نهایت می‌تواند نواحی تصمیم‌گیری با اشکال دلخواه را تفکیک کند. با توجه به لم ۱-۲-۳۴ و قضیه ۱-۲-۳۷ داریم

قضیه ۱-۳-۴ (قابلیت طبقه‌بندی). یک *SLFN* با هر تابع فعالیت غیرخطی کران دار پیوسته $g(x)$ و با تعداد نورون‌های پنهان کافی می‌تواند نواحی مجزای دلخواه با هر شکل در $\mathbb{R}^d$ یا Ω را تفکیک کند.

در واقع قضیه فوق نشان می‌دهد که ELM می‌تواند هر مرز تصمیم‌گیری پیچیده در مسئله طبقه‌بندی با تعداد نورون‌های پنهان کافی را تقریب بزند.

۱-۳-۳ ماشین یادگیر نهایی برای طبقه‌بندی

اصولاً ELM برای مسائل یادگیری باناظر به کاربرده می‌شود و یک الگوی یادگیری کارآمد برای طبقه‌بندی و رگرسیون می‌باشد. در **موارد طبقه‌بندی دوکلاسی**، ELM فقط یک نورون خروجی را استفاده می‌کند و برچسب کلاس نزدیک‌تر به مقدار خروجی ELM به عنوان برچسب پیش‌بینی شده کلاس داده ورودی انتخاب می‌شود. برای **طبقه‌بندی چندکلاسی** دو راه‌حل وجود دارد:

۱-۳-۳-۱ راه حل اول

ELM فقط یک نورون خروجی را استفاده می‌کند و در میان برچسب‌های چندکلاس، برچسب کلاس نزدیک‌تر به مقدار خروجی ELM به عنوان برچسب پیش‌بینی شده کلاس داده ورودی انتخاب می‌شود. در این مورد، جواب ELM برای مورد طبقه‌بندی دوکلاسی یک مورد خاص از طبقه‌بندی چندکلاسی می‌باشد. چون ELM می‌تواند

هر تابع هدف پیوسته را تقریب بزند و خروجی طبقه‌بند ELM ، $\mathbf{h}(\mathbf{x})\,\boldsymbol{\beta}$، نیز می‌تواند تا جاییکه ممکن است نزدیک به برچسب‌های کلاس مربوطه باشد، مسئله طبقه‌بندی برای ELM مبتنی بر بهینه سازی مقید با یک نورون خروجی (باتوجه به اهداف ELM : رسیدن به کوچکترین خطای آموزش و کوچکترین نرم وزن‌های خروجی) را می‌توان هم چون زیر فرموله‌بندی کرد [۱۹].

$\min_{\boldsymbol{\beta}\in R^{M\times K}} : \frac{۱}{۲}\|\boldsymbol{\beta}\|^{۲} + \frac{C}{۲}\sum_{i=۱}^{N}\|\mathbf{e}_i\|^{۲}$

$\text{Subject to : } \mathbf{h}(\mathbf{x}_i)\,\boldsymbol{\beta} = \mathbf{t}_i^T - \mathbf{e}_i^T, \quad i = ۱,...,N, \quad \mathbf{e}_i = [e_{i۱}, e_{i۲}, ..., e_{iK}]^T$

اولین جمله در تابع هدف یک عبارت منظم‌سازی است و پیچیدگی مدل را کنترل می‌کند، $\mathbf{e}_i = \mathbf{t}_i - \mathbf{y}_i$ بردار خطای آموزشی است و C ضریب جریمه روی خطاهای آموزشی است. حالا محدودیت در تابع هدف جایگزین و مسئله فوق به مسئله بهینه سازی نامقید تبدیل می‌شود.

$$\min_{\boldsymbol{\beta}\in R^{M\times K}} \mathrm{L_{ELM}} : \frac{۱}{۲}\|\boldsymbol{\beta}\|^{۲} + \frac{C}{۲}\|\mathbf{T} - \mathbf{H}\boldsymbol{\beta}\|^{۲}$$

این مسئله به عنوان مسئله حداقل مربعات منظم[۱] یا در آمار رگرسیون ریج[۲] شناخته می‌شود. با حل این مسئله مبتنی بر اهمیت اندازه متفاوت ماتریس خروجی لایه پنهان $\mathbf{H}$ ، جواب‌های متفاوتی به دست می‌آید.

الف) اگر تعداد نمونه‌های آموزشی بیشتر از تعداد نورون‌های پنهان ، تعداد سطرهای ماتریس $\mathbf{H}$ بیشتر از ستون‌ها ($N > M$) و ماتریس از مرتبه ستونی کامل باشد.

$\nabla L_{ELM} = \boldsymbol{\beta} - C\,\mathbf{H}^T(\mathbf{T} - \mathbf{H}\boldsymbol{\beta}) = ۰$

$\boldsymbol{\beta} - C\,\mathbf{H}^T\mathbf{T} + C\,\mathbf{H}^T\mathbf{H}\boldsymbol{\beta} = ۰$

$\boldsymbol{\beta}(\mathbf{I} + C\mathbf{H}^T\mathbf{H}) - C\,\mathbf{H}^T\mathbf{T} = ۰$

$\boldsymbol{\beta} = \frac{C\,\mathbf{H}^T\mathbf{T}}{\mathbf{I}+C\mathbf{H}^T\mathbf{H}} = \frac{C}{C(\mathbf{H}^T\mathbf{H}+\frac{\mathbf{I}}{C})}\mathbf{H}^T\mathbf{T}$

بنابراین

$$\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C})^{-۱}\mathbf{H}^T\mathbf{T}$$

که $\mathbf{I}$ ماتریس همانی با اندازه $M \times M$ است. در نهایت، تابع تصمیم طبقه‌بند ELM به صورت زیر است

$$f(\mathbf{x}) = sign\left(\mathbf{h}(\mathbf{x})\,(\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C})^{-۱}\mathbf{H}^T\mathbf{T}\right)$$

ب)اگر تعداد نمونه‌های آموزشی کمتر از تعداد نورون‌های پنهان، تعداد سطرهای ماتریس $\mathbf{H}$ کمتر از ستون‌ها

[۱] Regularized Least Squares [۲] Ridge Regression

و ماتریس از مرتبه سطری کامل باشد، منجر به مسئله حداقل مربعات فرومعین[1] می‌شود . در این مورد مقرون به صرفه نیست تا یک ماتریس $M \times M$ را معکوس کنیم. برای حل این مسئله باید $\boldsymbol{\beta}$ محدود شود تایک ترکیب خطی از سطرهای $\mathbf{H}$ باشد. زمانی که $N < M$ و $\mathbf{H}$ از مرتبه سطری کامل است، $\mathbf{HH}^T$ معکوس پذیر است.

$$\boldsymbol{\beta} = \mathbf{H}^T\boldsymbol{\alpha},\ \boldsymbol{\alpha} \in \mathbb{R}^{N\times K} \qquad (۲۵-۱)$$

رابطه زیر در $(\mathbf{HH}^T)^{-1}\mathbf{H}$ ضرب می‌شود.

$$\nabla L_{ELM} = \boldsymbol{\beta} - C\,\mathbf{H}^T(\mathbf{T} - \mathbf{H}\boldsymbol{\beta}) = 0$$
$$(\mathbf{HH}^T)^{-1}\mathbf{H}\boldsymbol{\beta} - C(\mathbf{HH}^T)^{-1}\mathbf{HH}^T(\mathbf{T} - \mathbf{H}\boldsymbol{\beta}) = 0$$
$$(\mathbf{H}^T)^{-1}\mathbf{H}^{-1}\mathbf{H}\boldsymbol{\beta} - C(\mathbf{H}^T)^{-1}\mathbf{H}^{-1}\mathbf{HH}^T(\mathbf{T} - \mathbf{H}\boldsymbol{\beta}) = 0$$
$$(\mathbf{H}^T)^{-1}\boldsymbol{\beta} - C(\mathbf{T} - \mathbf{H}\,\boldsymbol{\beta}) = 0$$

بنابر رابطه (۱-۲۵)، $\boldsymbol{\alpha} = (\mathbf{H}^T)^{-1}\boldsymbol{\beta}$ نتیجه می‌شود.

$$\boldsymbol{\alpha} - C(\mathbf{T} - \mathbf{HH}^T\boldsymbol{\alpha}) = 0$$
$$\boldsymbol{\alpha} - C\mathbf{T} + C\mathbf{HH}^T\boldsymbol{\alpha} = 0$$
$$\boldsymbol{\alpha}(\mathbf{I} + C\mathbf{HH}^T) - C\mathbf{T} = 0$$
$$\boldsymbol{\alpha}C(\mathbf{HH}^T + \tfrac{\mathbf{I}}{C}) = C\mathbf{T}$$
$$\boldsymbol{\alpha} = (\mathbf{HH}^T + \tfrac{\mathbf{I}}{C})^{-1}\mathbf{T}$$

$$\boldsymbol{\beta} = \mathbf{H}^T\boldsymbol{\alpha} = \mathbf{H}^T(\mathbf{HH}^T + \frac{\mathbf{I}}{C})^{-1}\mathbf{T}$$

$\mathbf{I}$ ماتریس همانی با اندازه $N \times N$ است. در نهایت، تابع تصمیم طبقه‌بند ELM به صورت زیر است

$$f(\mathbf{x}) = sign\left(\mathbf{h}(\mathbf{x})\,\mathbf{H}^T(\mathbf{HH}^T + \frac{\mathbf{I}}{C})^{-1}\mathbf{T}\right)$$

۱-۳-۳-۲ راه حل دوم

ELM به جای یک نورون خروجی از چند نورون خروجی استفاده می‌کند و اندیس نورون خروجی با بیشترین مقدار خروجی به عنوان برچسب داده ورودی در نظر گرفته می‌شود. طبقه‌بندهای K کلاسی دارای K نورون خروجی هستند. اگر برچسب کلاس اصلی p است، بردار خروجی K نورون خروجی $\mathbf{t}_i = [0, ..., 0, 1, 0, ..., 0]^T$

[1] under-determined

جدول ۱-۲: مقایسه خطا، دقت طبقه‌بند و زمان آموزش الگوریتم‌های ELM و BP بر روی گل‌های زنبق

نمونه‌ها	الگوریتم	میانگین خطای طبقه‌بندی	میانگین دقت طبقه‌بندی	میانگین زمان آموزش
تست	BP	۰.۱۱۵۲	۰.۸۸۴۸	۰.۲۰۴۰
تست	ELM	۰.۰۴۴۳	۰.۹۵۵۷	۰.۰۰۰۳

است. در این مورد تنها p امین عنصر $\mathbf{t}_i = [t_{i1}, ..., t_{ip}, ..., t_{iK}]^T$ برابر ۱ است و بقیه عناصر مجموعه صفر هستند.

در نظر بگیرید $f_j(\mathbf{x})$ نشان‌دهنده تابع خروجی jامین نورون خروجی است، یعنی،

$$f(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_K(\mathbf{x})]$$

آن گاه پیش‌بینی برچسب کلاس داده ورودی $\mathbf{x}$ به صورت زیر خواهد بود

$$\text{label}(\mathbf{x}) = \arg \max_{i \in \{1,...,K\}} f_i(\mathbf{x})$$

به عنوان مثال، الگوریتم‌های ماشین یادگیر نهایی (ELM) و پس انتشار (BP) به منظور طبقه‌بندی مجموعه داده گل‌های زنبق[1] (مجموعه‌ای شامل ۱۵۰ نمونه و ۴ ویژگی) به ۳ طبقه اجرا شده‌اند و معیارهایی که دو الگوریتم باهم مقایسه شده‌اند، خطای طبقه‌بند(نسبت نمونه‌های اشتباه طبقه‌بندی شده به تمام نمونه‌ها)، دقت طبقه‌بند و زمان آموزش (برحسب ثانیه) بر روی نمونه‌های تست است. در این مقایسه تعداد نورون‌های پنهان (M) برابر ۱۰ است. الگوریتم‌ها ۱۰۰۰ دفعه تکرار شده‌اند و تمامی معیارها به صورت میانگین بیان می‌شود. نتایج اجرا در جدول ۱-۲ آورده شده است. تعداد نمونه‌های آموزشی و تست به ترتیب، برابر۱۲۰ و ۳۰ است. همان طور که در جدول دیده می‌شود ELM نسبت به BP عملکرد بهتری دارد.

[1] FisherIris

فصل ۲

روش‌های آموزش ماشین یادگیر نهایی

این فصل در برگیرنده روش‌های آموزش ماشین یادگیر نهایی است که آن را در سه بخش آورده‌ایم. در بخش اول ماشین یادگیر نهایی افزایشی مبتنی بر تجزیه QR را بیان می‌کنیم، البته در این بخش فرض می‌شود خواننده با تجزیه QR آشناست. برای یادگیری این تجزیه خواننده را به مرجع [۲۲] ارجاع می‌دهیم. در بخش دوم آموزش ماشین یادگیر نهایی سریع مبتنی بر تجزیه مورد بررسی قرار می‌گیرد و در بخش پایانی نیز با ماشین یادگیر نهایی دو لایه پنهان آشنا می‌شویم.

۲-۱ ELM افزایشی مبتنی بر تجزیه QR

در بخش‌های قبلی گفته شد که وزن‌های خروجی در ELM با استفاده از رابطه $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$ به دست می‌آید. در صورت افزودن یک نورون جدید به لایه پنهان، وزن‌های ورودی و وزن‌های خروجی شبکه عصبی یا باید دوباره محاسبه شوند و یا وزن‌های قبلی به روز رسانی شوند. در این بخش نحوه به روز رسانی وزن‌های خروجی شبکه ELM بر اساس وزن‌های قبلی با استفاده از تجزیه QR بیان می‌شود [۲۳]. روش‌های مختلفی برای تجزیه QR وجود دارد. با آزمایشاتی که انجام گرفته است روش گرام اشمیت برای توسعه الگوریتم افزایشی مناسب‌تر است.

روش گرام اشمیت، روشی برای متعامد سازی مجموعه‌ای از بردارها در فضای ضرب داخلی است. نکته قابل توجه در روش گرام اشمیت آن است که ماتریس باید رتبه کامل باشد یا به عبارتی ستون‌ها (سطرها) باید مستقل خطی باشند. تجزیه QR ماتریس $\mathbf{H}$ به صورت زیر محاسبه می‌شود.

$$\mathbf{H} = \mathbf{QR}$$

$\mathbf{Q}$ ماتریسی متعامد و $\mathbf{R}$ یک ماتریس بالا مثلثی است. فرض کنید

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_M], \mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_M]$$

و

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & ... & r_{1M} \\ & r_{22} & ... & r_{2M} \\ & & \ddots & \vdots \\ & & & r_{MM} \end{bmatrix}$$

باشند بنابراین داریم

$$[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_M] = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_M] \begin{bmatrix} r_{11} & r_{12} & ... & r_{1M} \\ & r_{22} & ... & r_{2M} \\ & & \ddots & \vdots \\ & & & r_{MM} \end{bmatrix}$$

با استفاده از رابطه بالا ستون‌های M ,..., ۱ ماتریس خروجی لایه پنهان $\mathbf{H}$ به صورت زیر محاسبه می‌شوند

$$\mathbf{h}_1 = \mathbf{q}_1 r_{11} \tag{۱-۲}$$

$$\mathbf{h}_2 = \mathbf{q}_1 r_{12} + \mathbf{q}_2 r_{22} \tag{۲-۲}$$

$$\vdots$$

$$\mathbf{h}_M = \mathbf{q}_1 r_{1M} + \mathbf{q}_2 r_{2M} + ... + \mathbf{q}_M r_{MM} \tag{۳-۲}$$

از آن جا که $\mathbf{Q}$ یک ماتریس متعامد است دو رابطه زیر برقرارند

$$\mathbf{Q}^T\mathbf{Q} = I \rightarrow \mathbf{q}_i^T\mathbf{q}_j = \begin{cases} ۰ & i \neq j \\ ۱ & i = j \end{cases}$$

یا به عبارتی دیگر

$$\mathbf{Q}^T = \mathbf{Q}^{-۱}$$

اکنون می‌توان درایه‌های ماتریس بالا مثلثی $\mathbf{R}$ را به صورت زیر به دست آورد.

از رابطه (۲-۱) ، $\mathbf{q}_۱ = \frac{\mathbf{h}_۱}{r_{۱۱}}$ است. نتیجه می‌شود

$$\mathbf{q}_۱^T\mathbf{q}_۱ = \frac{\mathbf{h}_۱^T}{r_{۱۱}}\frac{\mathbf{h}_۱}{r_{۱۱}} = ۱ \rightarrow \boxed{r_{۱۱} = \sqrt{\mathbf{h}_۱^T\mathbf{h}_۱}}$$

و رابطه (۲-۲) در $\mathbf{q}_۱^T$ ضرب می‌شود بنابراین خواهیم داشت

$$\mathbf{q}_۱^T\mathbf{h}_۲ = \mathbf{q}_۱^T\mathbf{q}_۱ r_{۱۲} + \mathbf{q}_۱^T\mathbf{q}_۲ r_{۲۲} \quad \mathbf{q}_۱^T\mathbf{q}_۲ = ۰ \quad \mathbf{q}_۱^T\mathbf{q}_۱ = ۱$$

$$\Rightarrow \boxed{r_{۱۲} = \mathbf{q}_۱^T\mathbf{h}_۲}$$

به طور کلی اگر $۱ \leq i < M$ باشد رابطه زیر برقرار است

$$\boxed{r_{iM} = \mathbf{q}_i^T\mathbf{h}_M \quad i \neq M}$$

اکنون طبق رابطه (۲-۳) ، به روابط زیر می‌رسیم

$$\mathbf{h}_M - (\mathbf{q}_۱ r_{۱M} + \mathbf{q}_۲ r_{۲M} + ... + \mathbf{q}_{M-۱} r_{M-۱,M}) = \mathbf{q}_M r_{MM}$$

$$\delta\mathbf{h}_M = \mathbf{q}_M r_{MM} \rightarrow \boxed{\mathbf{q}_M = \frac{\delta\mathbf{h}_M}{r_{MM}}}$$

و از طرفی طبق متعامد بودن ماتریس $\mathbf{Q}$ ،

$$\mathbf{q}_M^T\mathbf{q}_M = \frac{\delta\mathbf{h}_M^T}{r_{MM}}\frac{\delta\mathbf{h}_M}{r_{MM}} = ۱ \rightarrow \boxed{r_{MM} = \sqrt{\delta\mathbf{h}_M^T\delta\mathbf{h}_M}}$$

فرض کنید یک نورون می‌خواهد به M نورون موجود اضافه شود. در واقع ستون $\mathbf{h}_{M+۱}$ می‌خواهد به ماتریس

$\mathbf{H}_M$ (ماتریس $\mathbf{H}$ با M نورون پنهان) اضافه شود پس

$$\mathbf{H}_{M+1} = [\mathbf{H}_M | \mathbf{h}_{M+1}]$$

تجزیه QR ماتریس $\mathbf{H}_{M+1}$ به صورت رابطه زیر می‌باشد

$$\mathbf{H}_{M+1} = \mathbf{Q}_{M+1}\mathbf{R}_{M+1}$$

که در آن

$$\mathbf{Q}_{M+1} = [\mathbf{Q}_M | \mathbf{q}_{M+1}]$$

و

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|c} \mathbf{R}_M & \delta r_{M+1} \\ \hline \circ & r_{M+1,M+1} \end{array}\right] = \left[\begin{array}{cccc|c} r_{11} & r_{12} & \dots & r_{1M} & r_{1,M+1} \\ & r_{22} & \dots & r_{2M} & r_{2,M+1} \\ & & \ddots & \vdots & \vdots \\ & & & r_{MM} & r_{M,M+1} \\ & & \circ & & r_{M+1,M+1} \end{array}\right]$$

با افزودن نورون جدید رابطه (۲-۳) به صورت زیر نوشته می‌شود

$$\mathbf{h}_{M+1} = \mathbf{q}_1 r_{1,M+1} + \dots + \mathbf{q}_M r_{M,M+1} + \mathbf{q}_{M+1} r_{M+1,M+1}$$

$$\mathbf{h}_{M+1} - (\mathbf{q}_1 r_{1,M+1} + \dots + \mathbf{q}_M r_{M,M+1}) = \mathbf{q}_{M+1} r_{M+1,M+1}$$

$$\delta\mathbf{h}_{M+1} = \mathbf{q}_{M+1} r_{M+1,M+1} \rightarrow \boxed{\mathbf{q}_{M+1} = \frac{\delta\mathbf{h}_{M+1}}{r_{M+1,M+1}}}$$

$$\mathbf{q}_{M+1}^T\mathbf{q}_{M+1} = \frac{\delta\mathbf{h}_{M+1}^T}{r_{M+1,M+1}}\frac{\delta\mathbf{h}_{M+1}}{r_{M+1,M+1}} = 1 \rightarrow \boxed{r_{M+1,M+1} = \sqrt{\delta\mathbf{h}_{M+1}^T\delta\mathbf{h}_{M+1}}}$$

با توجه به تعریف ۱-۲-۱۰ می توان معکوس $\mathbf{H}$ را به صورت زیر نشان داد.

$$\mathbf{H}^\dagger = (\mathbf{QR})^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{-1} \quad , \quad \mathbf{Q}^{-1} = \mathbf{Q}^T \Longrightarrow \mathbf{H}^\dagger = \mathbf{R}^{-1}\mathbf{Q}^T$$

اساس این روش محاسبه $\mathbf{R}_{M+1}^{-1}$ از روی $\mathbf{R}_M^{-1}$ است که تعداد نورون‌های پنهان M را افزایش می‌دهد که در زیر روش کار بیان شده است . براساس لم ۱-۲-۳۱

$$\mathbf{R}_{M+1}^{-1} = \left[\begin{array}{c|c} \mathbf{R}_M & \delta\mathbf{r}_{M+1} \\ \hline \mathbf{0} & r_{M+1,M+1} \end{array}\right]^{-1} = \left[\begin{array}{c|c} \mathbf{R}_M^{-1} & -\mathbf{R}_M^{-1}\delta\mathbf{r}_{M+1} r_{M+1,M+1}^{-1} \\ \hline \mathbf{0} & r_{M+1,M+1}^{-1} \end{array}\right]$$

هدف یافتن وزن‌های خروجی $\tilde{\boldsymbol{\beta}}_{M+1}$ با استفاده از $\tilde{\boldsymbol{\beta}}_M$ و $\mathbf{R}_M^{-1}$ است. در شبکه ELM وزن‌های خروجی به صورت $\tilde{\boldsymbol{\beta}}_M = \mathbf{H}^{\dagger}\mathbf{T}$ محاسبه می‌شود. پس می‌توان وزن‌های خروجی $\tilde{\beta}_{M+1}$ را به صورت زیر محاسبه کرد که باعث کاهش پیچیدگی محاسباتی می‌شود.

$$\tilde{\boldsymbol{\beta}}_{M+1} = \mathbf{H}_{M+1}^{\dagger}\mathbf{T} = (\mathbf{Q}_{M+1}\mathbf{R}_{M+1})^{-1}\mathbf{T} = \mathbf{R}_{M+1}^{-1}\mathbf{Q}_{M+1}^{T}\mathbf{T}$$

$$\tilde{\boldsymbol{\beta}}_{M+1} = \left[\begin{array}{c|c} \mathbf{R}_M^{-1} & -\mathbf{R}_M^{-1}\delta\mathbf{r}_{M+1} r_{M+1,M+1}^{-1} \\ \hline \mathbf{0} & r_{M+1,M+1}^{-1} \end{array}\right] \left[\begin{array}{c} \mathbf{Q}_M^T \\ \hline \mathbf{q}_{M+1}^T \end{array}\right] \mathbf{T} = \left[\begin{array}{c} \tilde{\boldsymbol{\beta}}_M - \mathbf{R}_M^{-1}\delta\mathbf{r}_{M+1}\boldsymbol{\beta}_{M+1} \\ \hline \boldsymbol{\beta}_{M+1} \end{array}\right]$$

که $\boldsymbol{\beta}_{M+1} = \frac{\mathbf{q}_{M+1}^T\mathbf{T}}{r_{M+1,M+1}}$ و $\tilde{\boldsymbol{\beta}}_M = \mathbf{R}_M^{-1}\mathbf{Q}_M^T\mathbf{T}$ است. شیوه فوق در به روز رسانی ELM با نام QRI-ELM[1] مورد استفاده قرار می‌گیرد که در الگوریتم ۲-۱ (برنامه متلب صفحه ۱۰۲) بیان شده است.

[1] QR factorization based Incremental - Extreme Learning Machine (QRI-ELM)

الگوریتم ۲-۱ الگوریتم QRI-ELM [۲۳]

ورودی: مقدار مثبت کوچک ε و M_{max} (حداکثر تعداد نورون‌های پنهان)

خروجی: ماتریس وزن‌های خروجی به روز رسانی شده با افزودن هر نورون جدید به لایه پنهان ($\tilde{\boldsymbol{\beta}}$)

۱: تولید به طور تصادفی وزن‌های ورودی و بایاس اولین نورون پنهان $\{w_{1i}\}_{i=1}^{D}$، b_1

۲: محاسبه ماتریس خروجی لایه پنهان $\mathbf{H}_1(=\mathbf{h}_1)$

۳: محاسبه معکوس $\mathbf{R}_1$ ، $\mathbf{R}_1^{-1} = (\mathbf{h}_1^T\mathbf{h}_1)^{-\frac{1}{2}}$

۴: محاسبه $\mathbf{Q}_1(=\mathbf{q}_1) = \frac{\mathbf{h}_1}{r_{11}}$

۵: محاسبه وزن خروجی $\tilde{\boldsymbol{\beta}}(=\boldsymbol{\beta}_1) = \mathbf{R}_1^{-1}\mathbf{Q}_1^T\mathbf{T}$

۶: تا زمانی که $M = 1$ به M_{max} برسد و $E(\mathbf{H}_M) = \|\mathbf{T} - \mathbf{H}_M\boldsymbol{\beta}_M\| > \varepsilon$ باشد باید مراحل ۷ تا ۹ تکرار شوند

۷: یک نورون جدید اضافه می‌شود، وزن‌های ورودی و بایاس متناظر $\{w_{M+1,i}\}_{i=1}^{D}$، b_{M+1} تولید و $\mathbf{h}_{M+1}$ متناظر محاسبه می‌شوند

۸: به روز رسانی متغیرهای زیر به ترتیب :

$$\delta\mathbf{r}_{M+1} = \mathbf{Q}_M^T\mathbf{h}_{M+1}$$

$$\delta\mathbf{h}_{M+1} = \mathbf{h}_{M+1} - \mathbf{Q}_M\delta\mathbf{r}_{M+1}$$

$$r_{M+1,M+1}^{-1}(= \frac{1}{r_{M+1,M+1}}) = (\delta\mathbf{h}_{M+1}^T\delta\mathbf{h}_{M+1})^{-\frac{1}{2}}$$

$$\mathbf{q}_{M+1} = \delta\mathbf{h}_{M+1}r_{M+1,M+1}^{-1}$$

$$\boldsymbol{\beta}_{M+1} = r_{M+1,M+1}^{-1}\mathbf{q}_{M+1}^T\mathbf{T}$$

$$\tilde{\boldsymbol{\beta}}_{M+1} = \begin{bmatrix} \tilde{\boldsymbol{\beta}}_M - \mathbf{R}_M^{-1}\delta\mathbf{r}_{M+1}\boldsymbol{\beta}_{M+1} \\ \boldsymbol{\beta}_{M+1} \end{bmatrix}$$

$$\mathbf{R}_{M+1}^{-1} = \left[\begin{array}{c|c} \mathbf{R}_M^{-1} & -\mathbf{R}_M^{-1}\delta\mathbf{r}_{M+1}r_{M+1,M+1}^{-1} \\ 0 & r_{M+1,M+1}^{-1} \end{array}\right]$$

$$\mathbf{Q}_{M+1} = [\mathbf{Q}_M|\mathbf{q}_{M+1}]$$

۹: $M = M + 1$

۱۰: انتهای مراحل تکرار

۲-۲ ELM سریع مبتنی بر تجزیه

بسیاری از مسائل پیچیده طبقه‌بندی و رگرسیون ELM ، تعداد زیادی نورون پنهان نیاز دارند. با افزایش تعداد نورون‌های پنهان زمان آموزش الگوریتم ELM خیلی طول خواهد کشید. از این رو، راه‌حلی برای کاهش زمان آموزش الگوریتم نیاز شده است. الگوریتم ELM سریع مبتنی بر تجزیه (DFELM)[۱] که در این بخش به بررسی آن خواهیم پرداخت به طور چشمگیری این زمان را کاهش می‌دهد. الگوریتم DFELM با استفاده از لم ۱-۲-۳۱ برای معکوس ماتریس بزرگ تجزیه به کاهش زمان آموزش الگوریتم دست می‌یابد. در مقایسه با الگوریتم ELM الگوریتم DFELM با تعداد نورون پنهان زیاد دارای زمان آموزش سریعتری است [۱۵].

ابتدا ماتریس وزن‌های خروجی $\boldsymbol{\beta}$ به دو بخش تجزیه می‌شود که در آن $\boldsymbol{\beta}_1 \in \mathbb{R}^{M_1 \times K}$ ، $\boldsymbol{\beta}_2 \in \mathbb{R}^{M_2 \times K}$ و $M = M_1 + M_2$. ماتریس $\mathbf{H}$ نیز به دو بخش تجزیه می‌شود به طوری که $\mathbf{H}_1 \in \mathbb{R}^{N \times M_1}$ و $\mathbf{H}_2 \in \mathbb{R}^{N \times M_2}$.

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{bmatrix} = \mathbf{H}^{\dagger}\mathbf{T} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} = \left([\mathbf{H}_1 \quad \mathbf{H}_2]\right)^T[\mathbf{H}_1 \quad \mathbf{H}_2])^{-1}[\mathbf{H}_1 \quad \mathbf{H}_2]^T\mathbf{T}$$

$$= \left(\begin{bmatrix} \mathbf{H}_1^T \\ \mathbf{H}_2^T \end{bmatrix}[\mathbf{H}_1 \quad \mathbf{H}_2]\right)^{-1}\begin{bmatrix} \mathbf{H}_1^T \\ \mathbf{H}_2^T \end{bmatrix}\mathbf{T}$$

$$= \begin{bmatrix} \mathbf{H}_1^T\mathbf{H}_1 & \mathbf{H}_1^T\mathbf{H}_2 \\ \mathbf{H}_2^T\mathbf{H}_1 & \mathbf{H}_2^T\mathbf{H}_2 \end{bmatrix}^{-1}\begin{bmatrix} \mathbf{H}_1^T\mathbf{T} \\ \mathbf{H}_2^T\mathbf{T} \end{bmatrix}$$

بنابراین، با توجه به لم ۱-۲-۳۱

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{-1}\mathbf{H}_1^T(\mathbf{T} - \mathbf{H}_2\mathbf{F}) \\ \mathbf{F} - \mathbf{E}^{-1}\mathbf{H}_2^T\mathbf{H}_1\boldsymbol{\beta}_1 \end{bmatrix} \tag{۴-۲}$$

که در آن

$$\mathbf{E} = \mathbf{H}_2^T\mathbf{H}_2 \in \mathbb{R}^{M_2 \times M_2} \tag{۵-۲}$$

[۱] Decomposition based Fast Extreme Learning Machine(DFELM)

جدول ۲-۱: مقایسه RMSE الگوریتم‌های ELM و DFELM بر روی مجموعه داده خانه بوستون

نمونه‌ها	الگوریتم	تعداد نورون پنهان	زمان آموزش	دقت آموزشی	دقت تست
آموزشی/تست	ELM	۱۰۰	۰.۰۴۹۱	۰.۰۵۰۳	۰.۱۱۵۸
		۱۵۰	۰.۰۵۰۶	۰.۰۵۰۳	۰.۱۱۳۸
آموزشی/تست	DFELM	۱۰۰	۰.۰۲۱۶	۰.۰۵۰۶	۰.۱۱۳۱
		۱۵۰	۰.۰۲۵۳	۰.۰۵۰۳	۰.۱۱۱۲

$$\mathbf{F} = \mathbf{E}^{-1}\mathbf{H}_2^T\mathbf{T} \in \mathbb{R}^{M_2 \times K} \tag{۶-۲}$$

$$\mathbf{G} = \mathbf{I} - \mathbf{H}_2\mathbf{E}^{-1}\mathbf{H}_2^T \in \mathbb{R}^{N \times N} \tag{۷-۲}$$

$$\mathbf{V} = \mathbf{H}_1^T\mathbf{G}\mathbf{H}_1 \in \mathbb{R}^{M_1 \times M_1} \tag{۸-۲}$$

که $\mathbf{I}$ نشان‌دهنده یک ماتریس یگانی با ابعاد $N \times N$ است. چارچوب کلی DFELM در الگوریتم ۲-۲ (برنامه متلب در صفحه ۱۰۶) نشان داده شده است. یکی از معیارهای ارزیابی عملکرد در مسئله رگرسیون ، اندازه دقت برحسب جذر میانگین مربعات خطا (RMSE[1]) است. برای درک بهتر الگوریتم DFELM، مسئله رگرسیون بر روی مجموعه داده خانه بوستون[2] (مجموعه‌ای شامل ۵۰۶ نمونه با ۱۳ ویژگی است که ما ۴۰۶ نمونه را به عنوان نمونه آموزشی و ۱۰۰ نمونه را به عنوان نمونه تست درنظر گرفته‌ایم) با تعداد ۱۰۰ و ۱۵۰ نورون پنهان با برنامه متلب ELM و DFELM اجرا و با هم از نظر زمان آموزش (برحسب ثانیه) و RMSE آموزش و تست در جدول ۲-۱ مقایسه شده‌اند. از نتایج جدول نتیجه می‌شود که الگوریتم DFELM زمانی که تعداد نورون‌های پنهان به اندازه کافی بزرگ باشد نسبت به الگوریتم ELM بهتر عمل می‌کند.

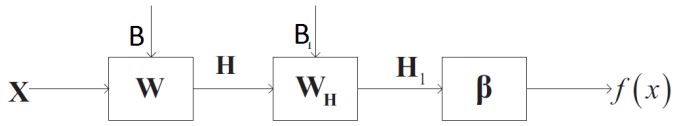
الگوریتم ۲-۲ الگوریتم DFELM [۱۵]

ورودی: مجموعه آموزشی $\{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \mathbb{R}^D, \mathbf{t}_i \in \mathbb{R}^K, i = 1, ..., N\}$، تعداد M نورون پنهان و تابع فعالیت نورون پنهان $g(\mathbf{w}_j^T\mathbf{x}_i + b_j), j = 1, ..., M$

خروجی: وزن‌های خروجی $\boldsymbol{\beta}$

۱: وزن‌های ورودی $\mathbf{w}_j$ و بایاس‌های b_j به طور تصادفی تولید می‌شوند.
۲: انتخاب M_1 و M_2 به طوری که در $M_1 + M_2 = M$ صدق کند.
۳: محاسبه E مبتنی بر رابطه (۲-۵)
۴: محاسبه F مبتنی بر رابطه (۲-۶)
۵: محاسبه G مبتنی بر رابطه (۲-۷)
۶: محاسبه V مبتنی بر رابطه (۲-۸)
۷: محاسبه $\boldsymbol{\beta}_1$ و $\boldsymbol{\beta}_2$ مبتنی بر رابطه (۲-۴)

[1] Root Mean Square Error(RMSE) $= \sqrt{\frac{\sum_{i=1}^{N}(\mathbf{t}_i - \mathbf{y}_i)^2}{N}}$ [2] Boston Housing



شکل ۲-۱: جریان کاری رویکرد TELM [۱]

۲-۳ ELM دو لایه پنهان

به دلیل انتساب تصادفی وزن‌های ورودی و بایاس نورون‌های پنهان، اکثراً میانگین دقت انواع ELM کم است لذا باید رویکردهای بهتری برای محاسبه پارامترهای لایه پنهان ارائه شوند. جریان کاری و ساختار شبکه ماشین یادگیر نهایی دولایه پنهان (TELM) که یک لایه پنهان به ELM تک لایه پنهان اضافه می‌کند و روشی جدید برای محاسبه پارامترهای لایه پنهان دوم به کار می‌گیرد، در شکل ۲-۱ و ۲-۲ نشان داده شده‌اند. TELM با استفاده از تعداد نورون‌های پنهان کمتر به عملکرد بهبود یافته‌ای دست می‌یابد [۱].

مجموعه‌ای از N نمونه آموزشی مجزای دلخواه $\{(\mathbf{x}_i, \mathbf{t}_i),\ \mathbf{x}_i \in \mathbb{R}^D,\ \mathbf{t}_i \in \mathbb{R}^K,\ i = 1, ..., N\}$ و $2M$ نورون پنهان (هریک از دولایه پنهان دارای M نورون پنهان) با تابع فعالیت $g(x)$ داده شده است، ابتدا به طور تصادفی ماتریس وزن اتصال دهنده بین لایه ورودی و اولین لایه پنهان $\mathbf{W}$ و بردار بایاس اولین لایه پنهان $\mathbf{B}$ مقداردهی اولیه می‌شوند. سپس ماتریس وزن بین دومین لایه پنهان و لایه خروجی $\boldsymbol{\beta}$ با استفاده از رابطه $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$ محاسبه می‌شود. مطابق با جریان کاری در شکل ۲-۱ نتیجه می‌شود

$$\mathbf{H}_1^T = g(\mathbf{W}_H\, \mathbf{H}^T + \mathbf{B}_1)$$

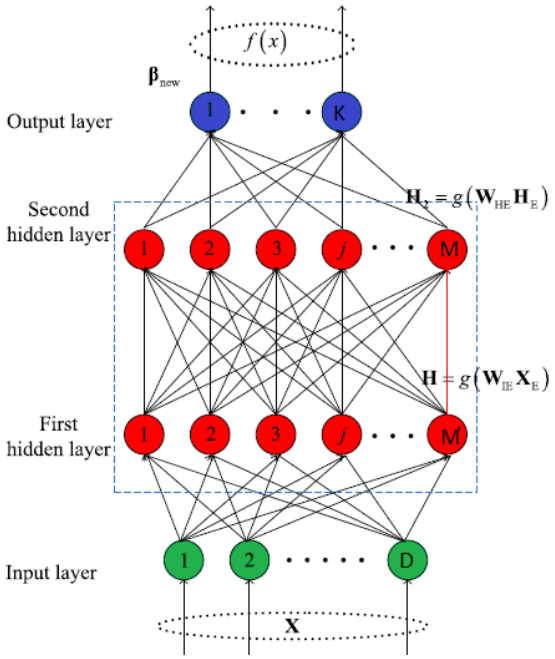
که $\mathbf{W}_H$ نشان‌دهنده ماتریس وزن بین لایه پنهان اول و لایه پنهان دوم و $\mathbf{B}_1$ بردار بایاس لایه پنهان دوم است. فرض می‌شود لایه‌های پنهان اول و دوم دارای تعداد نورون مشابهی هستند و بنابراین، $\mathbf{W}_H$ ماتریسی اسپارس است. علامت H نشان‌دهنده خروجی لایه پنهان اول با استفاده از N نمونه آموزشی است. ماتریس $\mathbf{H}_1$ نشان‌دهنده خروجی موردانتظار لایه پنهان دوم است. خروجی مورد انتظار لایه پنهان دوم می‌تواند با فرمول زیر محاسبه شود

$$\mathbf{H}_1 = \mathbf{T}\boldsymbol{\beta}^{\dagger}$$

که در آن $\boldsymbol{\beta}^{\dagger}$ معکوس تعمیم یافته مور-پنروز ماتریس $\boldsymbol{\beta}$ است. برای محاسبه $\boldsymbol{\beta}^{\dagger}$ از دو رابطه زیر استفاده می‌شود

الف) اگر $\boldsymbol{\beta}^T\boldsymbol{\beta}$ نامنفرد باشد، آن گاه $\boldsymbol{\beta}^{\dagger} = (\boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^T$.

ب) اگر $\boldsymbol{\beta}\boldsymbol{\beta}^T$ نامنفرد باشد، آن گاه $\boldsymbol{\beta}^{\dagger} = \boldsymbol{\beta}^T(\boldsymbol{\beta}\boldsymbol{\beta}^T)^{-1}$.



شکل ۲-۲: ساختار رویکرد TELM [۱]

سپس ماتریس افزونه[۱] $\mathbf{W}_{HE} = [\mathbf{B}_1 \quad \mathbf{W}_H]$ تعریف می‌شود و به صورت رابطه زیر به دست می‌آید

$$\mathbf{W}_{HE} = (g^{-1}(\mathbf{H}_1^T))^T(\mathbf{H}_E^T)^\dagger$$

که در آن $\mathbf{H}_E^\dagger$ معکوس تعمیم یافته مور-پنروز $\mathbf{H}_E = [1 \quad \mathbf{H}]$ است، " ۱" ماتریسی با اندازه $N \times 1$ است که عناصرش اسکالر ۱ هستند، $g^{-1}(x)$ معکوس تابع فعالیت $g(x)$ را نشان می‌دهد. خروجی واقعی لایه پنهان دوم به صورت زیر به دست می‌آید

$$\mathbf{H}_2^T = g(\mathbf{W}_{HE}\mathbf{H}_E^T)$$

ماتریس وزن $\boldsymbol{\beta}_{new}$ بین لایه پنهان دوم و لایه خروجی با رابطه زیر به دست می‌آید

$$\boldsymbol{\beta}_{new} = \mathbf{H}_2^\dagger\mathbf{T}$$

که در آن $\mathbf{H}_2^\dagger$ معکوس تعمیم یافته مور-پنروز $\mathbf{H}_2$ است. در نهایت خروجی TELM بعد آموزش به صورت رابطه زیر حاصل می‌شود

$$\mathbf{Y} = \mathbf{H}_2\boldsymbol{\beta}_{new}$$

[۱] augmented

الگوریتم TELM در الگوریتم ۲-۳ (برنامه متلب در صفحه ۱۰۹) نشان داده شده است.

الگوریتم ۲-۳ الگوریتم TELM [۱]

ورودی: N نمونه آموزشی $\{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \mathbb{R}^D, \mathbf{t}_i \in \mathbb{R}^K, i = ۱, ..., N\}$، و $۲M$ نورون پنهان با تابع فعالیت $g(x)$.

خروجی: ماتریس وزن‌های خروجی بین لایه پنهان دوم و لایه خروجی $\boldsymbol{\beta}_{new}$.

۱: تولید به طور تصادفی ماتریس وزن اتصال دهنده بین لایه ورودی و لایه پنهان اول $(\mathbf{W})$ و بردار بایاس لایه پنهان اول $(\mathbf{B})$ و برای سادگی، $\mathbf{W}_{IE} = [\mathbf{B} \quad \mathbf{W}]$ ، $\mathbf{X}_E = [۱ \quad \mathbf{X}]$ تعریف می‌شوند.

۲: محاسبه $\mathbf{H}^T = g(\mathbf{W}_{IE}\, \mathbf{X}_E^T)$

۳: به دست آوردن ماتریس وزن بین لایه پنهان دوم و لایه خروجی $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$

۴: محاسبه خروجی مورد انتظار لایه پنهان دوم $\mathbf{H}_۱ = \mathbf{T}\boldsymbol{\beta}^{\dagger}$

۵: تعیین پارامترهای لایه پنهان دوم $\mathbf{W}_{HE} = (g^{-۱}(\mathbf{H}_۱))^T(\mathbf{H}_E^T)^{\dagger}$ (ماتریس وزن اتصال دهنده لایه پنهان اول و دوم و بردار بایاس لایه پنهان دوم)

۶: به دست آوردن خروجی واقعی لایه پنهان دوم

$$\mathbf{H}_۲^T = g(\mathbf{W}_{HE}\, \mathbf{H}_E^T) = g([\mathbf{B}_۱ \quad \mathbf{W}_H]\,[۱ \quad \mathbf{H}]^T)$$

۷: محاسبه مجدد ماتریس وزن بین لایه پنهان دوم و لایه خروجی $\boldsymbol{\beta}_{new} = \mathbf{H}_۲^{\dagger}\mathbf{T}$

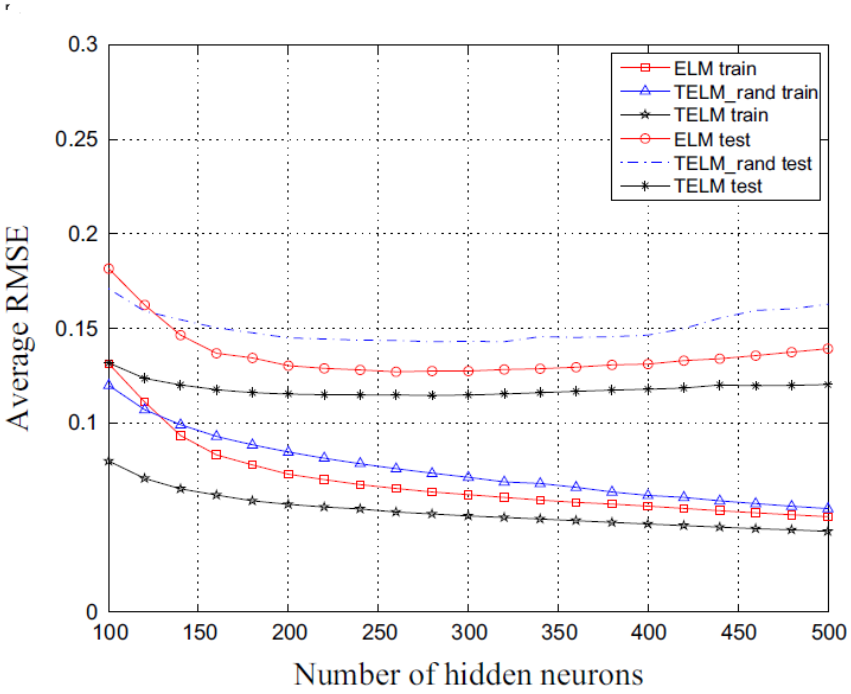
خروجی نهایی TELM

$$\mathbf{Y} = \mathbf{H}_۲\, \boldsymbol{\beta}_{new}$$

تابع بهینه سازی $f(x)$ به منظور تولید داده آموزشی و تست برای ارزیابی عملکرد الگوریتم‌های ELM ، TELM و TELM-rand که در آن پارامترهای لایه پنهان اول (وزن‌های ورودی بین لایه ورودی و لایه پنهان اول و بایاس‌ها) و پارامترهای لایه پنهان دوم (وزن‌های بین لایه پنهان اول و دوم و بایاس‌ها) به طور تصادفی متعامد انتخاب می‌شوند، استفاده می‌شود.

$$f(x) = ۲۰ - ۲۰e^{\sqrt{\sum_{i=۱}^{N} \frac{\mathbf{x}_i^۲}{D}}} - e^{\sum_{i=۱}^{N} \frac{cos(۲\pi \mathbf{x}_i)}{D}} + e$$

D بعد تابع است که برابر ۱۰ در نظر گرفته شده و $f(x)$ یک تابع غیرخطی چند نمایی پیچیده است. مسئله رگرسیون روی ۱۰۰۰ نمونه آموزشی و ۱۰۰۰ نمونه تست انجام می‌گیرد. ۱۰ ویژگی ورودی در فاصله $[۰, ۱]$ و ویژگی‌های خروجی در فاصله $[-۱, ۱]$ نرمال شده‌اند. معیار ارزیابی عملکرد RMSE در نظر گرفته شده است. در شکل ۲-۳ سه الگوریتم با هم مقایسه می‌شوند. از شکل می‌توان نتیجه گرفت که مقداردهی اولیه متعامد برای مسائل رگرسیون مناسب نیست و الگوریتم TELM تحت شرایطی که تعداد نورون‌های پنهان نسبتاً کم است خوب عمل می‌کند.



شکل ۲-۳: مقایسه RMSE برای الگوریتم‌های ELM، TELM و TELM-rand با استفاده از تابع چند نمایی پیچیده $f(x)$ بر روی نمونه‌های آموزشی و تست [۱]

فصل ۳

ماشین یادگیر نهایی نیمه ناظر و بدون ناظر

در این فصل ابتدا به بیان چارچوب تنظیم منیفلد می‌پردازیم. سپس ماشین یادگیر نهایی نیمه ناظر و در نهایت ماشین یادگیر نهایی بدون ناظر را به طور کامل بررسی خواهیم کرد.

در دو فصل قبل دیدیم که چگونه از ELM برای انجام کارهای یادگیری باناظر مانند رگرسیون و طبقه‌بندی استفاده می‌شود. در برخی موارد هم چون طبقه‌بندی متن، بازیابی اطلاعات و تشخیص خطا جمع‌آوری برچسب‌هایی برای یادگیری باناظر به طور کامل وقت‌گیر و گران است، در حالی که جمع‌آوری بسیاری از داده‌های بدون برچسب آسان و ارزان است. برای غلبه بر نقطه ضعف الگوریتم‌های یادگیری باناظر، که آن‌ها نمی‌توانند از داده بدون برچسب استفاده کنند، یادگیری نیمه ناظر[۲](SSL) که از هر دو داده بابرچسب و بدون برچسب استفاده می‌کند، پیشنهاد شده است. از آن جایی که SSL نیاز به تلاش کمتری برای جمع‌آوری داده بابرچسب دارد و می‌تواند دقت بالاتری را ارائه کند، از آن برای حوزه‌های مختلف استفاده می‌شود. در برخی موارد دیگر که هیچ داده بابرچسبی در دسترس نیست ممکن است مردم علاقه‌مند به کاوش در ساختار اصلی داده باشند. برای انجام این وظایف به طور گسترده روش‌های یادگیری بدون ناظر(USL) [۳] مانند خوشه‌بندی، کاهش ابعاد و یا نمایش داده‌ها استفاده می‌شوند [۲۴].

ELM برای مسائل یادگیری نیمه ناظر و بدون ناظر با معرفی چارچوب تنظیم منیفلد[۴] گسترش می‌یابد. تمام ELMs باناظر، نیمه ناظر و بدون ناظر در واقع می‌توانند در یک چارچوب قرار داده شوند، یعنی تمام الگوریتم‌ها از دو مرحله تشکیل شده‌اند: ۱) نگاشت ویژگی تصادفی ؛ و ۲) حل وزن‌های خروجی.

مرحله اول برای ساخت لایه پنهان با استفاده از نورون‌های پنهان به طور تصادفی تولید شده است. این مفهوم کلیدی در تئوری ELM است که آن را از بسیاری از روش‌های یادگیری ویژگی موجود متفاوت می‌سازد. نگاشت

---

[۲] Semi-Supervised Learning (SSL) [۳] UnSupervised Learning (USL) [۴] manifold regularization framework

ویژگی تصادفی، ELMs را برای تسریع یادگیری ویژگی غیرخطی قادر می‌سازد و مسئله بیش برازش[1] را کاهش می‌دهد. مرحله دوم حل وزن‌های بین لایه پنهان و لایه خروجی است و این جایی است که در آن تفاوت اصلی ELMs باناظر، نیمه ناظر و بدون ناظر نهفته است.

۳-۱ چارچوب تنظیم منیفلد

یادگیری نیمه ناظر روی دو فرضیه زیر ساخته شده است:

۱) هر دو داده بابرچسب $\mathbf{X}_l$ و داده بدون برچسب $\mathbf{X}_u$ از توزیع حاشیه‌ای یکسان $\mathcal{P}_x$ استخراج شده‌اند.

۲) فرض همواری[2] : اگر دو نقطه $\mathbf{x}_1$ و $\mathbf{x}_2$ نزدیک به هم باشند، آن گاه احتمالات شرطی $\mathcal{P}(\mathbf{y}|\mathbf{x}_1)$ و $\mathcal{P}(\mathbf{y}|\mathbf{x}_2)$ نیز باید نزدیک به هم باشند.

برای اعمال فرض دوم بر روی داده، چارچوب تنظیم منیفلد برای به حداقل رساندن تابع هزینه زیر پیشنهاد شده است

$$L_m = \frac{1}{2}\sum_{i,j} w_{ij} \left\|\mathcal{P}(\mathbf{y}|\mathbf{x}_i) - \mathcal{P}(\mathbf{y}|\mathbf{x}_j)\right\|^2 \qquad (۱-۳)$$

که w_{ij} مجاورت[3] بین دو نمونه $\mathbf{x}_i$ و $\mathbf{x}_j$ است. از آن جایی که اگر دو نمونه $\mathbf{x}_i$ و $\mathbf{x}_j$ نزدیک باشند (که $\mathbf{x}_i$ در میان k نزدیکترین همسایه $\mathbf{x}_j$ یا $\mathbf{x}_j$ در میان k نزدیکترین همسایه $\mathbf{x}_i$ است) بین آن‌ها فقط یک وزن غیرصفر قرار می‌گیرد بنابراین ماتریس مجاورت $\mathbf{W} = [w_{ij}]$ معمولاً اسپارس است. وزن‌های غیرصفر معمولاً با استفاده از تابع گوسی در رابطه (۱-۹) محاسبه می‌شوند و یا برای سادگی "ثابت یک" می‌باشند. به طور شهودی، زمانی که $\mathbf{x}$ دارای تغییر کوچکی است رابطه (۳-۱) برای احتمال شرطی $\mathcal{P}(\mathbf{y}|\mathbf{x})$ تغییر زیادی را موجب می‌شود. این مستلزم این است که $\mathcal{P}(\mathbf{y}|\mathbf{x})$ در طول دیسک[4] $\mathcal{P}(\mathbf{x})$ به طور هموار تغییر کند.

از آن جایی که محاسبه احتمال شرطی دشوار است، می‌توان رابطه (۳-۱) را با عبارت زیر تقریب زد

$$\hat{L}_m = \frac{1}{2}\sum_{i,j} w_{ij} \left\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\right\|^2 \qquad (۲-۳)$$

که در آن $\hat{\mathbf{y}}_i$ و $\hat{\mathbf{y}}_j$ با توجه به نمونه $\mathbf{x}_i$ و $\mathbf{x}_j$ به ترتیب پیش‌بینی می‌شوند. رابطه (۳-۲) طبق اولین خصوصیت

[1] over-fitting [2] smoothness assumption [3] similarity [4] geodesics

ماتریس لاپلاسین غیرنرمال به فرم ماتریسی زیر نوشته می‌شود

$$\hat{L}_m = Tr(\hat{\mathbf{Y}}^T \mathbf{L} \hat{\mathbf{Y}})$$

که $Tr(.)$ نشان‌دهنده اثر ماتریس و $\mathbf{L}$ به عنوان ماتریس لاپلاسین شناخته شده است.

در زمینه نیمه ناظر تعداد کمی داده بابرچسب و تعداد زیادی داده بدون برچسب وجود دارد. داده بابرچسب در مجموعه آموزشی با $\{\mathbf{X}_l, \mathbf{T}_l\} = \{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{l}$ و داده بدون برچسب با $\mathbf{X}_u = \{\mathbf{x}_i\}_{i=1}^{u}$ نشان داده می‌شوند که l و u به ترتیب تعداد داده‌های بابرچسب و بدون برچسب هستند.

۳-۲ ماشین یادگیر نهایی نیمه ناظر

ماشین یادگیر نهایی نیمه ناظر[۱](SS-ELM) زمانی که تعداد داده‌های بابرچسب کم است، به منظور بهبود دقت طبقه‌بند، چارچوب تنظیم منیفلد را برای مشارکت داده‌های بدون برچسب به کار می‌برد. مسئله SS-ELM به صورت زیر بیان می‌شود

$$\begin{aligned} \min_{\boldsymbol{\beta} \in \mathbb{R}^{M \times K}} & \frac{1}{2} \|\boldsymbol{\beta}\|^2 + \frac{1}{2} \sum_{i=1}^{l} C_i \|\mathbf{e}_i\|^2 + \frac{\lambda}{2} Tr(\mathbf{Y}^T \mathbf{L} \mathbf{Y}) \\ s.t \quad & \mathbf{h}(\mathbf{x}_i)\, \boldsymbol{\beta} = \mathbf{t}_i^T - \mathbf{e}_i^T, \quad i = 1, ..., l, \\ & \mathbf{y}_i = \mathbf{h}(\mathbf{x}_i)\, \boldsymbol{\beta}, \quad i = 1, ..., l+u \end{aligned} \tag{۳-۳}$$

که در آن $\mathbf{L} \in \mathbb{R}^{(l+u)\times(l+u)}$ ماتریس لاپلاسین ساخته شده از هر دو داده بابرچسب و بدون برچسب است و $\mathbf{Y} \in \mathbb{R}^{(l+u)\times K}$ ماتریس خروجی شبکه با i امین سطر آن برابر $\mathbf{y}(\mathbf{x}_i)$ است، λ یک پارامتر مصالحه[۲] است. فرض کنید $\mathbf{x}_i$ متعلق به کلاس $\mathbf{t}_i$ که دارای $N_{\mathbf{t}_i}$ نمونه آموزشی است،باشد آن گاه $\mathbf{e}_i$ با جریمه زیر مرتبط می‌شود

$$C_i = \frac{C_0}{N_{\mathbf{t}_i}}$$

[۱] Semi Supervised Extreme Learning Machine(SS-ELM) [۲] tradeoff

که در آن $C_۰$ یک پارامتر تعریف شده توسط کاربر در ELM سنتی است.
محدودیت‌ها در تابع هدف جایگزین و رابطه (۳-۳) به فرم ماتریسی بازنویسی می‌شود

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{M\times K}} \frac{۱}{۲}\|\boldsymbol{\beta}\|^۲ + \frac{۱}{۲}\left\|C^{\frac{۱}{۲}}(\mathbf{T}-\mathbf{H}\boldsymbol{\beta})\right\|^۲ + \frac{\lambda}{۲}Tr(\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{L}\mathbf{H}\boldsymbol{\beta})$$

که $\mathbf{T}\in\mathbb{R}^{(l+u)\times K}$ ماتریس خروجی مطلوب با اولین l سطر آن برابر با $\mathbf{T}_l$ و مابقی صفر است، $\mathbf{C}$ یک ماتریس قطری $(l+u)\times(l+u)$ با اولین عناصر قطر $[\mathbf{C}]_{ii}=\mathbf{C}_i,\ i=۱,...,l$ و بقیه برابر صفرند.
گرادیان تابع هدف برحسب $\boldsymbol{\beta}$ به صورت زیر است

$$\nabla L_{SS-ELM} = \boldsymbol{\beta} - \mathbf{H}^T\mathbf{C}(\mathbf{T}-\mathbf{H}\boldsymbol{\beta}) + \lambda\mathbf{H}^T\mathbf{L}\mathbf{H}\boldsymbol{\beta}$$

با قرار دادن گرادیان برابر با صفر، اگر **تعداد داده‌های آموزشی بیشتر از تعداد نورون‌های پنهان** باشد، جوابی برای SS-ELM به دست می‌آید

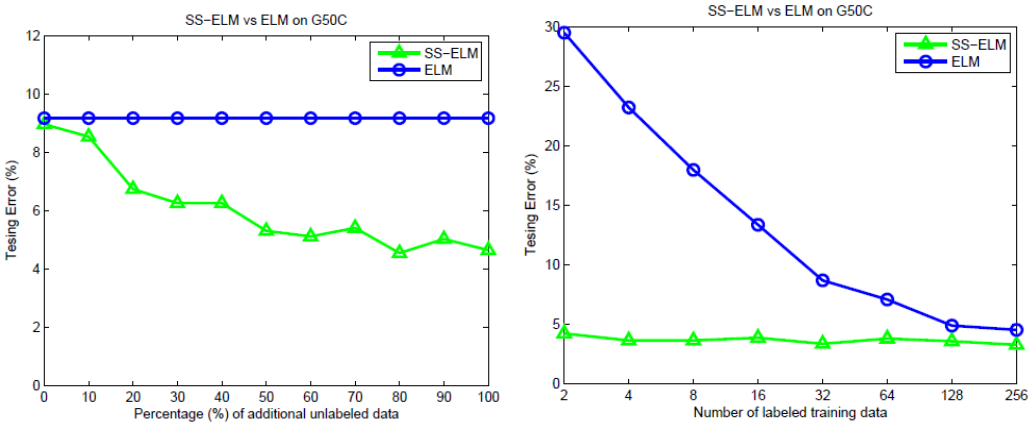
$$\boldsymbol{\beta}^* = (\mathbf{I}_M + \mathbf{H}^T\mathbf{C}\mathbf{H} + \lambda\mathbf{H}^T\mathbf{L}\mathbf{H})^{-۱}\mathbf{H}^T\mathbf{C}\mathbf{T} \qquad (۴-۳)$$

که $\mathbf{I}$ ماتریسی همانی با ابعاد $M\times M$ است. اگر **تعداد داده‌های آموزشی کمتر از تعداد نورون‌های پنهان** باشد جواب به صورت زیر محاسبه خواهد شد

$$\boldsymbol{\beta}^* = \mathbf{H}^T(\mathbf{I}_{l+u} + \mathbf{C}\mathbf{H}\mathbf{H}^T + \lambda\mathbf{L}\mathbf{H}\mathbf{H}^T)^{-۱}\mathbf{C}\mathbf{T} \qquad (۵-۳)$$

که $\mathbf{I}$ ماتریسی همانی با ابعاد $(l+u)\times(l+u)$ است.
V مجموعه Validation شامل مجموعه داده بابرچسب است و فقط برای انتخاب مدل ، یعنی، یافتن پارامترهای بهینه $C_۰$ و λ در الگوریتم SS-ELM استفاده می‌شود.
براساس مطالب بالا، الگوریتم SS-ELM در الگوریتم ۳-۱ (برنامه متلب در صفحه ۱۱۳) آورده شده است. در برنامه متلب از مجموعه داده G50C (مجموعه داده دو کلاسی شامل ۵۰ نمونه بابرچسب، ۳۱۴ نمونه بدون برچسب (U)[۱] ، ۵۰ نمونه اعتبارسنجی (V) ، ۱۳۶ نمونه تست (T) و ۵۰ ویژگی) استفاده می‌شود. الگوریتم یادگیری باناظر ELM و نیمه‌ناظر SS-ELM با تعداد ۱۰۰۰ نورون پنهان و انتخاب پارامترهای مصالحه $C_۰$ و λ از دنباله نمایی $\{۱۰^۶, ۱۰^{-۵}, ..., ۱۰^{-۶}\}$ با هم از نظر معیار خطای طبقه‌بند ($\pm$ انحراف معیار) بر روی مجموعه

[۱] Unlabeled(U)



(آ) عملکرد بهتر SS-ELM از ELM بر روی مجموعه داده G50C با تعداد داده بابرچسب کمتر

(ب) عملکرد بهتر SS-ELM از ELM بر روی مجموعه داده G50C با تعداد داده بدون برچسب بیشتر

شکل ۳-۱: عملکرد ELM و SS-ELM با افزایش تعداد داده بابرچسب و بدون برچسب [۲۴]

جدول ۳-۱: مقایسه خطای طبقه‌بند ELM و SS-ELM بر روی مجموعه داده G50C [۲۴]

الگوریتم	خطای طبقه‌بند U	خطای طبقه‌بند V	خطای طبقه‌بند T
ELM	۸٫۹۱ ± ۲٫۲۹	۸٫۲۰ ± ۳٫۰۵	۹٫۲۰ ± ۲٫۰۷
SS-ELM	۵٫۲۴ ± ۱٫۱۷	۴٫۰۷ ± ۰٫۹۵	۴٫۹۶ ± ۱٫۵۳

داده آموزشی بدون برچسب U ، مجموعه داده V و مجموعه داده تست T در جدول ۳-۱ مقایسه شده‌اند. همان طور که دیده می‌شود الگوریتم SS-ELM نسبت به الگوریتم ELM خطای کمتری دارد، به تعبیری دیگر دارای عملکرد بهتری است.

علاوه بر این عملکرد ELM و SS-ELM بر روی مجموعه داده G50C با تعداد متفاوت داده بابرچسب و تعداد متفاوت داده بدون برچسب در شکل ۳-۱(آ) و ۳-۱(ب) نشان داده می‌شود. زمانی که تعداد داده بابرچسب کمتر است، SS-ELM از ELM بهتر عمل می‌کند. هم چنین با تعداد بیشتر داده بدون برچسب SS-ELM خطای کمتری دارد.

۳-۳ ماشین یادگیر نهایی بدون ناظر

در ماشین یادگیر نهایی بدون ناظر[۱] (US-ELM) ، کل داده‌های آموزشی $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ بدون برچسبند و هدف یافتن ساختار اصلی داده است. هنگامی که داده بابرچسب وجود ندارد تابع هدف مسئله US-ELM به صورت

[۱] Un-Supervised Extreme Learning Machine(US-ELM)

الگوریتم ۳-۱ الگوریتم SS-ELM [۲۴]

ورودی: نمونه‌های بابرچسب $\{\mathbf{X}_l, \mathbf{T}_l\} = \{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{l}$ و نمونه‌های بدون برچسب $\mathbf{X}_u = \{\mathbf{x}_i\}_{i=1}^{u}$
خروجی: تابع نگاشت SS-ELM : $\mathbf{Y} : \mathbb{R}^D \to \mathbb{R}^K$
مرحله ۱) ساخت ماتریس لاپلاسین $\mathbf{L}$ با استفاده از $\mathbf{X}_l$ و $\mathbf{X}_u$
مرحله ۲) مقداردهی اولیه یک شبکه ELM با M نورون پنهان ، وزن‌های ورودی و بایاس‌های تصادفی و هم چنین محاسبه ماتریس خروجی نورون‌های پنهان $\mathbf{H} \in \mathbb{R}^{(l+u)\times M}$
مرحله ۳) انتخاب پارامتر مصالحه C و λ.
مرحله ۴) اگر $M \leqslant N$
محاسبه وزن‌های خروجی $\boldsymbol{\beta}$ با استفاده از رابطه (۳-۴)
در غیر این صورت
محاسبه وزن‌های خروجی $\boldsymbol{\beta}$ با استفاده از رابطه (۳-۵)
برگشتی الگوریتم: تابع نگاشت

$$\mathbf{y}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}$$

زیر است

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{M\times K}} \|\boldsymbol{\beta}\|^2 + \lambda Tr(\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{L}\mathbf{H}\boldsymbol{\beta}) \qquad (۳-۶)$$

مینیمم‌های مسئله (۳-۶) همیشه در $\boldsymbol{\beta} = 0$ است، برای جلوگیری از تباهیده شدن ($\boldsymbol{\beta} = 0$) باید محدودیت‌های اضافی معرفی شوند. فرمول US-ELM به صورت زیر ارائه می‌شود

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{M\times K}} \|\boldsymbol{\beta}\|^2 + \lambda Tr(\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{L}\mathbf{H}\boldsymbol{\beta})$$

$$s.t \quad (\mathbf{H}\boldsymbol{\beta})^T\mathbf{H}\boldsymbol{\beta} = \mathbf{I}_K \qquad (۳-۷)$$

مسئله (۳-۷) مجدد می‌تواند به صورت زیر نوشته شود

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{M\times K}, \boldsymbol{\beta}^T\mathbf{B}\boldsymbol{\beta}=\mathbf{I}_K} Tr(\boldsymbol{\beta}^T\mathbf{A}\boldsymbol{\beta}) \qquad (۳-۸)$$

$$\text{where} \quad \mathbf{A} = \mathbf{I}_M + \lambda\mathbf{H}^T\mathbf{L}\mathbf{H} \quad \text{and} \quad \mathbf{B} = \mathbf{H}^T\mathbf{H}$$

هر دو ماتریس $\mathbf{A}$ و $\mathbf{B}$ متقارن هستند

$$\mathbf{A}^T = \mathbf{A}$$

$$(\mathbf{I}_M + \lambda\mathbf{H}^T\mathbf{L}\mathbf{H})^T = \mathbf{I}_M^T + \lambda\mathbf{H}^T\mathbf{L}^T(\mathbf{H}^T)^T$$

با توجه به ویژگی دوم ماتریس لاپلاسین نتیجه می‌شود

$$(\mathbf{I}_M + \lambda \mathbf{H}^T \mathbf{L} \mathbf{H})^T = \mathbf{I}_M + \lambda \mathbf{H}^T \mathbf{L} \mathbf{H}$$

هم چنین

$$\mathbf{B}^T = \mathbf{B}$$

$$(\mathbf{H}^T \mathbf{H})^T = \mathbf{H}^T \mathbf{H}.$$

بنابراین با توجه به قضیه ریلی ریتز اگر **تعداد داده‌های بدون برچسب بیشتر از تعداد نورون‌های پنهان باشد** یک جواب بهینه برای مسئله (۳-۸)، با انتخاب $\boldsymbol{\beta}$ به عنوان ماتریسی که ستون‌هایش بردارهای ویژه (نرمال) متناظر با اولین K مقادیر ویژه کوچکتر مسئله مقدار ویژه تعمیم یافته زیر به دست می‌آید

$$(\mathbf{I}_M + \lambda \mathbf{H}^T \mathbf{L} \mathbf{H})\mathbf{v} = \gamma \mathbf{H}^T \mathbf{H} \mathbf{v} \qquad (۳-۹)$$

درنظر بگیرید $(\gamma_1 \leqslant \gamma_2 \leqslant ... \leqslant \gamma_{K+1})$ ، $\gamma_1, \gamma_2, ..., \gamma_{K+1}$ کوچکترین مقدار ویژه مسئله (۳-۹) و $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{K+1}$ بردارهای ویژه متناظر با آن‌ها باشند. آن گاه وزن‌های خروجی با استفاده از رابطه زیر به دست می‌آیند

$$\boldsymbol{\beta}^* = [\tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, ..., \tilde{\mathbf{v}}_{K+1}], \qquad (۳-۱۰)$$

که $\tilde{\mathbf{v}}_i$بردارهای ویژه نرمالند.

بنابراین با بردارهای ویژه نرمال مسئله (۳-۹) متناظر با کوچکترین K مقدار ویژه تعمیم یافته، یک جواب بهینه برای مسئله (۳-۷) به دست می‌آید. در الگوریتم نگاشت‌های ویژه لاپلاسین از آن جایی که بردار ویژه اول همیشه یک بردار با عناصر ثابت "یک" (متناظر با کوچکترین مقدار ویژه صفر) است، دور انداخته می‌شود. در الگوریتم US-ELM ، بردار ویژه اول مسئله (۳-۹) نیز منجر به تغییرات کوچک در نشاندن[1] می‌شود و برای نمایش داده‌ها مفید نیست. لذا این جواب بی اهمیت دور انداخته می‌شود.

اگر تعداد داده‌های بدون برچسب کمتر از تعداد نورون‌های پنهان باشد، یک جواب بهینه برای مسئله (۳-۸)، با انتخاب $\boldsymbol{\beta}$ به عنوان ماتریسی که ستون‌هایش بردارهای ویژه (نرمال) متناظر با اولین K مقادیر ویژه کوچکتر مسئله مقدار ویژه تعمیم یافته زیر به دست می‌آید

$$(\mathbf{I}_N + \lambda \mathbf{L} \mathbf{H} \mathbf{H}^T)u = \gamma \mathbf{H} \mathbf{H}^T \mathbf{u} \qquad (۳-۱۱)$$

درنظر بگیرید $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{K+1}$ بردارهای ویژه تعمیم یافته متناظر با $(K+1)$ کوچکترین مقدار ویژه مسئله

[1] embedding

جدول ۳-۲: مقایسه عملکرد الگوریتم‌های K-Means و US-ELM بر روی گل‌های زنبق

الگوریتم	دقت میانگین	بهترین دقت
K-Means	۷۹/۸۹۳۳	۸۹/۳۳۳۳
US-ELM	۹۵/۷۶۶۷	۹۸/۶۶۶۷

(۳-۱۱) باشند، آن گاه جواب نهایی به صورت زیر است

$$\boldsymbol{\beta}^* = \mathbf{H}^T[\tilde{\mathbf{u}}_2, \tilde{\mathbf{u}}_3, ..., \tilde{\mathbf{u}}_{K+1}], \qquad (۳-۱۲)$$

که $\tilde{\mathbf{u}}_i$بردارهای ویژه نرمالند.

اگر هدف خوشه‌بندی است پس می‌توان الگوریتم K-Means که در ادامه توضیح داده می‌شود را برای انجام خوشه‌بندی در نشاندن اتخاذ کرد. ماشین یادگیر نهایی بدون ناظر (US-ELM) در الگوریتم ۳-۲ (برنامه متلب در صفحه ۱۱۸) بیان می‌شود.

از US-ELM به منظور خوشه‌بندی مجموعه داده گل‌های زنبق استفاده می‌شود. این الگوریتم با الگوریتم K-Means مقایسه می‌شود، معیار مقایسه دو الگوریتم، دقت خوشه‌بندی [۱](ACC) ، طبق رابطه زیر به دست می‌آید

$$ACC = \frac{\sum_{i=1}^{N} \delta(\mathbf{t}_i, map(\mathbf{c}_i))}{N}$$

که در آن N تعداد نمونه‌های آموزشی بدون برچسب، $\mathbf{t}_i$ و $\mathbf{c}_i$ به ترتیب برچسب کلاس درست و برچسب خوشه پیش بینی شده نمونه $\mathbf{x}_i$ هستند،

$$\delta(\mathbf{t}_i, \mathbf{c}_i) = \begin{cases} 1 & \text{if} \quad \mathbf{t} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

، $map(.)$ تابع جایگشت بهینه‌ای است که با الگوریتم مجارستانی [۲] [۲۵] هر برچسب خوشه‌ای را به یک برچسب کلاس نگاشت می‌کند. تعداد نورون‌های پنهان برابر ۱۰۰۰ است. پارامتر λ از دنباله نمایی $\{10^{-4}, 10^{0}, ..., 10^{4}\}$ انتخاب می‌شود. دقت میانگین و بهترین دقت خوشه‌بندی این دو الگوریتم در جدول ۳-۲ بیان شده است. همان طور که دیده می‌شود که الگوریتم US-ELM دارای عملکرد بهتری است.

[۱] clustering ACCuracy (ACC) [۲] Hungarian

۳-۳-۱ خوشه‌بندی با روش K-Means

روش خوشه‌بندی K-Means روشی کلاسیک و ساده است و علی‌رغم سادگی یک روش پایه برای بسیاری از روش‌های خوشه‌بندی دیگر محسوب می‌شود. برای این روش انواع مختلفی بیان شده است که برای تعداد ثابتی از خوشه‌ها سعی در تخمین موارد زیر دارند [۱۸] :

۱) به دست آوردن نقاطی به عنوان مراکز خوشه‌ها، این نقاط در واقع همان میانگین نقاط متعلق به هر خوشه هستند.

۲) نسبت دادن هر نمونه داده به یک خوشه که آن داده کمترین فاصله تا مرکز آن خوشه را دارا باشد.

در نوع ساده‌ای از این روش ابتدا به تعداد خوشه‌های مورد نیاز نقاطی به صورت تصادفی انتخاب می‌شود. سپس داده‌ها با توجه با میزان نزدیکی (مجاورت) به یکی از این خوشه‌ها نسبت داده می‌شوند و بدین ترتیب خوشه‌های جدیدی حاصل می‌شود. با تکرار همین روال می‌توان در هر تکرار با میانگین‌گیری از داده‌ها مراکز جدیدی برای آن‌ها محاسبه کرد و مجدداً داده‌ها را به خوشه‌های جدید نسبت داد. این روند تا زمانی ادامه پیدا می‌کند که دیگر تغییری در داده‌ها حاصل نشود. تابع زیر به عنوان تابع هدف این روش مطرح است.

$$J = \sum_{j=۱}^{K} \sum_{i=۱}^{N} \|\mathbf{x}_i^{(j)} - c_j\|^۲$$

که $\|.\|$ معیار فاصله بین نقاط و c_j مرکز خوشه j ام است.

الگوریتم زیر الگوریتم پایه برای این روش محسوب می‌شود

۱) K نقطه به عنوان مراکز خوشه‌ها انتخاب می‌شوند.

۲) هر نمونه داده به خوشه‌ای که مرکز آن خوشه کمترین فاصله تا آن داده را داراست، نسبت داده می‌شود.

۳) پس تعلق تمام داده‌ها به یکی از خوشه‌ها برای هر خوشه یک نقطه جدید به عنوان مرکز محاسبه می‌شود. (میانگین نقاط متعلق به هر خوشه)

مراحل ۲ و ۳ تکرار می‌شوند تا زمانی که دیگر هیچ تغییری در مراکز خوشه‌ها حاصل نشود.

الگوریتم ۳-۲ الگوریتم US-ELM [۲۴]

ورودی: داده آموزشی $\mathbf{X} \in \mathbb{R}^{N \times D}$
خروجی: برای کار نشاندن
نشاندن در فضای K بعدی: $\mathbf{Y} \in \mathbb{R}^{N \times K}$
برای کار خوشه‌بندی
بردار برچسب شامل اندیس خوشه برای تمام نمونه‌های آموزشی: $\mathbf{c} \in \mathbb{N}^{N \times 1}$
مرحله ۱) ساخت ماتریس لاپلاسین $\mathbf{L}$ با استفاده از $\mathbf{X}$
مرحله ۲) مقداردهی اولیه شبکه ELM با M نورون پنهان، وزن‌های ورودی و بایاس‌های تصادفی و محاسبه ماتریس خروجی نورون‌های پنهان $\mathbf{H} \in \mathbb{R}^{N \times M}$
مرحله ۳) اگر $M \leqslant N$ باشد
یافتن بردارهای ویژه تعمیم یافته $\tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, ..., \tilde{\mathbf{v}}_{K+1}$ مطابق رابطه (۳-۱۰)
در غیر این صورت
یافتن بردارهای ویژه تعمیم یافته $\tilde{\mathbf{u}}_2, \tilde{\mathbf{u}}_3, ..., \tilde{\mathbf{u}}_{K+1}$ مطابق رابطه (۳-۱۲)
مرحله ۴) محاسبه ماتریس نشاندن $\mathbf{Y} = \mathbf{H}\boldsymbol{\beta}$
مرحله ۵) (فقط برای خوشه‌بندی): هر سطر $\mathbf{Y}$ یک نقطه است و N نقطه با استفاده از الگوریتم K-Means به K خوشه، خوشه‌بندی می‌شوند. در نظر بگیرید $\mathbf{c}$ بردار برچسب شامل اندیس خوشه برای تمام نمونه‌ها باشد
مقدار برگشتی: $\mathbf{Y}$ (نشاندن) یا $\mathbf{c}$ (خوشه‌بندی)

فصل ۴

نتایج محاسباتی

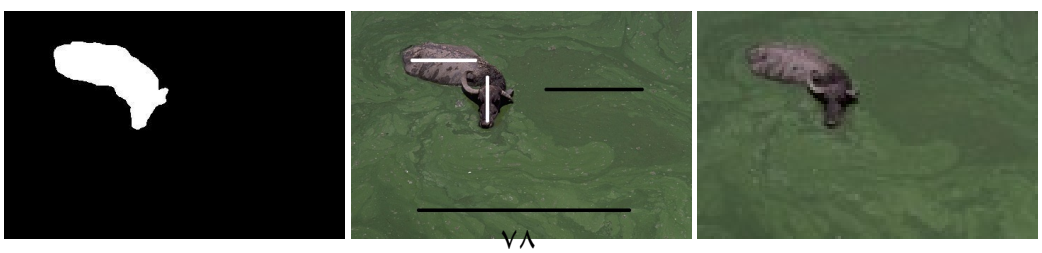
۴-۱ آزمایشات و نتایج تجربی

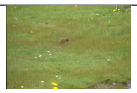
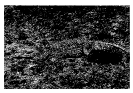
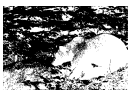
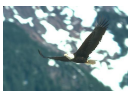
در این بخش عملکرد الگوریتم‌های MLP، ELM و TELM در کاربرد قطعه‌بندی تصاویر مورد مقایسه قرار خواهد گرفت. به این منظور از مجموعه دادگان BSD[۲] استفاده شده است. اکثر تصاویر این مجموعه با بیشتر از دو برچسب قطعه‌بندی شده‌اند اما در این جا فقط حالت دو کلاسی مورد بررسی قرار گرفته است، لذا از بین تصاویر موجود فقط ۱۵ تصویر که قطعه‌بندی دو کلاسی داشتند انتخاب شده‌اند. ستون دوم با عنوان «تصویر اصلی» در جداول ۴-۱ تا ۴-۳ - که شامل نتایج بصری هستند - این تصاویر را نشان می‌دهند. برای ایجاد داده‌های آموزشی از روش تعاملی[۳] استفاده شده است. در هر تصویر علامت‌هایی برای مشخص کردن قسمت‌هایی از نواحی زمینه و پیش زمینه توسط کاربر مشخص می‌شود پیکسل‌های آموزشی از ناحیه زمینه با رنگ سیاه و پیکسل‌های آموزشی از ناحیه پیش زمینه با رنگ سفید مشخص شده‌اند. در این تحقیق این نواحی با چند خط سیاه و سفید روی تصویر مشخص شده‌اند که این نواحی به عنوان داده‌های آموزشی به شبکه‌های مورد بحث داده شده است. یکی از این ۱۵ تصویر به همراه تصویر شامل نقاط آموزشی و جواب واقعی در شکل ۴-۱ آمده است.

[۲] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

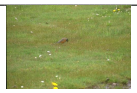
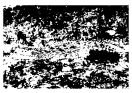
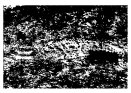
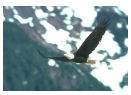
[۳] interactive

شکل ۴-۱: تصویر اصلی ، تصویر علامت گذاری شده با نقاط آموزشی (سیاه: زمینه، سفید: پیش‌زمینه) و جواب واقعی.

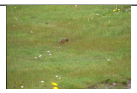
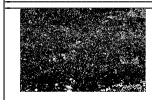
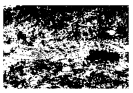
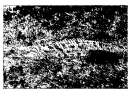
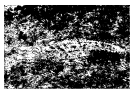
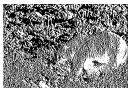
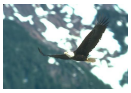
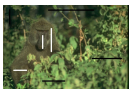


ردیف	تصویر اصلی	نقاط آموزشی	جواب واقعی	MLP	ELM	TELM
۱						
۲						
۳						
۴						
۵						
۶						
۷						
۸						
۹						
۱۰						
۱۱						
۱۲						
۱۳						
۱۴						
۱۵						

جدول ۴-۱: مقایسه خروجی قطعه‌بندی روش‌های مختلف با **۱۰ نورون** در لایه پنهان. خطوط سفید و سیاه در تصاویر ستون دوم نمایش‌دهنده نقاط آموزشی هستند. ستون سوم، قطعه‌بندی واقعی و سه ستون آخر خروجی سه شیوه موردنظر را نشان می‌دهند.

ردیف	تصویر اصلی	نقاط آموزشی	جواب واقعی	MLP	ELM	TELM
۱						
۲						
۳						
۴						
۵						
۶						
۷						
۸						
۹						
۱۰						
۱۱						
۱۲						
۱۳						
۱۴						
۱۵						

جدول ۴-۲: مقایسه خروجی قطعه‌بندی روش‌های مختلف با **۱۰۰ نورون** در لایه پنهان. خطوط سفید و سیاه در تصاویر ستون دوم نمایش‌دهنده نقاط آموزشی هستند. ستون سوم، قطعه‌بندی واقعی و سه ستون آخر خروجی سه شیوه موردنظر را نشان می‌دهند.

ردیف	تصویر اصلی	نقاط آموزشی	جواب واقعی	MLP	ELM	TELM
۱						
۲						
۳						
۴						
۵						
۶						
۷						
۸						
۹						
۱۰						
۱۱						
۱۲						
۱۳						
۱۴						
۱۵						

جدول ۴-۳: مقایسه خروجی قطعه‌بندی روش‌های مختلف با **۵۰۰ نورون** در لایه پنهان. خطوط سفید و سیاه در تصاویر ستون دوم نمایش‌دهنده نقاط آموزشی هستند. ستون سوم، قطعه‌بندی واقعی و سه ستون آخر خروجی سه شیوه موردنظر را نشان می‌دهند.

ستون سوم از جداول مقایسات بصری نمایش دهنده «نقاط آموزشی» در تصویر مربوطه است. تصاویر اصلی دارای ابعاد ۳۲۱ × ۴۸۱ هستند و این تصاویر هم‌اندازه با تصاویر اصلی هستند. به صورت میانگین حدود ۳ درصد پیکسل‌ها به عنوان داده آموزشی درنظر گرفته شده‌اند. تصاویر موجود در این پایگاه داده توسط افراد مختلف قطعه‌بندی شده و به عنوان داده‌های درست در تحقیقات مورد استفاده قرار گرفته‌اند و نتایجشان تحت عنوان «جواب واقعی» [۱] در ستون چهارم از جدول ۴-۱ آورده شده است.

به منظور حفظ هماهنگی در مقایسات هر ۳ شبکه با شرایط یکسان (تعداد نورون‌های ورودی، پنهان و خروجی یکسان) مورد مقایسه قرار گرفته‌اند. همه مقایسات در محیط محاسباتی متلب R۲۰۱۲a بر روی سیستم با واحد پردازش مرکزی Intel(R) Core(TM) i3-4010U CPU 1.70GHz و حافظه داخلی ۴GB اجرا شده‌اند. متناظر با هر پیکسل، مقادیر شدت رنگ RGB تمامی پیکسل‌ها در یک همسایگی ۳ × ۳ آن، واریانس همسایگی و گرادیان‌های افقی و عمودی آن به عنوان ویژگی‌های هر پیکسل مورد استفاده قرار گرفته است.

آزمایشات روی شبکه‌های عصبی مورد بحث با تعداد ۱۰، ۱۰۰ و ۵۰۰ نورون در لایه پنهان انجام شده است. به منظور مقایسه نتایج از مقایسات بصری و کمی استفاده شده است. برای هر یک از ۱۵ تصاویر الگوریتم‌های MLP،ELM و TELM ۱۰ مرتبه اجرا شده‌اند و تصویر متناظر با اجرای دارای بیشترین «صحت» در جداول ۴-۱ تا ۴-۳ آمده است. مقادیر ذکر شده در جداول مربوط به معیارهای مختلف، میانگین ۱۰ مرتبه اجراست.

معیارهای ارزیابی الگوریتم‌های طبقه‌بندی

در این بخش معیارهای ارزیابی عملکرد الگوریتم‌های طبقه‌بندی معرفی می‌شوند. کلیه معیارهای مورد بررسی در این بخش را می‌توان برای مجموعه نمونه‌های تست در مرحله ارزیابی محاسبه نمود. در این بخش فرض بر آن است که الگوریتم طبقه‌بندی مورد ارزیابی برای کلیه نمونه‌های مورد بررسی یکی از انواع برچسب‌های ممکن (طبقه‌های موجود) را به عنوان خروجی خود پیشنهاد می‌کند. برای سادگی، معیارهای طبقه‌بندی را برای یک مسئله با دو طبقه ارائه خواهیم نمود. ماتریس **درهم‌ریختگی** [۲] چگونگی عملکرد الگوریتم طبقه‌بندی را با توجه به مجموعه داده ورودی به تفکیک انواع طبقه‌های مسئله طبقه‌بندی نشان می‌دهد [۵] (جدول ۴-۱۶). در این ماتریس مفاهیم TN، FP، FN و TP به شرح ذیل می‌باشد

TN: این مقدار بیانگر تعداد نمونه‌هایی است که طبقه واقعی آن‌ها منفی بوده و الگوریتم طبقه‌بندی نیز طبقه آن‌ها را به درستی منفی تشخیص داده است.

FP: این مقدار بیانگر تعداد نمونه‌هایی است که طبقه واقعی آن‌ها منفی بوده و الگوریتم طبقه‌بندی طبقه آن‌ها را به اشتباه مثبت تشخیص داده است.

---

[۱] Ground Truth [۲] Confusion Matrix

No.	MLP	ELM	TELM
۱	۰/۹۸۲	۰/۹۹۴	۰/۹۸۷
۲	۰/۹۷۳	۰/۸۲۱	۰/۸۱۰
۳	۰/۷۰۰	۰/۷۶۲	۰/۷۸۳
۴	۰/۷۹۱	۰/۷۲۷	۰/۷۴۵
۵	۰/۹۱۱	۰/۹۰۷	۰/۸۷۷
۶	۰/۹۸۳	۰/۹۶۷	۰/۹۵۳
۷	۰/۹۱۹	۰/۸۳۵	۰/۸۰۶
۸	۰/۸۳۷	۰/۷۶۷	۰/۷۶۶
۹	۰/۹۹۲	۰/۹۳۴	۰/۸۹۷
۱۰	۰/۹۳۶	۰/۸۹۵	۰/۸۶۳
۱۱	۰/۸۲۰	۰/۷۹۱	۰/۷۵۶
۱۲	۰/۸۵۷	۰/۷۸۳	۰/۷۷۹
۱۳	۰/۸۱۶	۰/۸۴۴	۰/۸۶۱
۱۴	۰/۹۸۵	۰/۹۵۵	۰/۹۴۸
۱۵	۰/۸۸۰	۰/۸۴۳	۰/۸۳۵
Avg	۰/۸۹۲	۰/۸۵۵	۰/۸۴۴

جدول ۴-۴: معیار «صحت» سه شبکه، با ۱۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰/۴۴۸	۰/۰۴۶	۰/۱۸۷
۲	۰/۹۵۲	۰/۹۴۳	۰/۸۹۰
۳	۰/۷۳۲	۰/۱۹۳	۰/۱۳۸
۴	۰/۹۱۲	۰/۸۶۷	۰/۸۱۹
۵	۰/۹۱۱	۰/۸۶۵	۰/۸۴۳
۶	۰/۸۷۷	۰/۷۳۰	۰/۶۸۶
۷	۰/۶۸۱	۰/۵۵۰	۰/۴۲۶
۸	۰/۴۰۴	۰/۴۴۰	۰/۳۹۴
۹	۰/۹۷۰	۰/۸۷۹	۰/۸۴۹
۱۰	۰/۸۴۵	۰/۶۲۴	۰/۵۷۹
۱۱	۰/۷۷۷	۰/۶۹۵	۰/۷۳۶
۱۲	۰/۶۰۸	۰/۱۴۲	۰/۰۴۹
۱۳	۰/۴۹۸	۰/۰۷۴	۰/۰۲۲
۱۴	۰/۸۶۷	۰/۲۷۰	۰/۲۹۳
۱۵	۰/۵۱۰	۰/۲۶۶	۰/۲۵۸
Avg	۰/۷۳۳	۰/۵۰۶	۰/۴۷۸

جدول ۴-۵: معیار «حساسیت» سه شبکه، با ۱۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰/۰۱۶	۰/۰۰۲	۰/۰۰۹
۲	۰/۰۲۲	۰/۲۰۹	۰/۲۰۹
۳	۰/۳۰۶	۰/۱۳۸	۰/۱۰۴
۴	۰/۲۴۳	۰/۳۱۴	۰/۲۷۶
۵	۰/۰۸۹	۰/۰۷۹	۰/۱۱۲
۶	۰/۰۰۸	۰/۰۱۴	۰/۰۲۵
۷	۰/۰۴۱	۰/۱۱۷	۰/۱۲۹
۸	۰/۰۴۴	۰/۱۴۳	۰/۱۳۳
۹	۰/۰۰۷	۰/۰۶۳	۰/۱۰۰
۱۰	۰/۰۴۷	۰/۰۵۴	۰/۰۸۳
۱۱	۰/۱۴۸	۰/۱۳۶	۰/۲۲۹
۱۲	۰/۰۷۴	۰/۰۴۱	۰/۰۲۱
۱۳	۰/۱۳۶	۰/۰۳۹	۰/۰۱۲
۱۴	۰/۰۱۰	۰/۰۱۷	۰/۰۲۵
۱۵	۰/۰۴۹	۰/۰۴۵	۰/۰۵۳
Avg	۰/۰۸۳	۰/۰۹۴	۰/۱۰۱

جدول ۴-۶: معیار «نرخ مثبت کاذب» سه شبکه، با ۱۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰/۰۰۳	۰/۰۰۴	۰/۰۰۴
۲	۰/۰۱۲	۰/۰۱۴	۰/۰۲۷
۳	۰/۰۴۷	۰/۱۴۰	۰/۱۵۰
۴	۰/۰۲۵	۰/۰۳۸	۰/۰۵۲
۵	۰/۰۲۷	۰/۰۴۱	۰/۰۴۸
۶	۰/۰۱۰	۰/۰۲۲	۰/۰۲۶
۷	۰/۰۵۴	۰/۰۷۶	۰/۰۹۷
۸	۰/۱۶۳	۰/۱۵۳	۰/۱۶۶
۹	۰/۰۰۲	۰/۰۰۸	۰/۰۱۰
۱۰	۰/۰۲۹	۰/۰۷۲	۰/۰۸۰
۱۱	۰/۱۶۹	۰/۲۳۰	۰/۲۰۰
۱۲	۰/۱۰۸	۰/۲۳۶	۰/۲۶۱
۱۳	۰/۰۷۶	۰/۱۴۱	۰/۱۴۹
۱۴	۰/۰۰۶	۰/۰۳۰	۰/۰۲۹
۱۵	۰/۰۹۵	۰/۱۴۳	۰/۱۴۴
Avg	۰/۰۵۵	۰/۰۹۰	۰/۰۹۶

جدول ۴-۷: معیار «نرخ منفی کاذب» سه شبکه، با ۱۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۹۸۳	۰٫۹۷۴	۰٫۹۷۸
۲	۰٫۹۷۲	۰٫۸۸۸	۰٫۹۰۴
۳	۰٫۷۰۳	۰٫۷۵۱	۰٫۷۲۳
۴	۰٫۷۷۹	۰٫۶۹۰	۰٫۷۶۲
۵	۰٫۹۵۳	۰٫۹۳۶	۰٫۹۳۹
۶	۰٫۹۸۳	۰٫۹۷۸	۰٫۹۷۶
۷	۰٫۹۳۴	۰٫۸۷۱	۰٫۹۰۹
۸	۰٫۸۲۹	۰٫۷۶۴	۰٫۷۷۴
۹	۰٫۹۸۶	۰٫۹۷۷	۰٫۹۷۲
۱۰	۰٫۹۲۹	۰٫۹۱۱	۰٫۸۴۷
۱۱	۰٫۸۴۶	۰٫۸۳۸	۰٫۷۷۲
۱۲	۰٫۸۵۸	۰٫۸۵۵	۰٫۸۴۷
۱۳	۰٫۸۱۷	۰٫۷۹۰	۰٫۷۶۰
۱۴	۰٫۹۸۲	۰٫۹۷۷	۰٫۹۷۱
۱۵	۰٫۸۸۴	۰٫۸۶۳	۰٫۸۵۱
Avg	۰٫۸۹۶	۰٫۸۷۱	۰٫۸۶۶

جدول ۴-۸: معیار «صحت» سه شبکه، با ۱۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۴۰۳	۰٫۵۰۸	۰٫۴۶۱
۲	۰٫۹۶۷	۰٫۹۸۷	۰٫۹۸۴
۳	۰٫۶۰۹	۰٫۵۵۴	۰٫۵۷۴
۴	۰٫۹۰۶	۰٫۸۲۳	۰٫۷۸۹
۵	۰٫۹۱۵	۰٫۸۹۶	۰٫۹۲۹
۶	۰٫۸۸۸	۰٫۸۶۷	۰٫۸۶۷
۷	۰٫۸۵۱	۰٫۹۴۱	۰٫۹۰۶
۸	۰٫۳۴۱	۰٫۵۰۶	۰٫۴۷۷
۹	۰٫۹۷۳	۰٫۹۸۹	۰٫۹۸۶
۱۰	۰٫۷۹۸	۰٫۸۱۳	۰٫۸۹۹
۱۱	۰٫۸۵۱	۰٫۷۰۰	۰٫۸۹۹
۱۲	۰٫۶۳۱	۰٫۵۴۷	۰٫۵۲۰
۱۳	۰٫۵۱۰	۰٫۶۲۵	۰٫۶۳۴
۱۴	۰٫۷۶۶	۰٫۸۶۰	۰٫۹۱۴
۱۵	۰٫۵۸۳	۰٫۵۵۳	۰٫۵۵۲
Avg	۰٫۷۳۳	۰٫۷۴۵	۰٫۷۵۹

جدول ۴-۹: معیار «حساسیت» سه شبکه، با ۱۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۰۱۵	۰٫۰۲۴	۰٫۰۱۹
۲	۰٫۰۲۷	۰٫۱۳۶	۰٫۱۱۶
۳	۰٫۲۸۱	۰٫۲۱۵	۰٫۲۵۱
۴	۰٫۲۵۷	۰٫۳۴۸	۰٫۲۴۵
۵	۰٫۰۳۵	۰٫۰۵۲	۰٫۰۵۸
۶	۰٫۰۰۹	۰٫۰۱۳	۰٫۰۱۵
۷	۰٫۰۵۲	۰٫۱۴۱	۰٫۰۹۰
۸	۰٫۰۳۷	۰٫۱۶۵	۰٫۱۴۵
۹	۰٫۰۱۳	۰٫۰۲۴	۰٫۰۲۹
۱۰	۰٫۰۴۶	۰٫۰۷۰	۰٫۱۶۳
۱۱	۰٫۱۵۷	۰٫۰۵۸	۰٫۳۲۴
۱۲	۰٫۰۸۰	۰٫۰۶۱	۰٫۰۶۴
۱۳	۰٫۱۳۶	۰٫۱۸۵	۰٫۲۲۱
۱۴	۰٫۰۰۹	۰٫۰۱۹	۰٫۰۲۶
۱۵	۰٫۰۵۸	۰٫۰۷۷	۰٫۰۹۱
Avg	۰٫۰۸۱	۰٫۱۰۶	۰٫۱۲۴

جدول ۴-۱۰: معیار «نرخ مثبت کاذب» سه شبکه، با ۱۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۰۰۳	۰٫۰۰۲	۰٫۰۰۲
۲	۰٫۰۰۸	۰٫۰۰۳	۰٫۰۰۴
۳	۰٫۰۶۸	۰٫۰۷۸	۰٫۰۷۴
۴	۰٫۰۲۷	۰٫۰۵۱	۰٫۰۶۱
۵	۰٫۰۲۶	۰٫۰۳۲	۰٫۰۲۲
۶	۰٫۰۰۹	۰٫۰۱۱	۰٫۰۱۱
۷	۰٫۰۲۵	۰٫۰۱۰	۰٫۰۱۶
۸	۰٫۱۸۰	۰٫۱۳۵	۰٫۱۴۳
۹	۰٫۰۰۲	۰٫۰۰۱	۰٫۰۰۱
۱۰	۰٫۰۳۸	۰٫۰۳۶	۰٫۰۱۹
۱۱	۰٫۱۱۳	۰٫۲۲۷	۰٫۰۷۶
۱۲	۰٫۱۰۱	۰٫۱۲۴	۰٫۱۳۲
۱۳	۰٫۰۷۵	۰٫۰۵۷	۰٫۰۵۶
۱۴	۰٫۰۱۰	۰٫۰۰۶	۰٫۰۰۴
۱۵	۰٫۰۸۱	۰٫۰۸۷	۰٫۰۸۷
Avg	۰٫۰۵۱	۰٫۰۵۷	۰٫۰۴۷

جدول ۴-۱۱: معیار «نرخ منفی کاذب» سه شبکه، با ۱۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۹۸۳	۰٫۷۹۹	۰٫۹۳۳
۲	۰٫۹۷۳	۰٫۶۸۱	۰٫۷۶۴
۳	۰٫۶۸۵	۰٫۷۲۴	۰٫۷۴۵
۴	۰٫۷۷۶	۰٫۵۸۲	۰٫۵۳۲
۵	۰٫۹۵۶	۰٫۷۶۲	۰٫۸۴۷
۶	۰٫۹۸۱	۰٫۶۴۷	۰٫۹۴۴
۷	۰٫۹۳۵	۰٫۶۶۹	۰٫۷۸۳
۸	۰٫۸۳۳	۰٫۶۲۵	۰٫۶۷۷
۹	۰٫۹۸۷	۰٫۸۶۱	۰٫۹۴۲
۱۰	۰٫۹۲۸	۰٫۷۹۵	۰٫۸۵۷
۱۱	۰٫۸۴۴	۰٫۵۴۶	۰٫۷۱۵
۱۲	۰٫۸۶۲	۰٫۶۸۵	۰٫۷۷۵
۱۳	۰٫۸۱۱	۰٫۶۶۳	۰٫۷۴۱
۱۴	۰٫۹۸۳	۰٫۹۲۱	۰٫۹۶۶
۱۵	۰٫۸۸۴	۰٫۷۲۰	۰٫۷۹۲
Avg	۰٫۸۹۵	۰٫۷۱۲	۰٫۸۰۱

جدول ۴-۱۲: معیار «صحت» سه شبکه، با ۵۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۳۴۱	۰٫۶۳۷	۰٫۴۱۸
۲	۰٫۹۶۵	۰٫۸۴۸	۰٫۸۲۹
۳	۰٫۶۶۳	۰٫۵۸۲	۰٫۶۲۲
۴	۰٫۸۱۹	۰٫۷۰۲	۰٫۶۳۸
۵	۰٫۹۲۲	۰٫۶۶۹	۰٫۷۳۰
۶	۰٫۸۸۸	۰٫۷۴۱	۰٫۷۷۴
۷	۰٫۸۴۹	۰٫۷۰۶	۰٫۸۳۸
۸	۰٫۳۶۲	۰٫۵۲۷	۰٫۴۷۵
۹	۰٫۹۷۳	۰٫۸۰۱	۰٫۳۴۲
۱۰	۰٫۷۹۲	۰٫۵۹۹	۰٫۶۷۴
۱۱	۰٫۸۳۴	۰٫۵۳۵	۰٫۵۶۱
۱۲	۰٫۵۹۴	۰٫۵۲۱	۰٫۵۱۵
۱۳	۰٫۵۷۰	۰٫۶۲۳	۰٫۶۶۶
۱۴	۰٫۸۴۷	۰٫۷۱۰	۰٫۷۲۱
۱۵	۰٫۵۹۱	۰٫۵۵۷	۰٫۵۳۲
Avg	۰٫۷۳۴	۰٫۶۵۰	۰٫۶۲۳

جدول ۴-۱۳: معیار «حساسیت» سه شبکه، با ۵۰۰ نورون نهان

No.	MLP	ELM	TELM
۱	۰٫۰۱۴	۰٫۲۰۰	۰٫۰۶۴
۲	۰٫۰۲۵	۰٫۳۶۰	۰٫۲۵۲
۳	۰٫۳۱۱	۰٫۲۵۱	۰٫۲۳۳
۴	۰٫۲۳۶	۰٫۴۵۲	۰٫۴۹۹
۵	۰٫۰۳۳	۰٫۲۰۹	۰٫۱۱۷
۶	۰٫۰۱۱	۰٫۳۶۱	۰٫۰۴۲
۷	۰٫۰۵۰	۰٫۳۳۸	۰٫۲۲۷
۸	۰٫۰۳۹	۰٫۳۴۹	۰٫۲۶۸
۹	۰٫۰۱۲	۰٫۱۳۵	۰٫۰۱۹
۱۰	۰٫۰۴۶	۰٫۱۶۸	۰٫۱۰۸
۱۱	۰٫۱۴۹	۰٫۴۴۵	۰٫۱۶۸
۱۲	۰٫۰۶۵	۰٫۲۷۰	۰٫۱۵۴
۱۳	۰٫۱۵۳	۰٫۳۳۱	۰٫۲۴۸
۱۴	۰٫۰۱۱	۰٫۰۷۰	۰٫۰۲۴
۱۵	۰٫۰۵۹	۰٫۲۴۹	۰٫۱۵۷
Avg	۰٫۰۸۱	۰٫۲۷۹	۰٫۱۷۲

جدول ۴-۱۴: معیار «نرخ مثبت کاذب» سه شبکه، با ۵۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۰٫۰۰۳	۰٫۰۰۲	۰٫۰۰۳
۲	۰٫۰۰۹	۰٫۰۳۷	۰٫۰۴۲
۳	۰٫۰۵۹	۰٫۰۷۳	۰٫۰۶۶
۴	۰٫۰۵۲	۰٫۰۸۶	۰٫۱۰۴
۵	۰٫۰۲۴	۰٫۱۰۱	۰٫۰۸۲
۶	۰٫۰۰۹	۰٫۰۲۱	۰٫۰۱۸
۷	۰٫۰۲۶	۰٫۰۵۰	۰٫۰۲۷
۸	۰٫۱۷۴	۰٫۱۲۹	۰٫۱۴۴
۹	۰٫۰۰۲	۰٫۰۱۳	۰٫۰۴۳
۱۰	۰٫۰۴۰	۰٫۰۷۶	۰٫۰۶۲
۱۱	۰٫۱۲۵	۰٫۳۵۲	۰٫۳۳۲
۱۲	۰٫۱۱۱	۰٫۱۳۲	۰٫۱۳۳
۱۳	۰٫۰۶۵	۰٫۰۵۷	۰٫۰۵۱
۱۴	۰٫۰۰۶	۰٫۰۱۲	۰٫۰۱۲
۱۵	۰٫۰۸۰	۰٫۰۸۶	۰٫۰۹۱
Avg	۰٫۰۵۲	۰٫۰۸۲	۰٫۰۸۱

جدول ۴-۱۵: معیار «نرخ منفی کاذب» سه شبکه، با ۵۰۰ نورون پنهان

جدول ۴-۱۶: ماتریس درهم‌ریختگی برای یک مسئله طبقه‌بندی دو طبقه‌ای

	طبقه −	طبقه +
طبقه −	TN	FP
طبقه +	FN	TP

FN: این مقدار بیانگر تعداد نمونه‌هایی است که طبقه واقعی آن‌ها مثبت بوده و الگوریتم طبقه‌بندی طبقه آن‌ها را به اشتباه منفی تشخیص داده است.

TP: این مقدار بیانگر تعداد نمونه‌هایی است که طبقه واقعی آن‌ها مثبت بوده و الگوریتم طبقه‌بندی نیز طبقه آن‌ها را به درستی مثبت تشخیص داده است.

اکنون به تشریح انواع معیارهای ارزیابی الگوریتم‌های طبقه‌بندی با توجه به ماتریس درهم‌ریختگی می‌پردازیم. معیار اول برای تعیین کارایی یک الگوریتم طبقه‌بندی معیار **صحت**[۱] است. این معیار صحت یک طبقه‌بند را محاسبه می‌نماید. این معیار نشان‌دهنده این حقیقت است که طبقه‌بند طراحی شده چند درصد از کل مجموعه نمونه‌های تست را به درستی طبقه‌بندی کرده است .

$$Accuracy = \frac{TN + TP}{TN + FN + TP + FP}$$

معیار دوم **حساسیت**[۲] یا همان **نرخ تشخیص صحیح طبقه مثبت**[۳] (TPR) است . این معیار نشان می‌دهد که چه کسری از نمونه‌ها به درستی طبقه‌بندی شده‌اند، به عبارتی دیگر نشان می‌دهد که صحت تشخیص طبقه مثبت چه اندازه است.

$$TPR = \frac{TP}{FN + TP}$$

معیار سوم **نرخ مثبت کاذب**[۴] (FPR) است. این معیار بیانگر این حقیقت است که چه کسری از نمونه‌های منفی به اشتباه مثبت تشخیص داده خواهند شد و یا این که نرخ هشدار غلط را با توجه به طبقه منفی بیان می‌کند.

$$FPR = \frac{FP}{TN + FP}$$

معیار آخر **نرخ منفی کاذب**[۵] (FNR) است و نشان می‌دهد که چه کسری از نمونه‌های مثبت به اشتباه منفی تشخیص داده خواهند شد.

$$FNR = \frac{FN}{TP + FN}$$

[۱]Accuracy [۲]Sensivity [۳]True Positive Rate (TPR) [۴]False Positive Rate (FPR) [۵]False Negative Rate (FNR)

از معیارهای صحت، حساسیت، نرخ مثبت کاذب و نرخ منفی کاذب به منظور ارزیابی دقت روش‌ها استفاده شده و زمان آموزش[1] آن‌ها نیز مورد سنجش قرار گرفته است. در این مقاله نتایج مربوط به معیارهای صحت، حساسیت، نرخ مثبت کاذب و نرخ منفی کاذب برای تعداد ۱۰ نورون پنهان به ترتیب در جداول ۴-۴، ۴-۵، ۴-۶ و ۴-۷ ، برای تعداد ۱۰۰ نورون پنهان در جداول ۴-۸، ۴-۹، ۴-۱۰ و ۴-۱۱ و برای تعداد ۵۰۰ نورون پنهان در جداول ۴-۱۲، ۴-۱۳، ۴-۱۴ و ۴-۱۵ آورده شده است.

در تمامی این جداول ، هر سطر جدول بیانگر یک تصویر از مجموعه داده موردنظر است. سه ستون MLP، ELM و TELM نشان‌دهنده نتایج شبکه‌های مورد بحث‌اند. در هرسطر مورد با بهترین کارایی با رنگ زمینه خاکستری نشان داده شده است. آخرین سطر هر جدول میانگین مقادیر هر ستون را نشان می‌دهد و بهترین گزینه با زمینه خاکستری و بدترین با زمینه سیاه مشخص شده است. به منظور مقایسه بهتر، مقادیر سطر آخر جداول فوق الذکر (میانگین‌ها) براساس تعداد نورون‌های مختلف در لایه پنهان در نمودارهای ۴-۲، ۴-۳، ۴-۴، ۴-۵ و ۴-۶ آمده است.

۴-۱-۱ تجزیه و تحلیل نتایج

نتایج بصری حکایت از آن دارند که در حالت کلی، هر سه شبکه با ۱۰۰ نورون در لایه مخفی، بهترین خروجی‌ها را داشته‌اند. برای مقایسه دقیق‌تر مقادیر معیارهای کمی را مورد مداقه قرار می‌دهیم.

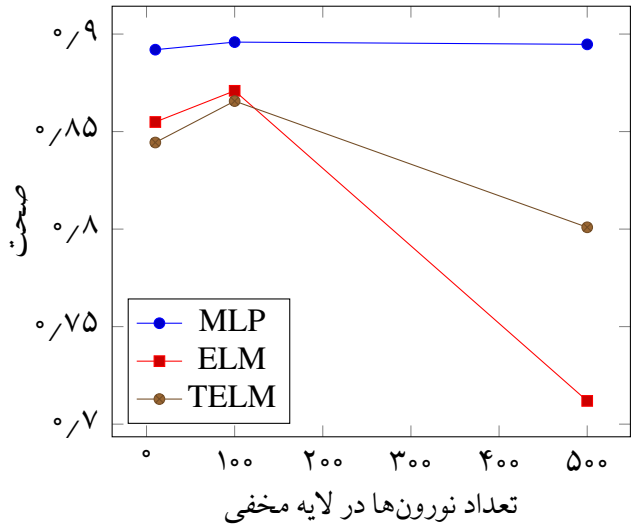
معیار صحت

براساس نتایج آزمایشات انجام شده و ملاحظه هر یک از جداول ۴-۴، ۴-۸ و ۴-۱۲ که معیار صحت را نشان می‌دهد شبکه عصبی MLP و ELM، MLP و TELM نتایج نسبتاً مشابهی دارند اما در حالت میانگین MLP بهتر است.

در مجموع و با ملاحظه نمودار ۴-۲ که میانگین همه را نشان می‌دهد مشخص است که با افزایش تعداد نورون‌های لایه پنهان، کارایی شبکه عصبی MLP به میزان کمی بیشتر شده است. اما در خصوص ELM و TELM نمی‌توان به صورت قاطع نظر داد. با افزایش تعداد نورون‌های لایه پنهان از ۱۰ به ۱۰۰، کارایی هر دو روش بهتر شده و تا حدود زیادی به کارایی MLP نزدیک شده است، اما با افزایش تعداد این نورون‌ها به ۵۰۰ نورون، کارایی هر دو به شدت افت کرده است. تصاویر خروجی در شکل‌های ۴-۱ تا ۴-۳ نیز مؤید همین مطلب است. نکته اصلی که در ادامه خواهیم دید آن است که افزایش کارایی MLP در حالت ۱۰۰ نورون در لایه پنهان به اندازه دو نیم درصد

[1] Training Time (TT)

بیشتر از ELM به بهای ۱۵۵ برابر شدن زمان آموزش به دست آمده است.



شکل ۴-۲: مقایسه میانگین «صحت» روش‌های مختلف در تعداد نورون‌های لایه پنهان مشخص شده.

معیار حساسیت

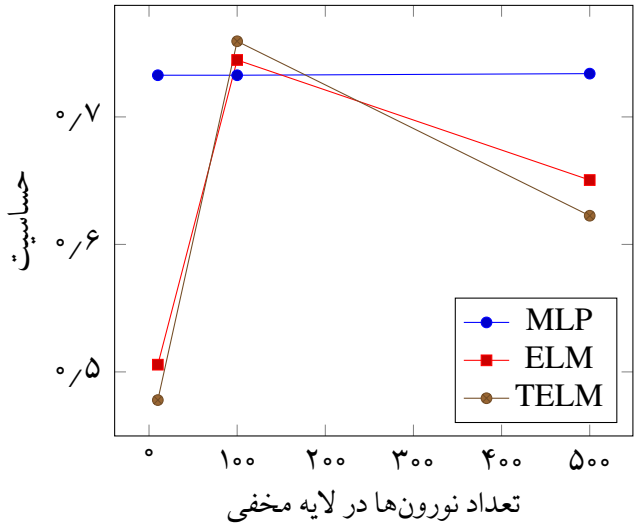
جداول ۴-۵، ۴-۹ و ۴-۱۳ معیار حساسیت را نشان می‌دهند، ملاحظه می کنیم در حالت میانگین با تعداد نورون‌های لایه پنهان برابر ۱۰ و ۵۰۰، شبکه عصبی MLP و با تعداد ۱۰۰ نورون در لایه پنهان، TELM بهتر است. در این حالت با این که هر یک از سه روش در یک سوم از تصاویر حائز بهترین امتیاز بوده‌اند (سطرهای خاکستری) اما در مجموع، MLP کمترین کارایی را داشته است.

در مجموع و با ملاحظه نمودار ۴-۳ که میانگین همه را نشان می‌دهد مشخص است که وضعیت مشابه معیار «صحت» است. افزایش تعداد نورون‌های لایه پنهان کارایی شبکه عصبی MLP را افزایش داده است اما ELM و TELM بهترین کارایی را در حالت ۱۰۰ نورون در لایه مخفی داشته‌اند.

معیار نرخ مثبت کاذب

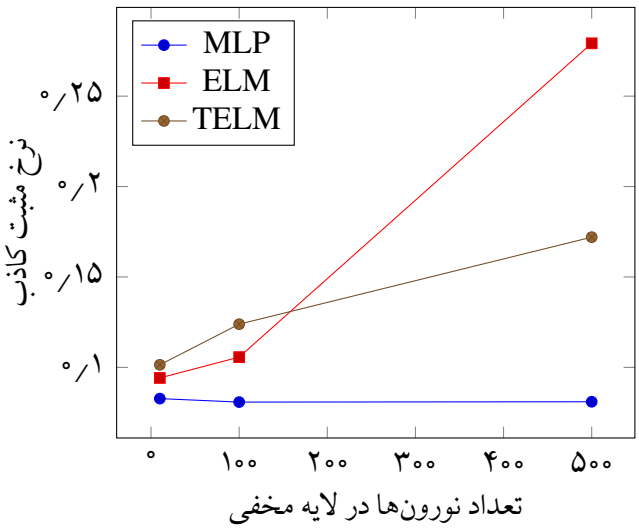
براساس ملاحظه جداول ۴-۶، ۴-۱۰ و ۴-۱۴ که معیار نرخ مثبت کاذب را نشان می‌دهد شبکه عصبی MLP و ELM، MLP و TELM نتایج نسبتاً مشابهی دارند مجدداً در حالت میانگین MLP از هر دو شبکه دیگر بهتر است. دقت داریم که هر چه مقدار این معیار کمتر باشد بهتر است.

در مجموع و با ملاحظه نمودار ۴-۴ که میانگین همه را نشان می‌دهد مشخص است که افزایش تعداد نورون‌های



شکل ۴-۳: مقایسه میانگین «حساسیت» روش‌های مختلف در تعداد نورون‌های لایه پنهان مشخص شده.

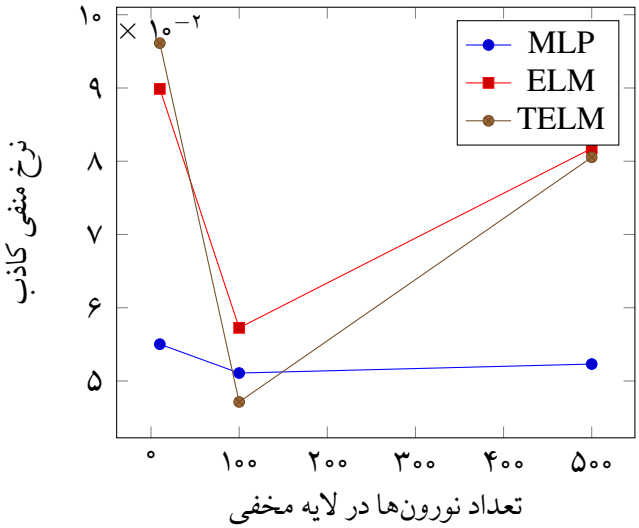
لایه پنهان معیار "نرخ مثبت کاذب" شبکه ELM و TELM را افزایش داده (کارایی کمتر) در حالی که شبکه عصبی MLP با افزایش تعداد نورون‌های لایه پنهان، کارایی بیشتری داشته است.



شکل ۴-۴: مقایسه میانگین «نرخ مثبت کاذب» روش‌های مختلف در تعداد نورون‌های لایه پنهان مشخص شده.

معیار نرخ منفی کاذب

جداول ۴-۷، ۴-۱۱ و ۴-۱۵ نرخ منفی کاذب را برای سه روش و بر روی ۱۵ تصویر مورد نظر نشان می‌دهند. این معیار نیز هر چه مقدار آن کمتر باشد بهتر است. در حالت‌های ۱۰ نورون و ۵۰۰ نورون در لایه پنهان، شبکه عصبی MLP پیشتاز است. فقط در حالت ۱۰۰ نورن در لایه پنهان (۴-۱۱) با این که هر یک از سه روش، در یک



شکل ۴-۵: مقایسه میانگین «نرخ منفی کاذب» روش‌های مختلف در تعداد نورون‌های لایه پنهان مشخص شده.

سوم از تصاویر بهترین کارایی را داشته‌اند، اما نسخه دولایه پنهان ELM از دو شیوه دیگر نتایج بهتری در حالت میانگین دارد.

در مجموع و با ملاحظه نمودار ۴-۵ که میانگین همه را نشان می‌دهد مشخص است که افزایش تعداد نورون‌های لایه پنهان "نرخ منفی کاذب" شبکه عصبی MLP را کاهش داده است ولی برای دو شیوه دیگر، بهترین کارایی در حالت ۱۰۰ نورون در لایه پنهان بوده است.

معیار زمان آموزش

براساس ملاحظه جداول ۴-۱۷، ۴-۱۸ و ۴-۱۹ که معیار زمان آموزش را نشان می‌دهند شبکه عصبی MLP نتایج بسیار متفاوتی با ELM ، دارد. برای تعداد برابر ۱۰، ۱۰۰ و ۵۰۰ نورون در لایه پنهان زمان آموزش MLP به ترتیب ۴۰۲ ، ۱۵۵ و ۵۰ برابر زمان آموزش ELM است. TELM زمان آموزشی اندکی بیش از ELM دارد ولی همچنان بسیار سریع‌تر از MLP است.

از نمودار ۴-۶ که میانگین همه را نشان می‌دهد مشخص است که افزایش تعداد نورون‌های لایه پنهان، زمان آموزش شبکه عصبی MLP را به میزان زیادی افزایش می‌دهد، در حالی که زمان آموزش ELM و TELM تقریباً وابسته به تعداد نورون‌ها لایه پنهان نیست و نسبتاً ثابت مانده است.

No.	MLP	ELM	TELM
۱	۳٫۴۹۸	۰٫۰۲۵	۰٫۰۱۹
۲	۵٫۲۸۹	۰٫۰۱۴	۰٫۰۳۴
۳	۱۲٫۸۲۸	۰٫۰۲۰	۰٫۰۳۷
۴	۸٫۹۵۵	۰٫۰۱۷	۰٫۰۲۵
۵	۶٫۰۱۱	۰٫۰۲۸	۰٫۰۲۸
۶	۵٫۵۷۵	۰٫۰۱۶	۰٫۰۲۷
۷	۶٫۷۶۳	۰٫۰۱۳	۰٫۰۲۸
۸	۸٫۹۲۸	۰٫۰۲۷	۰٫۰۳۳
۹	۳٫۳۱۹	۰٫۰۲۵	۰٫۰۲۲
۱۰	۸٫۲۳۸	۰٫۰۲۵	۰٫۰۲۵
۱۱	۵٫۷۳۰	۰٫۰۱۳	۰٫۰۲۵
۱۲	۱۸٫۱۹۱	۰٫۰۲۸	۰٫۰۳۴
۱۳	۱۰٫۲۳۱	۰٫۰۲۵	۰٫۰۳۴
۱۴	۹٫۹۴۴	۰٫۰۱۷	۰٫۰۳۴
۱۵	۱۳٫۲۹۲	۰٫۰۲۷	۰٫۰۳۳
Avg	۸٫۴۵۳	۰٫۰۲۱	۰٫۰۲۹

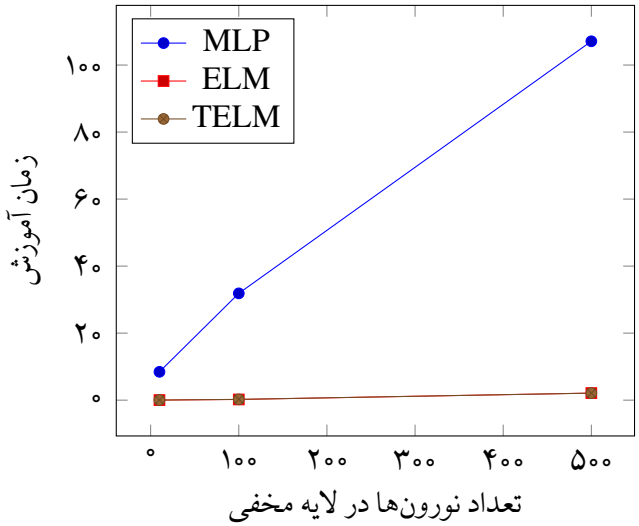
جدول ۴-۱۷: معیار «زمان آموزش» سه شبکه، با ۱۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۴۳٫۸۰۶	۱٫۳۲۳	۱٫۲۷۲
۲	۷۸٫۱۷۳	۲٫۰۹۷	۲٫۰۳۱
۳	۸۰٫۱۲۷	۲٫۳۵۸	۲٫۳۳۹
۴	۱۰۵٫۰۴۲	۱٫۹۰۶	۱٫۷۵۶
۵	۸۵٫۹۸۱	۲٫۰۰۸	۱٫۸۹۵
۶	۷۴٫۴۰۵	۱٫۷۹۲	۱٫۸۰۶
۷	۹۷٫۰۸۹	۱٫۵۲۳	۱٫۵۲۸
۸	۱۳۳٫۷۳۰	۲٫۶۵۳	۲٫۵۶۴
۹	۵۵٫۷۵۶	۱٫۵۵۵	۱٫۵۵۰
۱۰	۹۷٫۳۳۴	۲٫۴۰۰	۲٫۲۱۱
۱۱	۸۴٫۸۴۸	۱٫۴۵۸	۱٫۴۶۷
۱۲	۱۷۰٫۴۴۲	۳٫۰۸۱	۳٫۶۵۹
۱۳	۱۷۶٫۶۹۴	۲٫۵۵۸	۲٫۴۵۲
۱۴	۱۶۱٫۰۳۴	۲٫۶۵۳	۲٫۴۳۴
۱۵	۱۶۱٫۹۹۷	۲٫۳۳۳	۲٫۳۴۸
Avg	۱۰۷٫۰۹۷	۲٫۱۱۳	۲٫۰۸۸

جدول ۴-۱۹: معیار «زمان آموزش» سه شبکه، با ۵۰۰ نورون پنهان

No.	MLP	ELM	TELM
۱	۳۴٫۵۵۰	۰٫۱۳۳	۰٫۱۲۳
۲	۲۱٫۰۲۸	۰٫۲۰۰	۰٫۲۴۱
۳	۳۰٫۹۲۰	۰٫۲۳۰	۰٫۲۶۱
۴	۲۲٫۳۲۵	۰٫۱۸۱	۰٫۲۰۵
۵	۱۹٫۰۳۸	۰٫۱۸۹	۰٫۲۲۷
۶	۱۷٫۲۳۸	۰٫۱۶۷	۰٫۱۹۴
۷	۲۱٫۹۱۱	۰٫۱۵۵	۰٫۱۷۸
۸	۴۰٫۲۲۵	۰٫۲۶۱	۰٫۳۰۲
۹	۱۲٫۴۵۲	۰٫۱۵۵	۰٫۱۶۶
۱۰	۲۸٫۱۴۲	۰٫۲۲۷	۰٫۲۵۶
۱۱	۳۴٫۵۸۴	۰٫۱۶۱	۰٫۱۷۲
۱۲	۷۰٫۲۹۸	۰٫۲۹۸	۰٫۳۳۸
۱۳	۴۴٫۳۱۷	۰٫۲۴۱	۰٫۲۸۱
۱۴	۳۸٫۰۹۷	۰٫۲۴۲	۰٫۲۸۴
۱۵	۴۲٫۸۹۲	۰٫۲۴۲	۰٫۲۵۸
Avg	۳۱٫۸۶۸	۰٫۲۰۵	۰٫۲۳۲

جدول ۴-۱۸: معیار «زمان آموزش» سه شبکه، با ۱۰۰ نورون پنهان



شکل ۴-۶: مقایسه میانگین «زمان آموزش» روش‌های مختلف در تعداد نورون‌های لایه پنهان مشخص شده.

پیچیدگی حافظه موردنیاز

براساس تعداد N نمونه ورودی ، D نورون ورودی ، M نورون پنهان و K نورون خروجی در هر یک از شبکه‌ها و با فرض این که ماتریس نمونه‌های ورودی ، ماتریس خروجی واقعی مربوط به نمونه‌های ورودی ، ماتریس وزن‌های ورودی بین لایه ورودی و پنهان و ماتریس وزن‌های خروجی بین لایه پنهان و لایه خروجی به ترتیب دارای ابعاد $N * D$ ، $N * K$ ، $M * D$ و $M * K$ هستند بنابراین میزان حافظه مورد نیاز برای شبکه MLP، ELM و TELM به ترتیب در روابط (۴-۱) ، (۴-۲) و (۴-۳) آمده است

$$N(D+K+M)+M(D+K) \tag{۴-۱}$$

$$N(D+K+M)+M(D+K) \tag{۴-۲}$$

$$N(D+K+M/۲)+M/۲(D+K+M/۲) \tag{۴-۳}$$

نتایج فوق نشان می‌دهد که در حالت کلی اگر از ضریب $\frac{۱}{۲}$ صرف نظر کنیم پیچیدگی حافظه هر سه الگوریتم تقریباً مشابه یکدیگر است.

۲-۴ نتیجه گیری

تاکنون مقایسه جامعی در خصوص عملکرد ELM در قطعه‌بندی تصویر و براساس پایگاه داده‌های مشهور صورت نگرفته است. در این بخش دو نسخه مشهور ELM و شبکه عصبی MLP روی پایگاه داده BSD و در حالت دو کلاسی مورد مقایسه قرار گرفت. مقایسات با تعداد نورون‌های مختلف در لایه پنهان انواع شبکه‌های مورد بحث انجام پذیرفت. نتایج به صورت دیداری و هم چنین براساس معیارهای «صحت»، «حساسیت»، «نرخ مثبت کاذب»، «نرخ منفی کاذب» و «زمان آموزش» مورد ارزیابی قرار گرفتند. براساس نتایج به دست آمده، در حالت ۱۰۰ نورون در لایه پنهان، هر سه شبکه کارایی نسبتاً خوبی داشتند. در این وضعیت، در حالی که صحت MLP فقط حدود ۰٫۰۲۵ از ELM بیشتر است، اما زمان آموزش آن ۱۵۵ برابر زمان آموزش ELM است. وضعیت مشابهی در خصوص TELM برقرار است. لذا اگر در کاربرد مدنظر قطعه‌بندی، زمان آموزش از اهمیت بالایی برخوردار باشد و بتوان از مختصر کاهش دقت چشم‌پشی نمود، ماشین یادگیر نهایی (ELM) انتخاب بهتری نسبت به MLP می‌باشد.

از دیگر نتایج این بخش بررسی ادعای سرعت زیاد ماشین یادگیر نهایی می‌باشد. در مرجع [۲۶] ذکر شده است که سرعت یادگیری ELM می‌تواند هزاران مرتبه سریع‌تر از الگوریتم‌های یادگیری شبکه پیشخور سنتی هم چون BP باشد، اما در این بخش حداقل برای مسئله قطعه‌بندی تصویر ما به این نتیجه نرسیدیم و سرعت یادگیری ELM با تعداد ۱۰ نورون پنهان، ۱۰۰ نورون پنهان و ۵۰۰ نورون پنهان به ترتیب برابر ۴۰۲، ۱۵۵ و ۵۰ مرتبه سریع‌تر از الگوریتم BP در MLP می‌باشد.

فهرست منابع

[1] Qu, B.Y., Lang, B.F., Liang, J.J., Qin, A.K., and Crisalle, O.D. Two-hidden-layer extreme learning machine for regression and classification. *Neurocomputing.*, 175(PA):826–834, January 2016.

[2] Deng, ChenWei, Huang, GuangBin, Xu, Jia, and Tang, JieXiong. Extreme learning machines: new trends and applications. *Science China Information Sciences*, 58(2):1–16, 2015.

[3] Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 ed. , 2007.

[۴] منهاج، محمد باقر. هوش محاسباتی - جلد اول: مبانی شبکه‌های عصبی. دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)، ۱۳۹۳.

[۵] جکسون، راسل بیل و تام. آشنایی با شبکه‌های عصبی. ترجمه‌ی البرزی، محمود. دانشگاه صنعتی شریف، موسسه انتشارات علمی، ۱۳۹۳.

[6] Huang, Guang-Bin. What are extreme learning machines? filling the gap between frank rosenblatt's dream and john von neumann's puzzle. *Cognitive Computation*, 7:263–278, 2015.

[7] Watkins, David S. *Fundamentals of Matrix Computations*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[۸] رودین، والتر. اصول آنالیز ریاضی. ترجمه‌ی زاده، علی اکبر عالم، جلد بیست و چهارم. انتشارات علمی و فنی.

[۹] رودین، والتر. آنالیز حقیقی و مختلط. ترجمه‌ی زاده، علی اکبر عالم. ویرایش سوم.

[10] Huang, Guang-Bin, Chen, Lei, and Siew, Chee-Kheong. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, July 2006.

[11] Wang, Yuguang, Cao, Feilong, and Yuan, Yubo. A study on effectiveness of extreme learning machine. *Neurocomputing*, 74(16):2483 – 2490, 2011.

[12] Huang, Guang-Bin, Chen, Yan-Qiu, and Babri, H. A. Classification ability of single hidden layer feedforward neural networks. *IEEE Transactions on Neural Networks*, 11(3):799–801, may 2000.

[۱۳] فیاض، طیبه. خوشه‌بندی طیفی برای قطعه‌بندی تصاویر. پایان‌نامه کارشناسی‌ارشد، دانشگاه حکیم سبزواری، شهریور ماه ۱۳۹۴.

[14] Huang, Guang-Bin, Zhou, Hongming, Ding, Xiaojian, and Zhang, Rui. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics—part B: Cybernetics*, 42(2):513–529, April 2012.

[15] Li, Junpeng, Hua, Changchun, Tang, Yinggan, and Guan, Xinping. A fast training algorithm for extreme learning machine based on matrix decomposition. *Neurocomputing*, 173, Part 3:1951 – 1958, 2016.

[16] Liu, Xia, Lin, Shaobo, Fang, Jian, and Xu, Zongben. Is extreme learning machine feasible?a theoretical assessment (part i). *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 26(1), JANUARY 2015.

[17] Ulrike, Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, december 2007.

[18] Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elememnts of Statistical Learning*. Springer-Verlag New York.

[19] Huang, Gao, Huang, Guang-Bin, Song, Shiji, and You, Keyou. Trends in extreme learning machines: A review. *Neural Networks*, 61:32 – 48, 2015.

[20] Huang, Guang-Bin. An insight into extreme learning machines: Random neurons, random features and kernels. *Cognitive Computation*, 6(3):376–390, 2014.

[21] Kelley, J. L. and Stone, M. *General Topology*, vol. 233. 1995.

[۲۲] پورصدیق، سمیه. شناسایی فریم های زمینه ویدیو با استفاده از تجزیه .qr پایان‌نامه کارشناسی‌ارشد، دانشگاه حکیم سبزواری، بهمن ماه ۱۳۹۴.

[23] Ye, Yibin and Qin, Yang. Qr factorization based incremental extreme learning machine with growth of hidden nodes. *Pattern Recognition Letters*, 65(C):177–183, November 2015.

[24] Huang, G., Song, S., Gupta, J. N. D., and Wu, C. Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics*, 44(12):2405–2417, Dec 2014.

[25] Papadimitriou, C. H. and Steiglitz, K. *Combinatorial optimization:algorithms and complexity*. Courier Dover Publications, 1998.

[26] Huang, Guang-Bin, Zhu, Qin-Yu, and Siew, Chee Kheong. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

پیوست آ

برنامه‌های مرتبط با ELM

برنامه آ-۱: برنامه متلب پس انتشار گل‌های زنبق

```
n_input = 4; %number of input neurons
n_op = 3;  %number of  output neurons
n_hid = 4;  %number of hidden neurons

w_ih = rand(n_hid,n_input);
w_ho = rand(n_hid,n_op);

eta = 1;
k = 1;

load fisheriris

%define training set
in=[];
for i = 1:40

     temp = [meas(i,:);meas(50+i,:);meas(100+i,:)];
     in = [in; temp];

end

% coding (+1/-1) of 3 classes
a = [0.1 0.1 0.9];
b = [0.1 0.9 0.1];
c = [0.9 0.1 0.1];

% define targets
temp = [repmat(a,1,1); repmat(b,1,1); repmat(c,1,1)];
desired_out = repmat(temp,[40 1]);

iteration = 10;
error = zeros(iteration,n_op);

for iter = 1:iteration
```

```
    for j = 1:size(in,1)

%estimated output
        op_w = w_ih * in(j,:)';
        op_sig = 1 ./ (1 + exp(-(op_w)));
        out = 1 ./ (1 + exp(-(op_sig' * w_ho)));
        e = desired_out(j,:) - out;

%Output weights updation
        delta = (out .* (1 - out)) .* e;
        w_ho = w_ho + eta * op_sig * delta;

%Input weights updation
        delta_hid = op_sig' .* (1 - op_sig) '.* (delta * w_ho');
        w_ih = w_ih + eta * (in(j,:)' * delta_hid);

    end

    error(iter,:) = e;
    iter;

end

sse = sum((error(1:iter,:) .^ 2),2)

plot(sse);
title('error square plot for Iris Flowers training set');
xlabel('no of iterations');
ylabel('error.^2');

%Testing
in1 = [meas(41:50,:);meas(91:100,:);meas(141:150,:)];
out = [];

 for i = 1:size(in1,1)

    op_w = w_ih * in1(i,:)';
    op_sig = 1 ./ (1 + exp(-(op_w)));
    out(i,:) = 1 ./ (1 + exp(-(op_sig' * w_ho)));

 end

 out;
```

برنامه آ-۲: برنامه متلب ماشین یادگیر نهایی(ELM)

```
function [TrainingTime,TrainingAccuracy,TestingTime,TestingAccuracy]=
   ELM(TrainingData_File,TestingData_File,Elm_Type,
   NumberofHiddenNeurons,ActivationFunction)

% Input:
% TrainingData_File     - Filename of training data set
% TestingData_File      - Filename of testing data set
% Elm_Type              - 0 for regression; 1 for(binary and
```

```
%                           MultiClasses) classification
% NumberofHiddenNeurons - Number of hidden neurons assigned to the ELM
% ActivationFunction    - Type of activation function:
%                           'sig' for Sigmoidal function
%                           'sin' for Sine function
%                           'hardlim' for Hardlim function
%                           'tribas' for Triangular basis function
%                           'radbas' for Radial basis function
%
% Output:
% TrainingTime          - Time (seconds) spent on training ELM
% TestingTime           - Time (seconds) spent on predicting ALL
%                         testing data
% TrainingAccuracy      - Training accuracy:
%                         RMSE for regression or correct classification
%                         rate for classification
% TestingAccuracy       - Testing accuracy:
%                         RMSE for regression or correct
%                         classification rate for classification
%
% MULTI-CLASSE CLASSIFICATION: NUMBER OF OUTPUT NEURONS WILL BE
    AUTOMATICALLY SET EQUAL TO NUMBER OF CLASSES
% FOR EXAMPLE, if there are 7 classes in all, there will have 7 output
% neurons; neuron 5 has the highest output means input belongs to 5-th
    class

% regression:ELM('fisheriris_train','fisheriris_test',0,10,'sig')
% classification:ELM('fisheriris_train','fisheriris_test',1,10,'sig')

REGRESSION = 0;
CLASSIFIER = 1;

%%%%%%%%%% Load training dataset
train_data = load(TrainingData_File);
T = train_data(:,1);
X = train_data(:,2:size(train_data,2));
clear train_data;

%%%%%%%%%% Load testing dataset
test_data = load(TestingData_File);
TV.T = test_data(:,1);
TV.X = test_data(:,2:size(test_data,2));
clear test_data;

NumberofTrainingData = size(X,1);
NumberofTestingData = size(TV.X,1);
NumberofInputNeurons = size(X,2);

if Elm_Type~=REGRESSION
    %%%%%%%%%% Preprocessing the data of classification
    label = unique(T);
    number_class = numel(label);
    NumberofOutputNeurons = number_class;

    %%%%%%%%% Processing the targets of training
    temp_T = zeros(NumberofTrainingData,NumberofOutputNeurons);
```

```
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(j,1) == T(i,1)
                break;
            end
        end
        temp_T(i,j) = 1;
    end
    T = temp_T * 2 - 1;

    %%%%%%%%%% Processing the targets of testing
    temp_TV_T = zeros(NumberofTestingData,NumberofOutputNeurons);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(j,1) == TV.T(i,1)
                break;
            end
        end
        temp_TV_T(i,j) = 1;
    end
    TV.T = temp_TV_T * 2 - 1;

end

%%%%%%%%%% Random generate input weight and bias of hidden neurons
start_time_train=cputime;

W = rand(NumberofHiddenNeurons,NumberofInputNeurons) * 2 - 1;
b = rand(NumberofHiddenNeurons,1);
tempH = W * X';
clear X;
ind = ones(1,NumberofTrainingData);
B = b(:,ind);
tempH = tempH + B;

%%%%%%%%%% Calculate hidden layer output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = double(hardlim(tempH));
    case {'tribas'}
        %%%%%%%% Triangular basis function
        H = tribas(tempH);
    case {'radbas'}
        %%%%%%%% Radial basis function
        H = radbas(tempH);
end
clear tempH;
H = H';
```

```
%%%%%%%%%% Calculate output weights
OutputWeight = pinv(H) * T;

end_time_train = cputime;
TrainingTime = end_time_train - start_time_train

%%%%%%%%%% Calculate the ELM Network output
Y = H * OutputWeight;

%%%%%%%%%% Calculate the training accuracy
if Elm_Type == REGRESSION
    TrainingAccuracy = sqrt(mse(T - Y))
end
clear H;

%%%%%%%%%% Calculate the output of testing input
start_time_test = cputime;
tempH_test = W * TV.X';
clear TV.X;
ind = ones(1,NumberofTestingData);
B = b(:,ind);
tempH_test = tempH_test + B;

switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        H_test = sin(tempH_test);
    case {'hardlim'}
        H_test = hardlim(tempH_test);
    case {'tribas'}
        H_test = tribas(tempH_test);
    case {'radbas'}
        H_test = radbas(tempH_test);
end

 H_test = H_test';

%%%%%%%%%%TY: the actual output of the testing data
TY = H_test * OutputWeight;

end_time_test=cputime;
TestingTime = end_time_test - start_time_test

%%%%%%%%%% Calculate the testing accuracy
if Elm_Type == REGRESSION
    TestingAccuracy = sqrt(mse(TV.T - TY))
end

if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training and testing classification accuracy
    MissClassificationNumber_Training = 0;
    MissClassificationNumber_Testing = 0;

    for i = 1 : size(T,1)
        [~, label_index_expected] = max(T(i,:));
```

```
        [~, label_index_actual] = max(Y(i,:));
        if label_index_actual ~= label_index_expected
            MissClassificationNumber_Training =
               MissClassificationNumber_Training + 1;
        end
    end
    MissClassificationRate_Training = MissClassificationNumber_Training
        / size(T,1);
    TrainingAccuracy = 1 - MissClassificationRate_Training;
    TrainingError = 1 - TrainingAccuracy;

    for i = 1 : size(TV.T,1)
        [~, label_index_expected] = max(TV.T(i,:));
        [~, label_index_actual] = max(TY(i,:));
        if label_index_actual ~= label_index_expected
            MissClassificationNumber_Testing =
               MissClassificationNumber_Testing + 1;
        end
    end
    MissClassificationRate_Testing = MissClassificationNumber_Testing /
        size(TV.T,1);
    TestingAccuracy = 1 - MissClassificationRate_Testing;
    TestingError = 1 - TestingAccuracy;

end
```

برنامه آ-۳: برنامه متلب ELM افزایشی براساس تجزیه QR

```
function [TrainingTime,TestingTime,TrainingAccuracy,TestingAccuracy,
    TestingError] = QRI_ELM(TrainingData_File,TestingData_File,Elm_Type
    ,ActivationFunction,epsilon,Mmax)

% regression:QRI_ELM('fisheriris_train','fisheriris_test',0,'sig',.2,5)
% classification:QRI_ELM('fisheriris_train','fisheriris_test',1,'sig
    ',.2,5)

REGRESSION=0;
CLASSIFIER=1;

%%%%%%%%%% Load training dataset
train_data = load(TrainingData_File);
T = train_data(:,1);
X = train_data(:,2:size(train_data,2));

%%%%%%%%%% Load testing dataset
test_data = load(TestingData_File);
TY.T = test_data(:,1);
TV.X = test_data(:,2:size(test_data,2));

NumberofTrainingData = size(X,1);
NumberofInputNeurons = size(X,2);
NumberofTestingData = size(TV.X,1);

if Elm_Type~=REGRESSION
    %%%%%%%%%% Preprocessing the data of classification
```

```
label = unique(T);
number_class = numel(label);
NumberofOutputNeurons = number_class;

%%%%%%%%%% Processing the targets of training
temp_T = zeros(NumberofTrainingData,NumberofOutputNeurons);
for i = 1:NumberofTrainingData
    for j = 1:number_class
        if label(j,1) == T(i,1)
            break;
        end
    end
    temp_T(i,j) = 1;
end
T = temp_T * 2 - 1;

%%%%%%%%%% Processing the targets of testing
temp_TV_T = zeros(NumberofTestingData,NumberofOutputNeurons);
for i = 1:NumberofTestingData
    for j = 1:number_class
        if label(j,1) == TV.T(i,1)
            break;
        end
    end
    temp_TV_T(i,j) = 1;
end
TV.T = temp_TV_T * 2 - 1;

end

start_time_train = cputime;

W = rand(1,NumberofInputNeurons) * 2 - 1;
b = rand(1,1);
B = b(:,ones(1,NumberofTrainingData));
tempH = W * X' + B;

%Calculate the hidden-layer output matrix with single hidden node
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        H = sin(tempH);
    case {'hardlim'}
        H = double(hardlim(tempH));
    case {'tribas'}
        H = tribas(tempH);
    case {'radbas'}
        H = radbas(tempH);
end

H = H';

%%%%%%%%%%Calculate the inverse of R
p = (H'*H)^(-1/2);
```

```
%%%%%%%%%%Calculate Q
Q = H * p;

%%%%%Calculate the output weights between hidden layer and output layer
OutputWeight = p * Q' * T;

M = 1;
WeightInputs = [WeightInputs ; W];
BiasHidden = [BiasHidden ; b];

while  norm(T - H*OutputWeight,2) > epsilon && M <= Mmax

    W_Mplus1 = rand(1,NumberofInputNeurons) * 2 - 1;
    b_Mplus1 = rand(1,1);
    B_Mplus1 = b_Mplus1(:,ones(1,NumberofTrainingData));
    tempH_Mplus1 = W_Mplus1 * X '+ B_Mplus1;

    WeightInputs = [WeightInputs ; W_Mplus1];
    BiasHidden = [BiasHidden ; b_Mplus1];

    switch lower(ActivationFunction)
       case {'sig','sigmoid'}
          H_Mplus1 = 1 ./ (1 + exp(-tempH_Mplus1));
       case {'sin','sine'}
          H_Mplus1 = sin(tempH_Mplus1);
       case {'hardlim'}
          H_Mplus1 = double(hardlim(tempH_Mplus1));
       case {'tribas'}
          H_Mplus1 = tribas(tempH_Mplus1);
       case {'radbas'}
          H_Mplus1 = radbas(tempH_Mplus1);
    end

 H_Mplus1 = H_Mplus1';

%%%%%Calculate [Q R] of H Matrix after adding A New Hidden Node
deltaR_MPlus1 = Q'*H_Mplus1;

deltah_MPlus1 = H_Mplus1-Q*deltaR_MPlus1;

p(M+1,M+1) = 1 / sqrt(deltah_MPlus1' * deltah_MPlus1);

q_Mplus1 = deltah_MPlus1 * p(M+1,M+1);

OutputWeight_Mplus1 = p(M+1,M+1) * q_Mplus1' * T;

OutputWeight = [OutputWeight - p(1:M,1:M) * deltaR_MPlus1 *
    OutputWeight_Mplus1;
     OutputWeight_Mplus1];

p = [p(1:M,1:M)        -p(1:M,1:M) * deltaR_MPlus1 * p(M+1,M+1)
     zeros(1,M)           p(M+1,M+1)];

Q = [Q q_Mplus1];

H = [H H_Mplus1];
```

```
M = M+1;
end

end_time_train = cputime;
TrainingTime = end_time_train - start_time_train;

Y_qr = H * OutputWeight;

if Elm_Type == REGRESSION
    TrainingAccuracy = sqrt(mse(T - Y_qr))
end

%%%%%%%%%%Calculate the output of testing input
start_time_test = cputime;

B = BiasHidden(:,ones(1,NumberofTestingData));
tempH_test = WeightInputs * TV.X' + B;

switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        H_test = sin(tempH_test);
    case {'hardlim'}
        H_test = hardlim(tempH_test);
    case {'tribas'}
        H_test = tribas(tempH_test);
    case {'radbas'}
        H_test = radbas(tempH_test);
end

H_test = H_test';

%%%%%%%%%%TY:the actual output of the testing data
TY = H_test * OutputWeight;

end_time_test = cputime;
TestingTime = end_time_test - start_time_test;

if Elm_Type == REGRESSION
    TestingAccuracy = sqrt(mse(TV.T - TY))
end

if Elm_Type == CLASSIFIER
%%%%%%%%%%Calculate training and testing classification accuracy
MissClassificationNumber_Training = 0;
MissClassificationNumber_Testing = 0;

for i = 1 : size(T,1)
    [~, label_index_expected] = max(T(i,:));
    [~, label_index_actual] = max(Y_qr(i,:));
    if label_index_actual ~= label_index_expected
        MissClassificationNumber_Training =
            MissClassificationNumber_Training + 1;
    end
```

```
end

    MissClassificationRate_Training = MissClassificationNumber_Training
        / size(T,1);
    TrainingAccuracy = 1 - MissClassificationRate_Training;
    TrainingError = 1 - TrainingAccuracy;

for i = 1 : size(TV.T,1)
    [~, label_index_expected] = max(TV.T(i,:));
    [~, label_index_actual] = max(TY(i,:));
    if label_index_actual ~= label_index_expected
        MissClassificationNumber_Testing =
           MissClassificationNumber_Testing + 1;
    end
end

    MissClassificationRate_Testing = MissClassificationNumber_Testing /
        size(TV.T,1);
    TestingAccuracy = 1 - MissClassificationRate_Testing;
    TestingError = 1 - TestingAccuracy;
end
```

برنامه آ-۴: برنامه متلب ELM سریع مبتنی بر تجزیه

```
function [TrainingTime,TrainingAccuracy,TestingTime,TestingAccuracy]=
   DFELM(TrainingData_File,TestingData_File, Elm_Type,
   NumberofHiddenNeurons, ActivationFunction)

% regression:DFELM('fisheriris_train', 'fisheriris_test', 0, 10, 'sig')
% classification:DFELM('fisheriris_train', 'fisheriris_test', 1, 10, '
   sig')
%M1=M2=M/2

REGRESSION = 0;
CLASSIFIER = 1;

%%%%%%%%%%Load training dataset
train_data = load(TrainingData_File);
T = train_data(:,1);
X = train_data(:,2:size(train_data,2));
clear train_data;

%%%%%%%%%%Load testing dataset
test_data = load(TestingData_File);
TV.T = test_data(:,1);
TV.X = test_data(:,2:size(test_data,2));
clear test_data;

NumberofTrainingData = size(X,1);
NumberofTestingData = size(TV.X,1);
NumberofInputNeurons = size(X,2);

if Elm_Type ~= REGRESSION
    %%%%%%%%%%Preprocessing the data of classification
    label = unique(T);
```

```
    number_class = numel(label);
    NumberofOutputNeurons = number_class;

    %%%%%%%%%%Processing the targets of training
    temp_T = zeros(NumberofTrainingData,NumberofOutputNeurons);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(j,1) == T(i,1)
                break;
            end
        end
        temp_T(i,j) = 1;
    end
    T = temp_T * 2 - 1;

    %%%%%%%%%%Processing the targets of testing
    temp_TV_T = zeros(NumberofTestingData,NumberofOutputNeurons);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(j,1) == TV.T(i,1)
                break;
            end
        end
        temp_TV_T(i,j) = 1;
    end
    TV.T = temp_TV_T * 2 - 1;

end

start_time_train = cputime;

%%%%%Random generate input weights  and biases of hidden neurons
W = rand(NumberofHiddenNeurons,NumberofInputNeurons);
b = rand(NumberofHiddenNeurons,1);
B = b(:,ones(1,NumberofTrainingData));
tempH = W * X'+ B;

%%%%%%%%%%Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        H = sin(tempH);
    case {'hardlim'}
        H = double(hardlim(tempH));
    case {'tribas'}
        H = tribas(tempH);
    case {'radbas'}
        H = radbas(tempH);
end

H = H';

%%%%%%%%%%Calculate output weights
M1 = NumberofHiddenNeurons / 2;
M2 = NumberofHiddenNeurons / 2;
```

```
H1 = H(:,1:M1);
H2 = H(:,M1+1:end);

E = H2' * H2;
F = pinv(E) * H2' * T;
G = eye(NumberofTrainingData,NumberofTrainingData) - H2 * pinv(E)*H2';
V = H1' * G * H1;

OutputWeight1 = pinv(V) * H1' * (T - H2 * F);
OutputWeight2 = F - pinv(E) * H2' * H1 * OutputWeight1;
OutputWeight = [OutputWeight1;OutputWeight2];

end_time_train = cputime;
TrainingTime = end_time_train - start_time_train

%%%%%%%%%%Y:the actual output of the training data
Y = H * OutputWeight;

%%%%%%%%%%Calculate the training accuracy
if Elm_Type == REGRESSION
    TrainingAccuracy = sqrt(mse(T - Y))
end
clear H;

%%%%%%%%%%Calculate the output of testing input
start_time_test = cputime;

tempH_test = W * TV.X';
B = b(:,ones(1,NumberofTestingData));
tempH_test = tempH_test + B;

switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        H_test = sin(tempH_test);
    case {'hardlim'}
        H_test = hardlim(tempH_test);
    case {'tribas'}
        H_test = tribas(tempH_test);
    case {'radbas'}
        H_test = radbas(tempH_test);
end

H_test = H_test';

%%%%%%%%%%TY: the actual output of the testing data
TY = H_test * OutputWeight;

end_time_test = cputime;
TestingTime = end_time_test - start_time_test

%%%%%%%%%%Calculate testing accuracy (RMSE) for regression case
if Elm_Type == REGRESSION
    TestingAccuracy = sqrt(mse(TV.T - TY))
```

```
end

if Elm_Type == CLASSIFIER
%%%%%%%%%%Calculate training and testing classification accuracy
    MissClassificationNumber_Training = 0;
    MissClassificationNumber_Testing = 0;

    for i = 1 : size(T, 1)
        [~, label_index_expected] = max(T(i,:));
        [~, label_index_actual] = max(Y(i,:));
        if label_index_actual ~= label_index_expected
            MissClassificationNumber_Training =
                MissClassificationNumber_Training + 1;
        end
    end

    MissClassificationRate_Training = MissClassificationNumber_Training
        / size(T,1);
    TrainingAccuracy = 1 - MissClassificationRate_Training;
    TrainingError = 1 - TrainingAccuracy;

    for i = 1 : size(TV.T, 1)
        [~, label_index_expected] = max(TV.T(i,:));
        [~, label_index_actual] = max(TY(i,:));
        if label_index_actual ~= label_index_expected
            MissClassificationNumber_Testing =
                MissClassificationNumber_Testing + 1;
        end
    end

    MissClassificationRate_Testing = MissClassificationNumber_Testing /
        size(TV.T,1);
    TestingAccuracy = 1 - MissClassificationRate_Testing;
    TestingError = 1 - TestingAccuracy;

end
```

برنامه آ-۵: برنامه متلب ELM دو لایه پنهان

```
function  [TrainingTime,TrainingAccuracy,TestingTime,TestingAccuracy]=
   TELM(TrainingData_File,TestingData_File,Elm_Type,
   NumberofHiddenNeurons,ActivationFunction)

% regression:TELM('fisheriris_train','fisheriris_test',0,10,'sig')
% classification:TELM('fisheriris_train','fisheriris_test',1,10,'sig')

REGRESSION = 0;
CLASSIFIER = 1;

%%%%%%%%% Load training dataset
train_data = load(TrainingData_File);
T = train_data(:,1);
X = train_data(:,2:size(train_data,2));
clear train_data;
```

```
%%%%%%%%%% Load testing dataset
test_data = load(TestingData_File);
TV.T = test_data(:,1);
TV.X = test_data(:,2:size(test_data,2));
clear test_data;

NumberofTrainingData = size(X,1);
NumberofTestingData = size(TV.X,1);
NumberofInputNeurons = size(X,2);

if Elm_Type ~= REGRESSION
    %%%%%%%%%% Preprocessing the data of classification
    label = unique(T);
    number_class = numel(label);
    NumberofOutputNeurons = number_class;

    %%%%%%%%%% Processing the targets of training
    temp_T = zeros(NumberofTrainingData,NumberofOutputNeurons);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(j,1) == T(i,1)
                break;
            end
        end
        temp_T(i,j) = 1;
    end
    T = temp_T * 2 - 1;

    %%%%%%%%%%Processing the targets of testing
    temp_TV_T = zeros(NumberofTestingData,NumberofOutputNeurons);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(j,1) == TV.T(i,1)
                break;
            end
        end
        temp_TV_T(i,j) = 1;
    end
    TV.T = temp_TV_T * 2 - 1;

end

start_time_train = cputime;

%%%%%Random generate the connection weight matrix between the input
    layer and the first hidden layer
W = rand(NumberofHiddenNeurons/2,NumberofInputNeurons) * 2 - 1;

%%%%%%%%%%The bias matrix of the first hidden layer
B = rand(NumberofHiddenNeurons/2,1);

X_E = [ones(NumberofTrainingData,1) X]';
W_IE = [B W];

%%%%%%%%%% Calculate first hidden layer output matrix H
switch lower(ActivationFunction)
```

```
    case {'sig','sigmoid'}
        H = 1 ./ (1 + exp(-tempH));
    case {'hypertan'}
        H = (1-exp(-tempH))./(1+exp(-tempH));
end
clear tempH;

H = H';

%%%%%Calculate weights matrix between second hidden layer and output
    layer
Beta = pinv(H) * T;

%%%%%%%%%%Expected output of the second hidden layer
H1 = T * pinv(Beta);

%%%%%%%%%%g^-1(x):The inverse of the activation function g(x).
switch lower(ActivationFunction)
   case {'sig','sigmoid'}
        inv_func_H1 =-log((1-H1)./H1);
    case {'hypertan'}
        inv_func_H1 =-log((1-H1)./(1+H1));
end

inv_func_H1 = real(inv_func_H1);

%H_E=[1 H]
H_E = [ones(NumberofTrainingData,1) H];

%%%%%%%%%%Augmented matrix W_HE=[B_1 W_H]
W_HE = (inv_func_H1)' * pinv(H_E');

%%%%%%%%%%B_1 Bias of the second hidden layer.
B_1 = W_HE(:,1:NumberofTrainingData);

%%%%%W_H weight matrix between the first hidden layer and the second
    hidden layer
W_H = W_HE(:,NumberofTrainingData+1:end);

HH = W_HE * H_E';

%%%%%%%%%%The actual output of the second hidden layer
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H2 = 1 ./ (1 + exp(-HH));
    case {'hypertan'}
        H2 = (1-exp(-HH))./(1+exp(-HH));
end

H2 = (H2)';

%Recalculate the weight matrix new between the second hidden layer and
    the output layer
Beta_new = pinv(H2) * T;

end_time_train = cputime;
```

```
TrainingTime = end_time_train - start_time_train;

%%%%%%%%%%Y:The actual output of the training data
Y = H2 * Beta_new;

%%%%%%%%%% Calculate the training accuracy
if Elm_Type == REGRESSION
    TrainingAccuracy = sqrt(mse(T - Y))
end
clear H;

%%%%%%%%%% Calculate the output of testing input
start_time_test = cputime;

B_testLayer1 = B(:,ones(1,NumberofTestingData));

switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H_testLayer1 = 1 ./ (1 + exp(-(W*TV.X' + B_testLayer1)));
    case {'hypertan'}
        H_testLayer1 = (1-exp(-(W*TV.X' + B_testLayer1)))./(1+exp(-(W*
            TV.X' + B_testLayer1)));
end

H_testLayer1 = H_testLayer1';

B_testLayer2 = B_1(:,ones(1,NumberofTestingData));

switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H_testLayer2 = 1 ./ (1 + exp(-(W_H*H_testLayer1' + B_testLayer2
            )));
    case {'hypertan'}
        H_testLayer2 = (1-exp(-(W_H*H_testLayer1' + B_testLayer2)))
            ./(1+exp(-(W_H*H_testLayer1' + B_testLayer2)));
end

H_testLayer2=H_testLayer2';

Y_test = H_testLayer2*Beta_new;

end_time_test=cputime;
TestingTime=end_time_test-start_time_test;

if Elm_Type == REGRESSION
    TestingAccuracy=sqrt(mse(TV.T - Y_test))
end

if Elm_Type == CLASSIFIER

    %Calculate training & testing classification accuracy

    MissClassificationNumber_Training = 0;
    MissClassificationNumber_Testing = 0;

    for i = 1 : size(T, 1)
```

```
        [~, label_index_expected] = max(T(i,:));
        [~, label_index_actual] = max(Y(i,:));
        if label_index_actual ~= label_index_expected
            MissClassificationNumber_Training =
               MissClassificationNumber_Training + 1;
        end
    end

    MissClassificationRate_Training = MissClassificationNumber_Training
       /NumberofTrainingData;
    TrainingAccuracy = 1 - MissClassificationRate_Training;
    TrainingError = 1 - TrainingAccuracy;

        for i = 1 : size(TV.T, 1)
            [~, label_index_expected] = max(TV.T(i,:));
            [~, label_index_actual] = max(Y_test(i,:));
            if label_index_actual ~= label_index_expected
                MissClassificationNumber_Testing =
                   MissClassificationNumber_Testing + 1;
            end
        end
        MissClassificationRate_Testing =
           MissClassificationNumber_Testing/NumberofTestingData;
        TestingAccuracy = 1 - MissClassificationRate_Testing;
        TestingError = 1 - TestingAccuracy;
end
end
```

برنامه آ-۶: برنامه متلب ماشین یادگیر نهایی نیمه ناظر (SS-ELM)

```
format compact;
addpath(genpath('functions'))

%%%%%%%%%%load data
trial=1;
load g50c;

%%%%%%%%%%Create labeled data set
l=size(idxLabs,2);
u=ceil(size(y,1)*3/4)-2*l;
Xl=X(idxLabs(trial,:),:);
Yl=y(idxLabs(trial,:),:);

%%%%%%%%%%Creat validation data set
labels=unique(y);
idx_V=[];
for i=1:size(labels)
  idx_V=[idx_V;find(y(idxUnls(trial,:))==labels(i),l/length(labels),'
     first')];
end
Xv=X(idxUnls(trial,idx_V),:);
Yv=y(idxUnls(trial,idx_V));

%%%%%%%%%%Create unlabeled and testing data set
idxSet=1:size(idxUnls,2);
```

```
idx_UT=setdiff(idxSet,idx_V);
idx_rand=randperm(size(idx_UT,2));
Xu=X(idxUnls(trial,idx_UT(idx_rand(1:u))),:);
Yu=y(idxUnls(trial,idx_UT(idx_rand(1:u))),:);
Xt=X(idxUnls(trial,idx_UT(idx_rand(u+1:end))),:);
Yt=y(idxUnls(trial,idx_UT(idx_rand(u+1:end))),:);

%%%%%%%%%%Compute graph Laplacian
options.NN=50;
options.GraphWeights='binary';
options.GraphDistanceFunction='euclidean';
options.LaplacianNormalize=1;
options.LaplacianDegree=5;
L=laplacian(options,[Xl;Xu]);

paras.NumHiddenNeuron=2000;
paras.NoDisplay=1;
paras.ActivationFunction='sigmoid';

%%%%%%%%%% model selection using the validation set
acc_v=zeros(10,10);
acc_test=zeros(10,10);
acc_max=0;
for i=1:10
    paras.C=10^(i-5);
    for j=1:10
        paras.lambda=10^(7-j);
        elmModel=sselm(Xl,Yl,Xu,L,paras);
        [acc_v(i,j),MSE(i,j)]=sselm_predict(Xv,Yv,elmModel);
        [acc_test(i,j),~]=sselm_predict(Xt,Yt,elmModel);
        if acc_v(i,j)>acc_max
            acc_max=acc_v(i,j);
            elmModel_best=elmModel;
        end
    end
end

[acc_tmp,~]=sselm_predict(Xu,Yu,elmModel_best);
err_u(trial)=100-acc_tmp;

[acc_tmp,~,~]=sselm_predict(Xv,Yv,elmModel_best);
err_v(trial)=100-acc_tmp;

[acc_tmp,~,~]=sselm_predict(Xt,Yt,elmModel_best);
err_t(trial)=100-acc_tmp;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [L,options] = laplacian(options,X)
%computes the graph Laplacian.
%options:a structure with the following fields
%               options.NN: number of nearest neighbors to use
%               options.GraphDistanceFunction: 'euclidean' | 'cosine' |
%               'hamming_distance'
%               options.GraphWeights: 'distance' | 'binary' | 'heat'
%               options.GraphWeightParam: width for 'heat' kernel
%               (if set to 0, it uses the mean edge length distance
%                among neighbors)
```

```
%                options.LaplacianNormalize: 0 | 1
%                options.LaplacianDegree: degree of the iterated
%                 Laplacian
% X:N-by-D data matrix (N examples, D dimensions)
% L: sparse symmetric N-by-N Laplacian matrix
%     options:updated options structure with estimated heat kernel
%     width(only when you select to use 'heat' in the GraphWeights
%     option and 'default' as GraphWeightParam)
W = adjacency(options,X);
D = sum(W,2);

if options.LaplacianNormalize == 0
    L = spdiags(D,0,speye(size(W,1)))-W; % L = D-W
else
    D(D~=0)=sqrt(1./D(D~=0));
    D=spdiags(D,0,speye(size(W,1)));
    W=D*W*D;
    L=speye(size(W,1))-W; % L = I-D^-1/2*W*D^-1/2
end

if options.LaplacianDegree>1
    L=mpower(L,options.LaplacianDegree);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = adjacency(options,X)
%computes the graph adjacency matrix
%A:sparse symmetric N-by-N adjacency matrix
n=size(X,1);
p=2:(options.NN+1);

if n<500
    step=n;
else
        step=500;
end

idy=zeros(n*options.NN,1);
DI=zeros(n*options.NN,1);
t=0;
s=1;

for i1=1:step:n
    t=t+1;
    i2=i1+step-1;
    if (i2>n)
        i2=n;
    end

    Xblock=X(i1:i2,:);
    dt=feval(options.GraphDistanceFunction,Xblock,X);
    [Z,I]=sort(dt,2);
    Z=Z(:,p)';
    I=I(:,p)';
    [g1,g2]=size(I);
    idy(s:s+g1*g2-1)=I(:);
    DI(s:s+g1*g2-1)=Z(:);
```

```
    s=s+g1*g2;
end

I=repmat((1:n),[options.NN 1]);
I=I(:);

if strcmp(options.GraphDistanceFunction,'cosine')
    DI=DI.*(DI<1);
end

switch options.GraphWeights
    case 'distance'
        A=sparse(I,idy,DI,n,n);
    case 'binary'
        A=sparse(I,idy,1,n,n);
    case 'heat'
        if options.GraphWeightParam==0
            t=mean(DI(DI~=0));
        else
            t=options.GraphWeightParam;
        end
        A=sparse(I,idy,exp(-DI.^2/(2*t*t)),n,n);
    otherwise
        error('Unknown weight type');
end

A=A+((A~=A').*A'); %symmetrize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D = euclidean(A,B)
% computes the Euclidean distance.
% A: M-by-P , B: N-by-P
% D: M-by-N distance matrix
if (size(A,2) ~= size(B,2))
    error('A and B must be of same dimensionality.');
end

if (size(A,2) == 1)
    A = [A, zeros(size(A,1),1)];
    B = [B, zeros(size(B,1),1)];
end

aa=sum(A.*A,2);
bb=sum(B.*B,2);
ab=A*B';
D = real(sqrt(repmat(aa,[1 size(bb,1)]) + repmat(bb',[size(aa,1) 1])
   -2*ab));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function elmModel=sselm(Xl,Yl,Xu,L,paras)
[l,elmModel.InputDim]=size(Xl);
u=size(Xu,1);
N=l+u;

%Decide whether it is a binary or multi-class problem
elmModel.labs=unique(Yl);
if length(elmModel.labs)==2
    elmModel.MultiOutput=0;
```

```
    elmModel.OutputDim=1;
else
    elmModel.MultiOutput=1;
    elmModel.OutputDim=length(elmModel.labs);
end

%Random generate input weights
elmModel.W=rand(paras.NumHiddenNeuron,elmModel.InputDim)*2-1;
elmModel.b=rand(paras.NumHiddenNeuron,1);
elmModel.B=elmModel.b(:,ones(1,N));

%Calculate hidden layer output matrix
elmModel.ActivationFunction=paras.ActivationFunction;
switch paras.ActivationFunction
    case 'sigmoid'
        H=1 ./ (1 + exp(-(elmModel.W*[Xl;Xu]')+elmModel.B));
    case 'rbf'
        H=calckernel(elmModel,elmModel.W,X');
end

H=H';
Hl=H(1:l,:);
clear Xl Xu

%Calculate output weights
if elmModel.MultiOutput==0  %Binary classification
    Y=[Yl;zeros(u,1)];
    Cl_diag=l/2*((Yl==max(Yl))/sum(Yl==max(Yl))+(Yl==min(Yl))/sum(Yl==
        min(Yl)));
else    %Multi-class classification
    Y=zeros(N,elmModel.OutputDim);
    Y(1:l,:)=-1;
    Cl_diag=zeros(l,1);
    for i=1:elmModel.OutputDim
        Y(Yl==elmModel.labs(i),i)=1;
        Cl_diag=Cl_diag+(Yl==elmModel.labs(i))/(sum(Yl==elmModel.labs(i
            ))*elmModel.OutputDim/l);
    end
end

Cl=diag(paras.C*Cl_diag);
if  (paras.NumHiddenNeuron>N)
%Comput C only the training instances is less than hidden nodes
    C=diag([paras.C*Cl_diag;zeros(u,1)]);
end

t_elm_start=tic;
if  (paras.NumHiddenNeuron<N)
    elmModel.OutputWeight=(eye(paras.NumHiddenNeuron)+Hl'*Cl*Hl+paras.
        lambda*H'*L*H)\ (Hl'*Cl* Y(1:l,:));
else
    A=eye(N)+(C+paras.lambda*L)*(H*H');
    B=C*Y;
    D=A\B;
    elmModel.OutputWeight=H'*D;    %(Beta)
end
```

```
elmModel.TrainTime=toc(t_elm_start);

%Calculate training accuracy
if elmModel.MultiOutput==0  % Binary classification
    TrainAccuracy=100*mean(sign(Hl*elmModel.OutputWeight)==Yl);
    disp(['  Traning time is=',num2str(elmModel.TrainTime)])
    disp(['  TrainAccuracy=',num2str(TrainAccuracy)])
    disp('************************')
else     %Multi-class classification
    [~,idx]=max((Hl*elmModel.OutputWeight)');
    TrainAccuracy=100*mean(elmModel.labs(idx)==Yl);
    disp(['  Traning time is=',num2str(elmModel.TrainTime)])
    disp(['  TrainAccuracy=',num2str(TrainAccuracy)])
    disp('************************')
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [acc,MSE,auc,predict]=sselm_predict(X,Y,elmModel)
%Calculate hidden neuron output matrix
switch elmModel.ActivationFunction
    case 'sigmoid'
        H=1 ./ (1 + exp(-elmModel.W*X'));
    case 'rbf'
        H=calckernel(elmModel,elmModel.W,X');
end

%Calculate training accuracy
if elmModel.MultiOutput==0  % Binary classification
    out=H*elmModel.OutputWeight;
    acc=100*mean(sign(H*elmModel.OutputWeight)==Y);
    [~,~,~,auc] = perfcurve(Y,out,1);
    predict=sign(out);
    MSE=mse(Y-out);
else        %Multi-class classification
    y=-ones(length(Y),elmModel.OutputDim);
    for i=1:elmModel.OutputDim
        y(Y==elmModel.labs(i),i)=1;
    end
    out=H*elmModel.OutputWeight;
    [~,idx]=max(out');
    predict=elmModel.labs(idx);
    acc=100*mean(predict==Y);
    auc=NaN;
    MSE=mse(y-out);
end
```

برنامه آ-۷: برنامه متلب ماشین یادگیر نهایی بدون ناظر (US-ELM)

```
%Unsupervised ELM for embedding and clustering.
format compact;
addpath(genpath('functions'))

%load data
data=load ('iris.txt');
X=data(:,1:end-1);
```

```
y=data(:,end);
NC=length(unique(y)); %specify number of clusters

%hyper-parameter settings for graph
options.GraphWeights='binary';
options.GraphDistanceFunction='euclidean';
options.LaplacianNormalize=0;
options.LaplacianDegree=1;
options.NN=5;

%Step 1: construct graph Laplacian
L=laplacian(options,X);

%Step 2: Run US-ELM for embedding
%hyper-parameter settings for us-elm
paras.NE=3; % specify dimensions of embedding
paras.NumHiddenNeuron=2000;
paras.NormalizeInput=0;
paras.NormalizeOutput=0;
paras.ActivationFunction='sigmoid';
paras.lambda=0.1;
elmModel=uselm(X,L,paras);

%Step 3: Run k-means for clustering
acc_kmeans=[];acc_le=[];acc_uselm=[];
for i=1:100
    [label_kmeans, center] = litekmeans(X,NC,'MaxIter', 200);
    [label_uselm, center] = litekmeans(elmModel.Embed, NC, 'MaxIter',
       200);
end

%3-D plot of the results
figure(1)
E=X;
hold on
title('The original IRIS data')
view(3)
plot3(E(y==1,1),E(y==1,2),E(y==1,3),'gx','MarkerSize',8,'LineWidth'
    ,1.5)
plot3(E(y==2,1),E(y==2,2),E(y==2,3),'c+','MarkerSize',6,'LineWidth'
    ,1.5)
plot3(E(y==3,1),E(y==3,2),E(y==3,3),'b.','MarkerSize',10,'LineWidth'
    ,1.5)
grid on
axis square

figure(2)
E=elmModel.Embed;
hold on
title('The embedded IRIS data')
view(3)
plot3(E(y==1,1),E(y==1,2),E(y==1,3),'gx','MarkerSize',8,'LineWidth'
    ,1.5)
plot3(E(y==2,1),E(y==2,2),E(y==2,3),'c+','MarkerSize',6,'LineWidth'
    ,1.5)
plot3(E(y==3,1),E(y==3,2),E(y==3,3),'b.','MarkerSize',10,'LineWidth'
```

```
    ,1.5)
grid on
axis square

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function elmModel=uselm(X,L,paras)
[N,elmModel.InputDim]=size(X);

%Normalize the input
elmModel.NormalizeInput=paras.NormalizeInput;
if paras.NormalizeInput
    [X,elmModel.PreProcess]=mapminmax(X,-1,1);
end

%Random generate input weights
elmModel.W=rand(paras.NumHiddenNeuron,elmModel.InputDim)*2-1;

%Calculate hidden neuron output matrix
elmModel.ActivationFunction=paras.ActivationFunction;
switch paras.ActivationFunction
    case 'sigmoid'
        H=1 ./ (1 + exp(-elmModel.W*X'));
    case 'rbf'
        H=calckernel(elmModel,elmModel.W,X');
end

H=H';
%Calculate output weights
opts.tol = 1e-9;
opts.issym=1;
opts.disp = 0;
if  (paras.NumHiddenNeuron<N)
    A=eye(paras.NumHiddenNeuron)+paras.lambda*H'*L*H;
    B=H'*H;
    [E,V] = eigs(A,B,paras.NE+1,'sm',opts);
    [~,idx]=sort(diag(V));
    elmModel.OutputWeight=E(:,idx(2:end));
    norm_term=H*E(:,idx(2:end));
    elmModel.OutputWeight=bsxfun(@times,E(:,idx(2:end)),sqrt(1./sum(
       norm_term.*norm_term)));
else
    B=H*H';
    A=eye(N)+paras.lambda*L*B;
    [E,V] = eigs(A,B,paras.NE+1,'sm',opts);
    [~,idx]=sort(diag(V));
    norm_term=B*E(:,idx(2:end));
    elmModel.OutputWeight=bsxfun(@times,H'*E(:,idx(2:end)),sqrt(1./sum(
       norm_term.*norm_term)));
end

Embed=H*elmModel.OutputWeight;

if ~paras.NormalizeOutput
    elmModel.Embed=Embed;
else
    elmModel.Embed=bsxfun(@times,Embed,1./sqrt(sum(Embed.*Embed,2)));
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [label, center, bCon, sumD, Dist] = litekmeans(X, k, varargin)
% label = LITEKMEANS(X, K) partitions the points in the N-by-D data
%               matrix X into K clusters.This partition minimizes the
%               sum,over all clusters, of the within-cluster sums of
%               point-to- cluster-centroid distances.  Rows of X
%               correspond to points, columns correspond to variables.
%               KMEANS returns  an N-by-1 vector label containing the
%               cluster indices of each point.
%
% [label, center] = LITEKMEANS(X, K) returns the K cluster centroid
%                               locations in the K-by-D matrix center.
%
% [label, center, bCon] = LITEKMEANS(X, K) returns the bool value bCon
%                           to indicate whether the iteration is
%                           converged.
%
% [label, center, bCon, SUMD] = LITEKMEANS(X, K) returns the
%                                 within-cluster sums of point-to-
%                                 centroid distances in the 1-by-K
%                                 vector sumD.
%
% [label,center,bCon,SUMD,Dist]=LITEKMEANS(X,K) returns distances
%                                    from each point to every centroid
%                                    in the N-by-K matrix Dist.
%
%   [ ... ] = LITEKMEANS(..., 'PARAM1',val1, 'PARAM2',val2, ...)
%              specifies optional parameter name/value pairs to control
%              the iterative algorithm  used by KMEANS.  Parameters are:
%
% 'Distance' - Distance measure, in D-dimensional space, that KMEANS
%               should minimize with respect to.  Choices are:
%               {'sqEuclidean'} - Squared Euclidean distance (the
%                                 default)
%                 'cosine'      - One minus the cosine of the included
%                                 angle  between points (treated as
%                                 vectors). Each row of X SHOULD be
%                                 normalized to unit. If  the intial
%                                 center matrix is provided, it SHOULD
%                                 also be normalized.
%
% 'Start' - Method used to choose initial cluster centroid positions,
%           sometimes known as "seeds".  Choices are:
%           {'sample'}  - Select K observations from X at random (the
%                         default)
%            'cluster' - Perform preliminary clustering phase on random
%                        10% subsample of X.  This preliminary phase
%                        is  itself initialized using 'sample'. An
%                        additional parameter clusterMaxIter can be
%                        used to control the maximum number of
%                        iterations in each preliminary clustering
%                        problem.
%           matrix    - A K-by-D matrix of starting locations; or a
%                       K-by-1 indicate vector indicating which K
%                       points in X should be used as the initial
```

```
%                           center.  In this case,you can pass in []
%                           for K, and KMEANS infers K from the
%                           first dimension of the matrix.
%
% 'MaxIter'    - Maximum number of iterations allowed.  Default is 100.
%
% 'Replicates' - Number of times to repeat the clustering, each with a
%                new set of initial centroids. Default is 1. If the
%                initial centroids are provided, the replicate will be
%                automatically set to be 1.
%
% 'clusterMaxIter' - Only useful when 'Start' is 'cluster'. Maximum
%                          number of iterations of the preliminary
%                          clustering phase. Default is 10.
% Examples:
%     fea = rand(500,10);
%     [label, center] = litekmeans(fea, 5, 'MaxIter', 50);
%
%     [label,center]= litekmeans(fea,5,'MaxIter',50,'Replicates',10);
%
%     [label,center,bCon,sumD,Dist] = litekmeans(fea,5,'MaxIter',50);
%     TSD = sum(sumD);
%
%     initcenter = rand(5,10);
%     [label,center]= litekmeans(fea,5,'MaxIter',50,'Start',initcenter)
%
%     idx=randperm(500);
%     [label,center] = litekmeans(fea,5,'MaxIter',50,'Start',idx(1:5));

if nargin < 2
    error('litekmeans:TooFewInputs','At least two input arguments
       required.');
end

[N, D] = size(X);

pnames = {'distance' 'start'   'maxiter'  'replicates' 'onlinephase' '
   clustermaxiter'};
dflts =  {'sqeuclidean' 'sample'   []     []     'off'     [] };
[eid,errmsg,distance,start,maxit,reps,online,clustermaxit] = getargs(
   pnames, dflts, varargin{:});

if ~isempty(eid)
    error(sprintf('litekmeans:%s',eid),errmsg);
end

if ischar(distance)
    distNames = {'sqeuclidean','cosine'};
    j = strcmpi(distance, distNames);
    j = find(j);
    if length(j) > 1
        error('litekmeans:AmbiguousDistance', ...
            'Ambiguous ''Distance'' parameter value:  %s.', distance);
    elseif isempty(j)
        error('litekmeans:UnknownDistance', ...
            'Unknown ''Distance'' parameter value:  %s.', distance);
```

```
    end
    distance = distNames{j};
else
    error('litekmeans:InvalidDistance', ...
        'The ''Distance'' parameter value must be a string.');
end

center = [];
if ischar(start)
    startNames = {'sample','cluster'};
    j = find(strncmpi(start,startNames,length(start)));
    if length(j) > 1
        error(message('litekmeans:AmbiguousStart', start));
    elseif isempty(j)
        error(message('litekmeans:UnknownStart', start));
    elseif isempty(k)
        error('litekmeans:MissingK', ...
            'You must specify the number of clusters, K.');
    end
    if j == 2
        if floor(.1*n) < 5*k
            j = 1;
        end
    end
    start = startNames{j};
elseif isnumeric(start)
    if size(start,2) == p
        center = start;
    elseif (size(start,2) == 1 || size(start,1) == 1)
        center = X(start,:);
    else
        error('litekmeans:MisshapedStart', ...
            'The ''Start'' matrix must have the same number of columns
               as X.');
    end
    if isempty(k)
        k = size(center,1);
    elseif (k ~= size(center,1))
        error('litekmeans:MisshapedStart', ...
            'The ''Start'' matrix must have K rows.');
    end
    start = 'numeric';
else
    error('litekmeans:InvalidStart', ...
        'The ''Start'' parameter value must be a string or a numeric
           matrix or array.');
end

% The maximum iteration number is default 100
if isempty(maxit)
    maxit = 100;
end

%The maximum iteration number for preliminary clustering phase on
% random,10% subsamples is default 10
if isempty(clustermaxit)
```

```
    clustermaxit = 10;
end

% Assume one replicate
if isempty(reps) || ~isempty(center)
    reps = 1;
end

if ~(isscalar(k) && isnumeric(k) && isreal(k) && k > 0 && (round(k)==k)
   )
    error('litekmeans:InvalidK', ...
        'X must be a positive integer value.');
elseif n < k
    error('litekmeans:TooManyClusters', ...
        'X must have more rows than the number of clusters.');
end

bestlabel = [];
sumD = zeros(1,k);
bCon = false;

for t=1:reps
    switch start
        case 'sample'
            center = X(randsample(n,k),:);
        case 'cluster'
            Xsubset = X(randsample(n,floor(.1*n)),:);
            [dump, center] = litekmeans(Xsubset, k, varargin{:}, 'start
               ','sample', 'replicates',1 ,'MaxIter',clustermaxit);
        case 'numeric'
    end

    last = 0;label=1;
    it=0;

    switch distance
        case 'sqeuclidean'
            while any(label ~= last) && it<maxit
                last = label;

                bb = full(sum(center.*center,2)');
                ab = full(X*center');
                D = bb(ones(1,n),:) - 2*ab;

                [val,label] = min(D,[],2); %assign samples to the
                   nearest centers
                ll = unique(label);
                if length(ll) < k
                    %disp([num2str(k-length(ll)),' clusters dropped at
                       iter ',num2str(it)]);
                    missCluster = 1:k;
                    missCluster(ll) = [];
                    missNum = length(missCluster);

                    aa = sum(X.*X,2);
                    val = aa + val;
```

```
            [dump,idx] = sort(val,1,'descend');
            label(idx(1:missNum)) = missCluster;
        end
        E = sparse(1:n,label,1,n,k,n);  %transform label into
            indicator matrix
        center = full((E*spdiags(1./sum(E,1)',0,k,k))'*X);
        %compute center of each cluster
        it=it+1;
    end
    if it<maxit
        bCon = true;
    end
    if isempty(bestlabel)
        bestlabel = label;
        bestcenter = center;
        if reps>1
            if it>=maxit
                aa = full(sum(X.*X,2));
                bb = full(sum(center.*center,2));
                ab = full(X*center');
                D = bsxfun(@plus,aa,bb') - 2*ab;
                D(D<0) = 0;
            else
                aa = full(sum(X.*X,2));
                D = aa(:,ones(1,k)) + D;
                D(D<0) = 0;
            end
            D = sqrt(D);
            for j = 1:k
                sumD(j) = sum(D(label==j,j));
            end
            bestsumD = sumD;
            bestD = D;
        end
    else
        if it>=maxit
            aa = full(sum(X.*X,2));
            bb = full(sum(center.*center,2));
            ab = full(X*center');
            D = bsxfun(@plus,aa,bb') - 2*ab;
            D(D<0) = 0;
        else
            aa = full(sum(X.*X,2));
            D = aa(:,ones(1,k)) + D;
            D(D<0) = 0;
        end
        D = sqrt(D);
        for j = 1:k
            sumD(j) = sum(D(label==j,j));
        end
        if sum(sumD) < sum(bestsumD)
            bestlabel = label;
            bestcenter = center;
            bestsumD = sumD;
            bestD = D;
        end
```

```
        end
    case 'cosine'
        while any(label ~= last) && it<maxit
            last = label;
            W=full(X*center');
            [val,label] = max(W,[],2);
            %assign samples to the nearest centers
            ll = unique(label);
            if length(ll) < k
                missCluster = 1:k;
                missCluster(ll) = [];
                missNum = length(missCluster);
                [dump,idx] = sort(val);
                label(idx(1:missNum)) = missCluster;
            end
            E = sparse(1:n,label,1,n,k,n);
           %transform label into indicator matrix
            center = full((E*spdiags(1./sum(E,1)',0,k,k))'*X);
           %compute center of each cluster
            centernorm = sqrt(sum(center.^2, 2));
            center = center ./ centernorm(:,ones(1,p));
            it=it+1;
        end
        if it<maxit
            bCon = true;
        end
        if isempty(bestlabel)
            bestlabel = label;
            bestcenter = center;
            if reps>1
                if any(label ~= last)
                    W=full(X*center');
                end
                D = 1-W;
                for j = 1:k
                    sumD(j) = sum(D(label==j,j));
                end
                bestsumD = sumD;
                bestD = D;
            end
        else
            if any(label ~= last)
                W=full(X*center');
            end
            D = 1-W;
            for j = 1:k
                sumD(j) = sum(D(label==j,j));
            end
            if sum(sumD) < sum(bestsumD)
                bestlabel = label;
                bestcenter = center;
                bestsumD = sumD;
                bestD = D;
            end
        end
end
```

```
end

label = bestlabel;
center = bestcenter;
if reps>1
    sumD = bestsumD;
    D = bestD;
elseif nargout > 3
    switch distance
        case 'sqeuclidean'
            if it>=maxit
                aa = full(sum(X.*X,2));
                bb = full(sum(center.*center,2));
                ab = full(X*center');
                D = bsxfun(@plus,aa,bb') - 2*ab;
                D(D<0) = 0;
            else
                aa = full(sum(X.*X,2));
                D = aa(:,ones(1,k)) + D;
                D(D<0) = 0;
            end
            D = sqrt(D);
        case 'cosine'
            if it>=maxit
                W=full(X*center');
            end
            D = 1-W;
    end
    for j = 1:k
        sumD(j) = sum(D(label==j,j));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [eid,emsg,varargout]=getargs(pnames,dflts,varargin)
%GETARGS Process parameter name/value pairs
%   [EID,EMSG,A,B,...]=GETARGS(PNAMES,DFLTS,'NAME1',VAL1,'NAME2',VAL2
    ,...)
%  accepts a cell array PNAMES of valid parameter names, a cell array
    DFLTS of default values for the parameters named in PNAMES, and
    additional parameter name/value pairs.  Returns parameter values A,
    B,... in the same order as the names in PNAMES. Outputs
    corresponding to entries in PNAMES that are not specified in the
    name/value pairs are set to the corresponding value from DFLTS.If
    nargout is equal to length(PNAMES)+1,then unrecognized name/value
    pairs are an error.If nargout is equal to length(PNAMES)+2,then all
     unrecognized name/value pairs are returned in a single cell array
    following any other outputs.
%
%  EID and EMSG are empty if the arguments are valid.  If an error
    occurs, EMSG is the text of an error message and EID is the final
    component  of an error message id.  GETARGS does not actually
    throwany errors,  but rather returns EID and EMSG so that the
    caller may throw the error.  Outputs will be partially processed
    after an error occurs.
%
%   Example:
```

```
%       pnames = {'color' 'linestyle', 'linewidth'}
%       dflts  = {    'r'         '_'          '1'}
%       varargin = {{'linew' 2 'nonesuch' [1 2 3] 'linestyle' ':'}
%       [eid,emsg,c,ls,lw]=statgetargs(pnames,dflts,varargin{:}) %error
%       [eid,emsg,c,ls,lw,ur]=statgetargs(pnames,dflts,varargin{:}) %ok

% Initialize some variables
emsg = '';
eid = '';
nparams = length(pnames);
varargout = dflts;
unrecog = {};
nargs = length(varargin);

% Must have name/value pairs
if mod(nargs,2)~=0
    eid = 'WrongNumberArgs';
    emsg = 'Wrong number of arguments.';
else
    % Process name/value pairs
    for j=1:2:nargs
        pname = varargin{j};
        if ~ischar(pname)
            eid = 'BadParamName';
            emsg = 'Parameter name must be text.';
            break;
        end
        i = strcmpi(pname,pnames);
        i = find(i);
        if isempty(i)
            % if they've asked to get back unrecognized names/values,
               add this
            % one to the list
            if nargout > nparams+2
                unrecog((end+1):(end+2)) = {varargin{j} varargin{j+1}};
                % otherwise, it's an error
            else
                eid = 'BadParamName';
                emsg = sprintf('Invalid parameter name:  %s.',pname);
                break;
            end
        elseif length(i)>1
            eid = 'BadParamName';
            emsg = sprintf('Ambiguous parameter name:  %s.',pname);
            break;
        else
            varargout{i} = varargin{j+1};
        end
    end
end

varargout{nparams+1} = unrecog;
```

واژه‌نامه فارسی به انگلیسی

اکیدا غالب قطری . . . . . Strictly Diagonally Dominant

تابع پایه شعاعی . . . . . Radial Basis Function

تجزیه QR . . . . . QR Factorization

نشاندن . . . . . Embedding

تقریب سراسری . . . . . Universal Approximation

خوشه‌بندی . . . . . Clustering

داده آموزشی . . . . . Training Data

رگرسیون . . . . . Regression

شبکه‌های عصبی پیشخور . . . . . Feedforward Neural Networks

طبقه‌بندی . . . . . Classification

نامنفرد . . . . . Non-Singular

لایه پنهان . . . . . Hidden Layer

لایه خروجی . . . . . Output Layer

لایه ورودی . . . . . Input Layer

ماتریس لاپلاسین . . . . . Laplacian Matrix

مدل خطی . . . . . Linear Model

مرتبه ستونی کامل . . . . . Full Rank Column

معکوس‌پذیر . . . . . Invertible

نورون‌های پنهان تصادفی . . . . . Random Hidden Nodes

یادگیری ماشین . . . . . Machine Learning

واژه‌نامه انگلیسی به فارسی

طبقه‌بندی . . . . . Classification

خوشه‌بندی . . . . . Clustering

نشاندن . . . . . Embedding

شبکه‌های عصبی پیشخور . . . . . Feedforward Neural Networks

مرتبه ستونی کامل . . . . . Full Rank Column

لایه پنهان . . . . . Hidden Layer

لایه ورودی . . . . . Input Layer

معکوس‌پذیر . . . . . Invertible

ماتریس لاپلاسین . . . . . Laplacian Matrix

مدل خطی . . . . . Linear Model

یادگیری ماشین . . . . . Machine Learning

نامنفرد . . . . . Non-Singular

لایه خروجی . . . . . Output Layer

تجزیه QR . . . . . QR Factorization

تابع پایه شعاعی . . . . . Radial Basis Function

نورون‌های پنهان تصادفی . . . . . Random Hidden Nodes

رگرسیون . . . . . Regression

اکیدا غالب قطری . . . . . Strictly Diagonally Dominant

داده آموزشی . . . . . Training Data

تقریب سراسری . . . . . Universal Approximation

Hakim Sabzevari University

An Outline of MSc. Thesis

Surname:Khoorsandi	Name:Sakineh	Student No.:9313137053

Supervisor: Dr. Mahmood Amintoosi

Advisor: Dr. Mehdi Zaferanieh

Faculty of Mathematics and Computer Science

Program: Decision Science and Knowledge Engineering

Title of thesis: Extreme Learning Machines

Keywords:Single-Layer Feedforward Networks,ExtremeLlearning Machine,Universal Approximation Capability,Classification Capability,Regression

Abstract: The slow speed of feedforward neural networks learning and repeatedly adjust the parameters has been a major bottleneck in their applications for past decades. The extreme learning machine, that is proposed by Huang, is designed based on ”generalized“ single-hidden layer feedforward neural networks , which randomly choose the parameters of hidden nodes as well as the output weights gotten analytically. However, the learning time of extreme learning machine is mainly spent on calculating the Moore-Penrose generalized inverse matrices of the hidden layer output matrix. Extreme learning machines is efficient and effective learning paradigm for pattern classification and regression. Extreme learning machines has become a hot area of research over the past years, which is attributed to the growing research activities researchers around the world .Theory and experiments prove that the extreme learning machine has simple structure as well as powerful approximation capability. The aim of this thesis, introducing this new method, capabilities, features and its training methods and review ELMs for both semi-supervised and unsupervised tasks based on the manifold regularization. Moreover, it is shown that all the supervised, semi-supervised and unsupervised ELMs can actually be put into a unified framework.

Hakim Sabzevari University
Faculty of Mathematics and Computer Science

A Thesis Submitted in Partial Fulfilment of the Requirement for the Degree of Master of Science in Decision Science and Knowledge Engineering

Extreme Learning Machines

Supervisor:
Dr. Mahmood Amintoosi

Advisor:
Dr. Mehdi Zaferanieh

By:
Sakineh Khoorsandi

Feb. 2017