
بسم الله الرحمن الرحيم

دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش پروژه

عنوان:

بررسی متدهای مختلف برای استخراج ویژگی مناسب

نگارش :

محمد کلباسی

استاد:

دکتر هدی محمدزاده

فهرست

1 مقدمه

1-1 مقدمه

در این گزارش ما روش‌های مختلف انتخاب ژن‌های مناسب برای طبقه‌بندی ارایه‌های بیان ژن سرطانی از غیر سرطانی را انجام بدهیم، دقیقا مانند انتخاب ویژگی که ویژگی همان بیان هر ژن است، برای این امر ما دو مقاله را شبیه سازی کرده‌ایم([2],[1]) علاوه برآن، ما روش های پایه دیگری را نیز پیاده کرده و مقایسه کرده ایم، این روش ها شامل متدهای کاهش بعد ویژگی مانند PCA,NMF[1],Autoencoder استفاده میکنیم و همچنین از کتابخانه آماده پایتون نیز استفاده میکنیم تا با دو روش chi2 , MRMR[2] مقایسه کنیم، از طرف دیگر روش backward نیز تست می‌شود. دقت کنید برای مقایسه روش‌ها ما زمان اجرا شدن را نیز مقایسه کرده‌ایم که ممکن است با توجه به سخت افزار فرق کند (cpu استفاده شده ما core i7 7700 میباشد در صورت مشابهت باید نتایج نزدیک باشند) ولی مقدار مهم، مقایسه روش ها با یکدیگر است و خود زمان دقیق آنقدر اهمیت ندارد.

1-2 معرفی پایگاه دادگان استفاده شده

مطابق مقالات شبیه سازی شده، ما از دادگان سرطانی و سالم در مقابل هم استفاده میکنیم، در جدول زیر پایگاه دادگان استفاده شده آورده شده است:

[1] Non negative matrix factorization
[2] Maximum relevance minimum redundancy

نوع سرطان	داده بیمار	داده سالم
leukaemia	23	49
lymphomadata	20	42
lung	31	150
breast	30	30

دو پایگاه داده اول به خوبی قابل جدایی پذیرهستند(با انتخاب ویژگی مناسب) ولی دو پایگاه اخر اینگونه نیست و در پایگاه داده breast نیز ما داده گم شده (که با مقدار صفر مقدار دهی شده) بسیار زیاد داریم.در کد ما دیتالودری نوشتیم که با دادن نام هرکدام داده ها لود شود، فقط لطفا فایل های دادگان را در فولدری که کد قرار دارد قرار دهیم و دایرکتوری را آن قرار بدهید تا کد به درستی اجرا بشود.

2 بررسی مقالات

2-1 استخراج ویژگی با استفاده از **double rbf kernel**

به صورت کلی، ما داده بیان ژن را در یک آرایه به صورت:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1l} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nl} \end{bmatrix}$$

ذخیره می‌کنیم، برای بررسی داده‌ها در شرایط یکسان دادگان به صورت z-norm نرمالیزه می‌شوند:

$$Z = \frac{(x-\mu)}{\sigma} \quad (1-1)$$

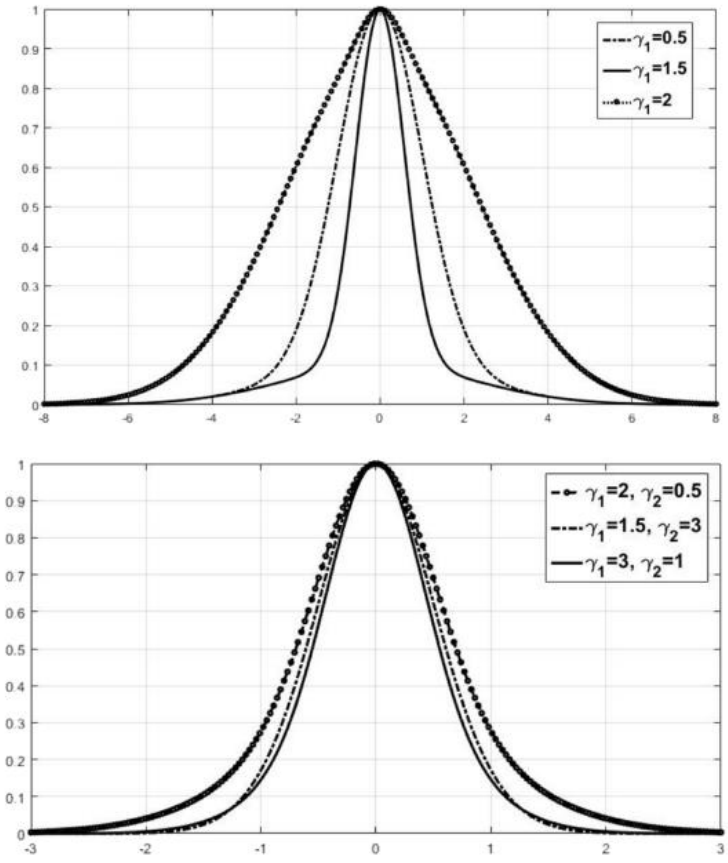
حال اگر بخواهیم کلیت روش را بگوییم، این یک روش انتخاب ویژگی به فرم filter محور(یعنی از خود طبقه بند استفاده نمیشود، برعکس متدهای مبتنی بر wrapper) که در آن ابتدا مراکز دسته‌های کلاس های مختلف استخراج میشود، سپس فاصله هر فرد از مراکز مربوطه به خود(براساس کلاس) محاسبه می‌شود، حال ما میخواهیم وزن‌هایی به فرم w که در هر ویژگی (بخوانید بیان ژن) و با بهینه کردن تابع هزینه که در جلوتر تعریف میکنیم، میتوانیم ما براساس توزیع w های تولید شده ما درنهایت با قواعد دلخواه میتوانیم ویژگی‌های مورد نظر را انتخاب کنیم.

برای رابطه مرکز هر دسته داریم:

$$\mathbf{v_{ik}} = \frac{1}{|C_i|} \sum_{X_j \in C_i} x_{jk} \qquad (2\text{-}1)$$

که در رابطه i کلاس و k عنصر بیان ژن را بیان میکند.

حال ما میخواهیم بیاییم و از double rbf kernel استفاده کنیم، در شکل زیر تفاوت کرنل rbf عادی و حالت گفته شده را رسم میکنیم:



شکل یک: مقایسه کرنل

و با بررسی فرمول آن داریم:

$$K_{\gamma_1\gamma_2}(x, x_j) = ce^{-\gamma_1\|x-x_i\|^2} + (1-c)e^{-\gamma_2\|x-x_i\|^2} \quad (\gamma_1 > 0, \gamma_2 > 0) \tag{3-1}$$

در حالت دوگانه، ما همواره میتوانیم به رفتار یک کرنل rbf تنهایی برسیم (ضریب c بین صفر و یک است) ولی قدرت بیشتری که در این حالت داریم این است که در حالت کرنل ساده، براساس واریانسی که تعریف میکنیم یا شدت کاهش نسبتا به شدت زیاد است یعنی اگر خیلی نزدیک باشد تنها امتیاز مناسبی نسبت داده می‌شود، ولی اگر مقداری دور شود فاصله (که مثلا میتواند به دلایل مختلفی از جمله اضافه شدن نویز رخ بدهد) ممکن است امتیاز بسیار کمی نسبت بدهیم، از طرف دیگر در نظر گرفتن واریانس بالا باعث میشود وزن برای رفتار های نویزی (بدون داشتن الگو و وابسته نبودن) نیز نسبتا زیاد باشد، ولی

اگر ما تعداد بیشتری گاوسی با هم دیگر جمع کنیم میتوانیم رفتار پیچیده تری تولید کنیم، به طوری که در مقادیر نزدیک به هم تا حدی معقول مقدار گاوسی زیاد باشد ولی در مقادیر دور کم، به نوعی با اضافه کردن پارامتر ما رفتار پیچیده تری تولید میکنیم، هرچه مقادیر کرنل های گاوسی اضافه شده بیشتر باشد میتوان با تنظیم هایپر پارامترها رفتار پیچیده تر براساس نیاز تولید کرد، در مقاله استفاده از دو گاوسی به صورت گفته شده پیشنهاد شده، دقت کنید هنگام جمع حتما باید تضمین کنیم فرم کلی به صورت احتمالی باقی میماند پس برای همین نیز ضرب c و (1-c) ضرب شده است.

برای هایپرپارامترهای مقاله ما 0.5 و 2 و 1.5 و 3 را بررسی کرده ایم، برای دادگان جدایی پذیر حالت دوم بهتر است و دادگان نسبتا رفتار سخت تری دارن رفتار بهتری برای مقادیر اولیه معرفی شده بدست می اید، در کد برای شبیه سازی کرنل ما تابع rbf_kernel را نوشتیم که دو مد گفته شده را دارد، یعنی اگر متد یک باشد کرنل rbf ساده انجام میشود و اگر حالت 2 را انتخاب کنید به حالت دوگانه است و براساس آن فاصله محاسبه میشود.

حال که کرنل را داریم، ما رابطه فاصله را براساس کرنل مینویسیم:

$$K(x_k, x_l) = \Phi(x_k)^T \Phi(x_l) \tag{4-1}$$

$$\begin{aligned} \|\Phi(x_k)-\Phi(x_l)\|^2 &= (\Phi(x_k)-\Phi(x_l))^T(\Phi(x_k)-\Phi(x_l) \\ &= K(x_k,x_k)-2K(x_k,x_l)+K(x_l,x_l) \end{aligned} \tag{5-1}$$

$$\begin{aligned} \phi^2(x_j, v_i) &= \sum_{k=1}^{l} \|\Phi(x_{jk})-\Phi(v_{ik})\|^2 \\ &= \sum_{k=1}^{l} (K(x_{jk},x_{jk})-2K(x_{jk},v_{ik})+K(v_{ik},v_{ik})) \end{aligned} \tag{6-1}$$

براساس رابطه 4 و 5 ما فاصله را برحسب کرنل میتوانیم بنویسیم، هرچه شباهت بیشتر فاصله کمتر، و در رابطه 6 ما جمع رابطه فاصله بیان ژن فرد با مرکز دسته (همانطور که گفتم مراکز با v نشان داده میشود) حال ما تابع بهینه سازی را تعریف میکنیم:

$$\begin{aligned} J &= \sum_{i=1}^{C}\sum_{x_j\in C_i}\phi^2(X_j, V_i) + \delta\sum_{k=1}^{l} W_k^2 \\ &= \sum_{i=1}^{C}\sum_{x_j\in C_i}\sum_{k=1}^{l} W_k\|\Phi(X_{jk})-\Phi(V_{ik})\|^2 \\ &+\delta\sum_{k=1}^{l} W_k^2 \end{aligned}$$

(7-1)

اگر بخواهیم رابطه گفته شده را تفسیر کنیم، این است که برای هر بیان ژن (هر ویژگی) یک W تعریف میشود(به صورت W_k) ما میخواهیم رابطه گفته شده را مینیمم کنیم، پس به ژنهایی که بیشترین نزدیکی را دارند وزن بیشتری نسبت میدهیم(و معیار نزدیکی را نیز براساس کرنل گفته شده در نظر میگیریم) اگر ترم نرمالیزاسیون نگذاریم، کمترین فاصله وزن یک میگیرد و بقیه وزن ها صفر میشود و رسما نمیتوان از آن برای انتخاب ویژگی استفاده کرد، برای همین ترم این امر که نرم دو وزن ها نیز مینیمم شود با وزنی اضافه میشود.

حال اگر رابطه بالا را بهینه کنیم داریم:

$$J(w_k,\lambda)=\sum_{i=1}^{C}\sum_{x_j\in C_i}\phi^2(x_j,v_i)+\delta\sum_{k=1}^{l}w_k^2-\lambda\left(\sum_{k=1}^{l}w_k-1\right) \tag{8-1}$$

که در رابطه بالا تنها لاگرانژ نوشتیم (با شرط مجموع وزن ها برابر یک است) اگر حل را جلو ببریم داریم:

$$\begin{cases}\dfrac{\partial J(w_k,\lambda)}{\partial\lambda}=\sum_{k=1}^{l}w_k-1\\ \dfrac{\partial J(w_k,\lambda)}{\partial w_k}=\sum_{i=1}^{C}\sum_{x_j\in C_i}\left\|\Phi(x_{jk})-\Phi(v_{ik})\right\|^2+2\delta w_k-\lambda\end{cases} \tag{9-1}$$

و در نهایت داریم:

$$w_k=\frac{1}{l}+\frac{1}{2\delta}\sum_{i=1}^{C}\sum_{x_j\in C_i}\left(\frac{\sum_{i=1}^{C}\sum_{x_j\in C_i}\left\|\Phi(x_{jk})-\Phi(v_{ik})\right\|^2}{l}-\left\|\Phi(x_{jk})-\Phi(v_{ik})\right\|^2\right) \tag{10-1}$$

برای ضریب دلتا، که دو ترم بین تابع هزینه را کنترل میکند به صورت زیر عمل میکنیم:

$$\delta^{(t)} = \alpha \frac{\sum_{i=1}^{C} \sum_{x_j \in C_i} \sum_{k=1}^{l} w_k^{(t-1)} \left\| \Phi(x_{jk}) - \Phi(v_{ik}) \right\|^2}{\sum_{k=1}^{l} \left(w_k^{(t-1)} \right)^2} \qquad (11\text{-}1)$$

براساس این رابطه، ما همواره میتوانیم نسبت دو ترم را کنترل کنیم به گونه ای که در مراحل بهینه سازی به این سمت متمایل نشویم که یک ترم بر ترم دیگر برتری داشته باشد، برای همین نیز مقاله پیشنهاد داده است که الفا مقدار آن یک باشد.

ما تابع cost_calc,delta_calculator و w_update را براساس همین موارد مینویسیم و الگوریتم زیر را برای بدست آوردن وزن ها استفاده میکنیم:

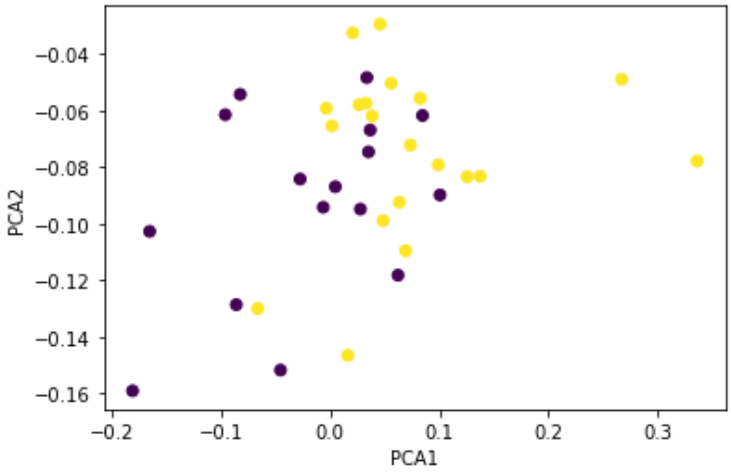
1-ابتدا به همه وزن‌ها مقدار $\frac{1}{l}$ نسبت میدهیم که L همان تعداد ژن های استفاده شده است.

2-مقدار دلتا را بدست می‌آوریم.

3-وزن ها را بروز میکنیم

4- بررسی میکنیم ایا اختلاف تابع هزینه ها در گام جدید با گام قبلی از ترشلدی که ما مشخیص میکنیم کمتر شده است یا نه، اگر شده باشد فرایند تمام و وزن ها بدست آمده و در غیر اینصورت به گام دو برگرد.

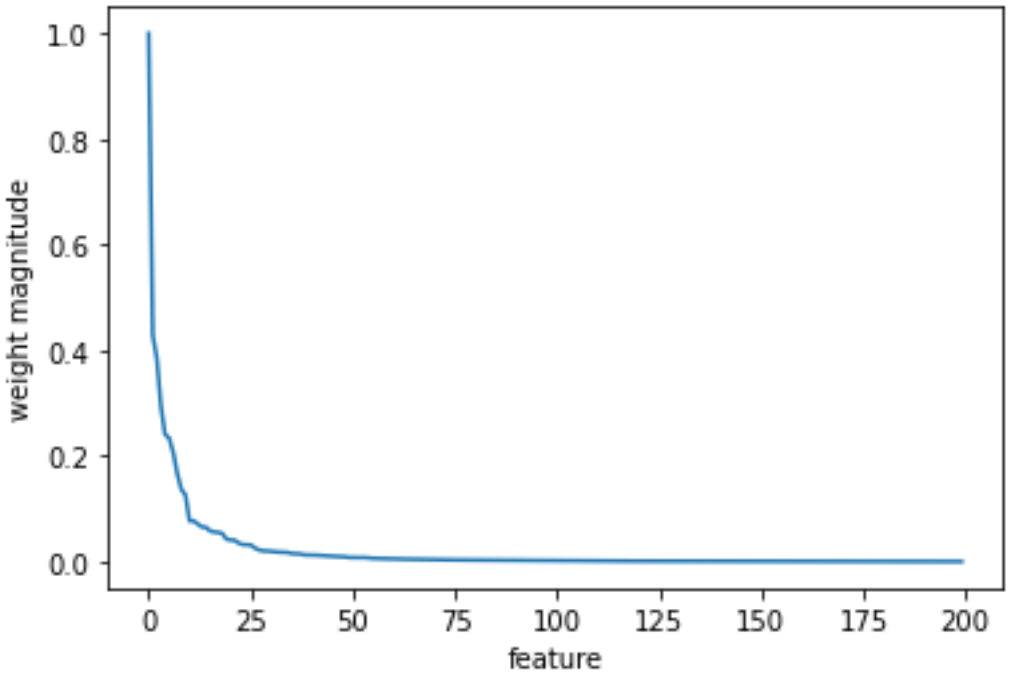
همانطور که از رابطه مشخص است در اینجا به اینصورت است که انگار ما داریم فاصله از مرکز هر دسته را بهینه میکنیم، به طور مثال برای پایگاه داده سرطان شش استفاده میکنم و pca ان را محاسبه میکنیم و رسم میکنیم داریم:



شکل دو: تبدیل *PCA* دادگان سرطانی

همانطور که مشخص است فرض نزدیکی به مرکز میتواند تا حدی به ما کمک کند.

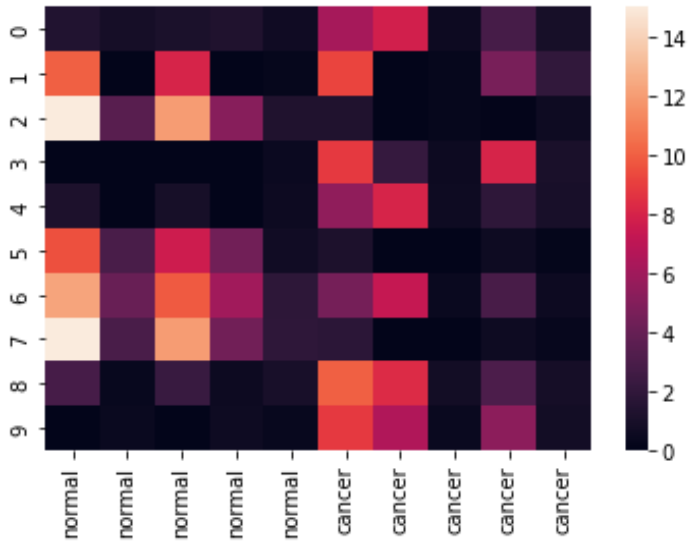
نکته مهم: در خود مقاله بعد از بهینه سازی از وزن‌های داده شده استفاده میکرد تا اینکه ویژگی انتخاب کند، ولی از نظر من این کار به صورت مستقیم کار درستی نیست، چون اگر به رابطه هزینه نگاه کنید جمع خطی وزن ها به علاوه مینیمم کردن نرم دو است، پس بدیهتا هر فاصله ای که کمتر باشد مقدار وزن بیشتر است و اصلا نیاز به طی کردن گام های گفته شده برای بهینه سازی وجود ندارد و میتوان مستقیم تنها از معیار فاصله استفاده کرد، ولی چیزی که من از ان نتیجه خوبی گرفتم، توزیع وزن های خروجی است یعنی به طور مثال برای دادگان سرطان *lymphoma* اگر بهینه سازی را انجام بدهیم، سپس W ها را به ترتیب سورت کرده و رسم کنیم داریم:



شکل 3

از این شکل(دقت کنید ماکسیمم وزن به یک مپ شده تا بتوانیم بهتر تغییرات را ببینیم وگرنه مجموع وزن ها باید یک شود) به طور واضح مشخص است که در حدود 50 ویژگی تنوع وزن ها به اندازه کافی است و بعد از ان دیگر تغییرات محسوس نیست، بنابر این بر اساس این میتوان معیار های مختلفی را تعیین کرد، ویژگی که از آن به بعد واریانس تغییرات W کمینه میشود را میتوان انتخاب کرد، یا اینکه ترشلد قرار داد، ما با بررسی به این نتیجه رسیدیم که اگر مقدار از $w[i]-w[i-1]$ را پیدا کنیم که دیگر در آن تغییرا خیلی کم است، مقدار i را به عنوان وزن های انتخاب شده برمیداریم و از آن استفاده میکنیم.

اگر به طور مثال هیت مپ ویژگی ها انتخاب شده برای سرطان *lymphoma* را رسم کنیم داریم(برای 4 نمونه سالم و مریض):



شکل 4

که مشاهده میکنید که در این حالت به صورت نسبتا خوبی جدا شده (ما فقط 10 ویژگی را رسم کرده ایم اگر بیشتر کنیم میتوانیم انتظار داشته باشیم کامل جدا شود)

تفاوت اصلی کرنل *rbf* عادی با *double rbf*چیست؟

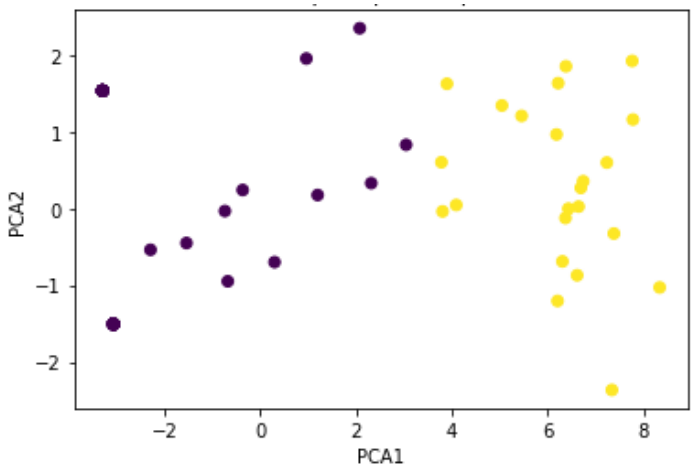
برای تست ما برای یک داده سرطانی حالت تنهایی و دوگانه را مقایسه کرده ایم، دو معیار ارزیابی داریم یک زمان همگرایی و دو دقت بدست آمده است، و همانطور که گفتیم از توزیع گفته شده استفاده میکنیم، مقدار ترشلد تمام کردن آموزش را 0 قرار میدهیم، درواقع در این حالت به صورت عادی متوقف نمیشود(یعنی باید حتما خطا در دو گام پشت هم یکی باشد که اتفاق نمیافتد) و برای متوقف شدن ما تعداد گام ماکس بروز رسانی را 10000 قرار میدهیم(یعنی بعد از 10000 گام متوقف شود) برای حالت تک گاوسی ما مقدار را 1 و 1.5 تست کردیم ولی برای کرنل دوگانه مقدار 0.5 و 2 (با عوض کردن متد از یک به دو در کد این کار به راحتی قابل انجام و مقایسه است) در هر دو حالت کرنل دوگانه با معیاری که برای انتخاب ویژگی است مقدار بهینه تری را معرفی میکند، یعنی همانطور که گفتیم مقداری در حدود 50 ولی برای کرنل یک گانه مقدار حدود 70 بدست می اید، دقت در هر دو حالت صد است ولی مشاهده میشود با کرنل دوگانه بهینه تر مینیمم ویژگی مورد نیاز برای دستیابی به دقت کامل را حساب کرده ایم.

حال ما یک اصلاحیه برروی روش نیز اعمال میکنیم، در خود مقاله معیار بهینه سازی براساس تنها فاصله با مرکز استفاده شده است ما پیشنهاد میکنیم که علاوه بر نزدیکی با مرکز دسته خود از مرکز دسته دیگر (در حالت باینری البته این مورد صادق است و چون ما همه دادگانمان دوکلاسه است میتوانیم از این روش استفاده کنیم) نسبت را قرار میدهیم یعنی به صورت:

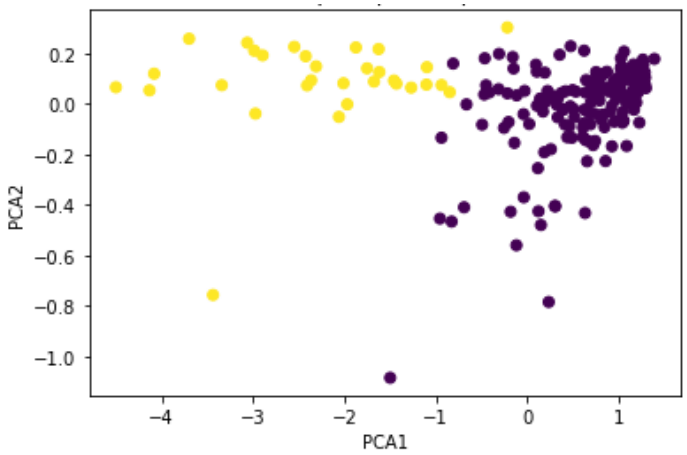
$$\varphi(x_i, v_{c1})_{x_{i\epsilon c1}} \rightarrow \frac{\varphi(x_i,v_{c1})_{x_{i\epsilon c1}}}{\varphi(x_i,v_{c2})_{x_{i\epsilon c1}}} \qquad (12\text{-}1)$$

در این حالت وقتی که ما تنها 25 ویژگی ابتدایی بر حسب وزن ها را در دو حالت مقایسه میکنیم(برای این امر که ببینیم چقدر بهینه انتخاب ویژگی داریم) دقت از 0.84 به 0.91 افزایش پیدا کرد، پس در بخش نتایج و مقایسه تنها از حالت پیشنهاد شده خودمان استفاده میکنیم.

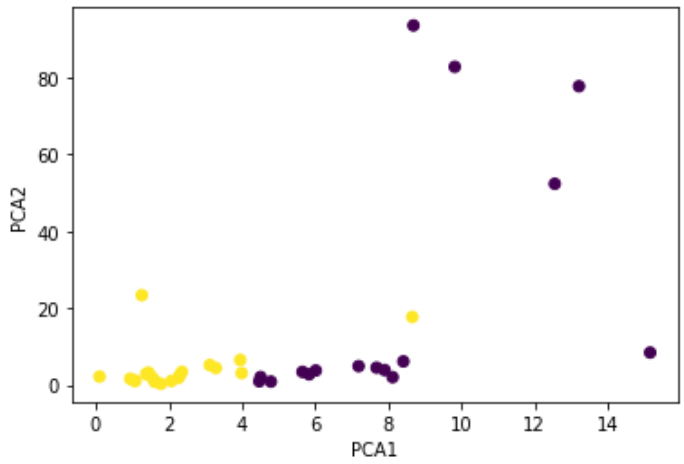
اگر بخواهیم نتایج *PCA* را برروی 4 داده مختلف رسم کنیم داریم:



برای *leukemia*



برای *lung*



برای *breast*

همانطور که از شکل ها نیز معلوم است، در دو حالت تفکیک پذیر به خوبی جدا شده است(*lymphoma* را در قبل تر نشان دادیم) ولی در بقیه حالات نیز تفکیک پذیری مناسب است، در حالت *breast* خیلی فرمت اسپارسی به خود وزن ها گرفته اند که احتمالا به علت بالا بودن تعداد دادگان گمشده است. ما دقت های نهایی را در فصل نتایج گزارش میکنیم.

2-2 کاهش ویژگی براساس **mutual information**

این فصل به صورت کلی برای این امر است که مقدمات مورد نیاز برای مقاله بعدی را توضیح دهیم، یکی از روش ها استخراج ویژگی استفاده از mutual information است در واقع ما میخواهیم اطلاعات متقابل بین برچسپ و ویژگی را بیشینه کنیم و به اینصورت جلو برویم، برای رابطه کلی داریم:

$$I(X;Y) = \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy, \tag{1-2}$$

و هدف ما این است که :

$$\tilde{S} = \arg\max_S I(\mathbf{X}_S; y), \qquad s.t. |S| = k, \tag{2-2}$$

یعنی ویژگی ها را به گونه ای اضافه کنیم که واقعا اطلاعات متقابل بالایی داشته باشند را انتخاب کنیم، اگر رابطه را بخواهیم ساده تر کنیم داریم:

$$S^{t-1} = \{x_{f_1}, \dots, x_{f_{t-1}}\} \longrightarrow \quad f_t = \arg\max_{i \notin S^{t-1}} I(\mathbf{X}_{S^{t-1} \cup i}; y).$$
$$f_t = \arg\max_{i \notin S^{t-1}} \underbrace{I(x_i;y)}_{\text{relevancy}} - \underbrace{[I(x_i; x_{S^{t-1}}) - I(x_i; x_{S^{t-1}}|y)]}_{\text{redundancy}}. \tag{3-2}$$

یعنی ما دسته ویژگی را داریم و میخواهیم ویژگی بعدی استفاده کنیم (به صورت forward) که مقدار بهینه باشد، یعنی بیشترین اطلاعات را نسبت به برچسپ داشته باشد، ولی کمترین اطلاعات اضافه را نسبت به برچسپ های قدیمی داشته باشد. با فرض مستقل بودن ویژگی ها میتوانیم روابط را به صورت ساده تر در نظر بگیریم یعنی داریم:

$$I(x_i; x_{S^{t-1}}) \approx \alpha \sum_{k=1}^{t-1} I(x_{f_k}; x_i),$$
$$I(x_i; x_{S^{t-1}}|y) \approx \beta \sum_{k=1}^{t-1} I(x_{f_k}; x_i|y).$$
$$f_t = \arg\max_{i \notin S^{t-1}} \underbrace{I(x_i; y)}_{\text{relevancy}} - \underbrace{\left[\alpha \sum_{k=1}^{t-1} I(x_{f_k}; x_i) - \beta \sum_{k=1}^{t-1} I(x_{f_k}; x_i|y)\right]}_{\text{redundancy}} \quad (4\text{-}2)$$

در این حالت ما تنها باید توزیع توام ویژگی ها با هم و با برچسپ ها را با هم داشته باشیم، حال به ازای مقادیر مختلف الفا و بتا تعاریف مختلفی داریم:

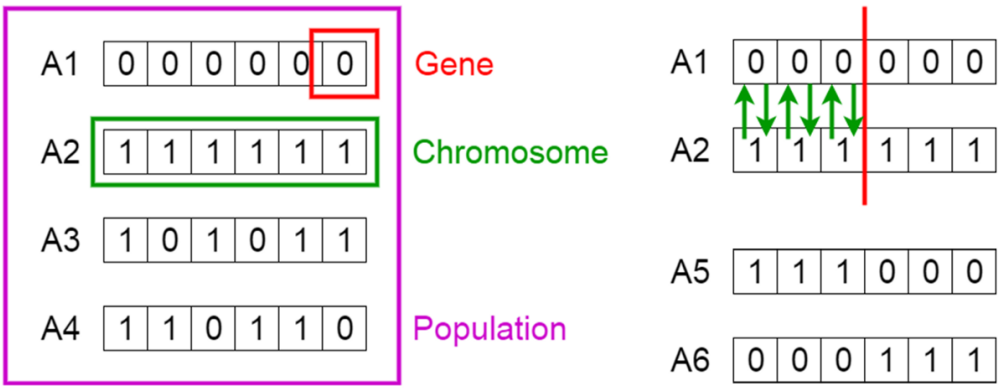
Joint mutual information: $\alpha = \frac{1}{t-1}, \beta = \frac{1}{t-1}$
Maximum relevancy minimum redundancy: $\alpha = \frac{1}{t-1}, \beta = 0$
Mutual information maximization: $\alpha = 0, \beta = 0$

یک حالت خیلی معروف که نتایج بسیار مناسبی دارد، mrmr است که برای آن ما از کتابخانه اماده استفاده میکنیم، در بخش نتایج نهایی نتایج این روش نیز گزارش شده است.

2-3 استخراج ویژگی براساس الگوریتم ژنتیک و اطلاعات متقابل

در این حالت که شبیه سازی مقاله [2] است. برای این روش ابتدا براساس اطلاعات متقابل ساده (هردو ضریب صفر، پس تنها ما باید برای هربرچسپ یک توزیع تخمین بزنیم که مثلا از حالت گاوسی میشود استفاده کرد) و از این برای یک فیلترینگ اولیه استفاده میشود که ویژگی های اولیه را استخراج میکند، یعنی در این روش فرض میشود که به صورت کلی این روش بهینه ویژگی های بد را حذف میکند، حال با این روش ویژگی ها را به 300 کاهش میدهیم، حال از الگوریتم ژنتیک استفاده میکنیم که در شکل زیر کلیت آن آورده شده است:

Genetic Algorithms



در واقع ما یک جمعیت اولیه همواره داریم، که تعداد ان را ما به دلخواه انتخاب میکنیم، هر ژن تولید شده به اندازه کل بیان ژن موجود است، صفر بودن آن یعنی اینکه ویژگی استفاده نمیشود و اگر یک باشد یعنی استفاده میشود، حال برای تولید جمعیت جدید ما دو فرایند تعریف میکنیم، یک mutation و دیگری cross-over، در mutation ما تعدادی از ژن های یک ژن را انتخاب کرده و قرینه میکنیم (صفر به یک و یک به صفر). حال بعد از این فرایند ما به تعدادی که میخواهیم بهترین نتایج و کد ژن های مربوطه را نگه میداریم و بقیه را حذف میکنیم، به اینصورت همواره روند بهبود دارد (چون فقط بهترین هارا نگه میداریم) و چون به صورت تعدادی نگه میداریم(یعنی به طور مثال ما همیشه 30 حالت را نگه میداریم، برای همین نیز حالت هایی که کامل بهینه نیستند نیز باقی میماند و این حالات ممکن است بعد از تغییر به minimum برسندglobal)

برای این موارد به تعریف امتیاز نیاز هست، ما میتوانیم نتایج خروجی طبقه بندهای مختلف را بررسی و معرفی کنیم ولی از انجا که هدف این است که مقالات مختلف را در شرایط یکسان بررسی کنیم، پس

از طبقه بند خطی svm استفاده میکنیم و دقت بدست آمده را امتیاز در نظر میگیریم، حال روابط زیر را برای احتمال cross over و mutation معرفی میکنیم:

$$P_C = \begin{cases} \frac{k_1(f' - f_{avg})}{f_{\max} - f_{avg}} & f' > f_{avg} \\ k2 & o.w \end{cases}$$

$$P_M = \begin{cases} \frac{k_3(f' - f_{avg})}{f_{\max} - f_{avg}} & f' > f_{avg} \\ k4 & o.w \end{cases}$$

که در روابط،f مقدار فیتنس و همان دقت بدست آمده براساس ویژگی های انتخاب شده است، منظور از f' جمعیت جدید ایجاد شده است و مقدار f_{max}, f_avg دقت فیتنس روی کل داده ها (داده های جمعیت اصلی و جهش یافته است، پس همواره $f' < f_{\max}$ است. مقادیر $k1$ تا $k4$ ثابت های ما هستند، در واقع براساس این روابط انگار اگر دقت از میانگین دقت ها بیشتر شود، ما با شدت بیشتری تشویق میکنیم که جهش (یا کراس اور) رخ دهد، مقادیر $k1$ تا $k4$ را براساس 0.1 و 0.01 یکی درمیان تعریف میکنیم.

برای به پایان رسیدن فرایند ما میتوانیم قاعده تعریف کنیم، قاعده ای که ما تعریف کرده ایم یا ماکسیمم تعداد نسل ها است(تغییر نسل ها) و یا اینکه به دقت مورد نظر برسیم، ما اصولا تعداد گام های مجاز را خیلی بالا قرار میدهیم و مقدار فیتنس را اصولا اینگونه قرار میدهیم که به دقتی بهتر از بقیه حالات برسیم. این موارد را در بخش نتایج بررسی میکنیم.

2-4 روش **backward svm**

در این روش از یک منطق ساده استفاده میشود، در svm خطی وقتی که ما اموزش میدهیم، کمترین وزن w (از نظر قدرمطلق) میتوان گفت کمترین اهمیت را دارد، حال ما به صورت عقب گرد میاییم و هربار مقدار ویژگی که کمترین وزن به آن نسبت داده شده است را حذف میکنیم حال براساس ویژگی های جدید یک بار دیگر svm را اموزش میدهیم و دوباره کم اهمیت ترین ویژگی براساس معیار گفته شده را حذف میکنیم تا اینکه به تعداد ویژگی مورد نظر برسیم.

دقت کنید چون به صورت تک به تک حذف میکنیم زمان اجرای این بخش طول میکشد.

3 روش های مبتنی بر کاهش ویژگی

3-1 مقدمه

در بخش قبل ما روش های مختلف انتخاب ویژگی را معرفی کرده ایم، در آن حالت هر ویژگی در واقع بیان یک ژن خاص است، و بنابراین هرکدام به تنهایی معنا دارند، ولی در این حالت ما از متدهای مختلف که ویژگی ها را به صورتی ترکیب میکند که بهینه نمایش دهد، از نظر مفهومی هر دوی انها کاری که انجام میدهند کاهش ابعاد است ولی روش های این بخش به صورت اولیه به اطلاعات بیشتری (اطلاعات کامل) نیاز دارند برای همین انتظار داریم این روش ها دقت بالاتری داشته باشند.

PCA 3-2

این روش به صورت کامل در خود درس توضیح داده شده است، تنها اگر بخواهیم جزییات پیاده سازی را بیان کنیم ما به صورت مستقیم ماتریس کرنل را حساب میکنیم، و ابتدا میانگین داده های ترین را حساب میکنیم و این را به صورت کلی روی بقیه بخش ها نیز استفاده میکنیم، براساس روابط ما بردار های PCA را استخراج میکنیم و براساس بردار های پایه و میانیگن، به تعدادی که میخواهیم بعد را کاهش میدهیم (با تصویر کردن روی بردار های PCA) حال از این کاهش بعد یافته در بخش های بعد استفاده میکنیم تا اینکه طبقه بندی صورت بگیرد.

NMF 3-3

روش NMF یا همان non negative matrix factorization براین اساس است که تجزیه ماتریس ورودی را به صورت:

$$X = WH$$

تجزیه میکنیم، خاصیت مهم آن است که W و H هردو به اینصورت هستند که مقادیر آن تنها مثبت هستند، و این خاصیت از انجا برای ما کمک کننده است که ارایه های بیان ژن به صورت ذاتی خود مقادیر مثبت تنها دارند، بنابراین میتواند به خوبی برای ما کار کند و روابط با به صورت درست استخراج کند، اگر ما فرض کنیم هر ستون برای یک فرد و هر سطر یک ویژگی است، ما اگر

$$pinv(W)X = H$$

را محاسبه کنیم که در این حالت H همان ماتریس کاهش ویژگی یافته است، بنابراین ما اگر شبه وارون W را در هر ورودی (چه ترین و چه تست)ضرب کنیم مقدار بردار ویژگی کاهش یافته بدست می اید. برای محاسبه ماتریس H و W ما از روش multiplicative استفاده کرده ایم که به نوعی همان

الگوریتم گرادیان کاهشی است که در آن استپ سایز اپدیت به گونه ای تعیین شده است که بروزرسانی وزن ها به صورت رابطه ساده بدست بیایید که فرم ان به صورت زیر است:

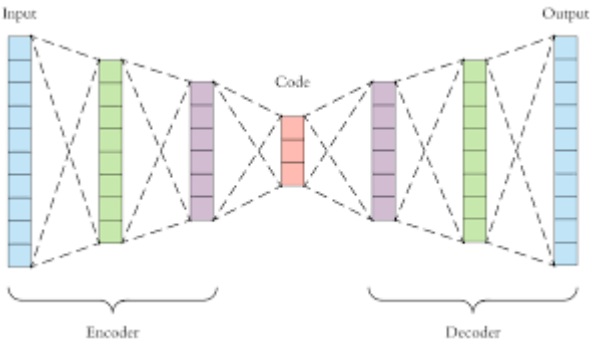
$$\mathbf{W}^{k+1} = \mathbf{W}^k \circ \frac{\mathbf{X}\mathbf{H}^\top}{\mathbf{W}\mathbf{H}\mathbf{H}^\top} \quad \mathbf{H}^{k+1} = \mathbf{H}^k \circ \frac{\mathbf{W}^{k\top}\mathbf{X}}{(\mathbf{W}^k)^\top\mathbf{W}^k\mathbf{H}^k}$$

برای متوقف شدن فرایند کاری که میتوانیم انجام دهیم به اینصورت است که یا میگوییم خطای بازسازی (اخلاف نرم فروبینیوس WH با X) از حدی کمتر باشد یا اینکه تعداد گام مورد نظر خود را انجام داده باشیم که با ازمون خطا به این نتیجه رسیدیم که حدودا اگر همواره 10000 قدم فرایند را انجام دهیم دقت خوبی میرسیم و بهتر است براین اساس فرایند را قطع کنیم.

در نهایت از خروجی بدست آمده میتوانیم استفاده کنیم به صورت کد شده و طبقه بندی را روی آن انجام میدهیم.

3-4 شبکه **autoencoder**

در نهایت ما از روش های مبتنی بر شبکه عصبی استفاده میکنیم تا اینکه ویژگی ها را کاهش بدهیم، برای این کار از ساختار انکودر دیکودر ساده استفاده میکنیم، در این حالت ما برای لایه bottleneck تعداد ویژگی های مورد نظر اندازه آن را قرار میدهیم. ساختار به صورت این است که ابتدا یک لایه مخفی با 1280 نورون، سپس 640 نورون و سپس لایه باتل نک که کد کننده ما هست قرار میدهیم، حال همین ساختار را برعکس پیاده سازی میکنیم تا دیکود کردن صورت بگیرد، در نهایت بعد از اینکه ما اموزش میدهیم تا خود سیگنال را بازسازی کند، تنها بخش انکودر شبکه را برمیداریم تا بخش کد کردن را برای ما انجام بدهد. ساختار کلی اگر به صورت شکل بخواهیم بگوییم به صورت زیر است:



ما به صورت خاص پیش پردازشی برای این قسمت انجام ندادیم(تنها همان z-score normalization) در سایر مقالات دیده شده به طور مثال برای حالاتی که اطلاعات گم شده زیاد است سطر های مربوطه حذف میشود و ورودی تمام صفر برای ویژگی نداریم ولی این موارد به صورت جزیی بررسی نشد، همچنین مدل بدون ترم رگولاریزاسیون خاصی استفاده نمیشود، برای بهنیه سازی از بهینه ساز adam استفاده کرده‌ایم.

4 بررسی نتایج

4-1 مقدمه

حال براساس متد هایی که در بخش های قبل معرفی کرده ایم میاییم و نتایج را بررسی میکنیم، برای این کار دو بخش داریم ابتدا برای دو پایگاه داده ابتدایی که نتیجه دقت کامل بدست میایید تنها روش های انتخاب ویژگی را بررسی میکنیم، چون با روشی مثل pca ما حتی با دو ویژگی میتوانیم تفکیک کنیم در حالی که با انتخاب مقدار بیشتر است، برای مقایسه در این حالت بررسی میکنیم هر روش چه مقدار ویژگی لازم است انتخاب کنیم تا به دقت کامل برسیم.

حال برای دو پایگاه داده اخر روش های کاهش ویژگی را نیز وارد میکنیم و همه را با هم بررسی میکنیم، در این حالت ما ویژگی ها را روی مقادیری ست میکنیم تا شرایط برای همه یکسان باشد و سپس دقت را بدست میاوریم و مقایسه میکنیم.

4-2 پایگاه دادگان جدایی پذیر

در دو پایگاه داده اول(lymphoma, leukemia) جدول به صورت زیر میشود:(در جدول روش های ستاره دار ما از کتابخانه اماده استفاده کرده ایم و بقیه روش ها را خودمان پیاده سازی کرده ایم و برای همین شاید از نظر بهینگی زمانی در بهترین حالت نباشد ولی باز قابل مقایسه کردن است)

برای :lymphoma

method	DKBCGS	Chi 2*	MRMR*	Genetic Algorithm
Number of features	53	53	42	40
Run time	0.01233	0.010495	~ 1 min	~10 min

و برای leukemia داریم:

method	DKBCGS	Chi 2*	MRMR*	Genetic Algorithm
Number of features	35	39	29	28
Run time	0.015576	0.017495	~ 1 min	~10 min

ما برای الگوریتم ژنتیک این شرط را گذاشتیم که تعداد ویژگی انتهخابی کمتر از 29(که توسط MRMR به آن رسیده بودیم) باشد، برای این کار میتوان در جمعیت تولیدی اگر بیش از 29 باشد به صورت رندون حذف کنیم و این کار را تا وقتی انجام داده ایم که دقت به 100 درصد برسد.

همانطور که مشخص است روش مقاله اول که DKBCGS نام دارد در مقایسه با روش های پایه دیگر مانند chi2 در تعداد یکسان ویژگی به دقت کامل میرسد، و حتی از نظر زمانی نیز به شدت بهنیه هستند ولی روش های زمان گیری مانند MRMR و الگوریتم ژنتیک از همه بهتر ولی به شدت زمان گیر تر است.

از این جدول ها مشخص است که ما اگر دنبال بهینگی زمانی باشیم باید از روش های پایه مانند مقاله اول گفته شده و یا chi2 استفاده کنیم، ولی اگر دقت مهم باشد روش های سرچ مانند ژنتیک و یا روش زمان گیری مانند MRMR دقت بهتری میدهد.

4-3 پایگاه دادگان غیرجدایی پذیر

در این حالت مقایسه را برای همه حالات انجام میدهیم چون دیگر واقعا از نظر مقایسه معنا دار میشود، ما تعداد ویژگی را برروی 75 ست میکنیم حال برای پایگاه دادهlung داریم:

method	DKBCGS	Chi 2*	MRMR*	Genetic Algorithm	Backward (svm)	PCA	NMF	Encoder-decoder
Accuracy	0.97	0.94	0.98	0.98	0.78	0.87	0.85	0.91
runtime	0.01896	0.02225	~1min	~10min	-	~5min	~1min	-

در اینجا مشاهده میشود که کاهش ویژگی مانند PCA که به نوعی اطلاعات همه ابعاد را نگه میدارد هنگامی که واقعا ویژگی ها همه مناسب نباشد به مشکل میخورد، در NMF این مشکل نیز برقرار است.

حالت encoder decoderکه ویژگی های سطح بالاتر جدا میکند میتوان دید دقت نسبت به بقیه حالات بهتر است ولی در اینجا روش های انتخاب ویژگی که به صورت خالص بهترین ویژگی های تفکیک پذیر را انتخاب میکنند مشاهده میشود، باز هم این امر مشاهده میشود که روش ژنتیک و MRMR بهترین هستند و سپس روش مقاله اول یعنی DKBCGS و درنهایت chi2.

حال برای پایگاه داده breast مقایسه میکنیم، نکته مهم این است که در این حالت ما داده missing زیاد داریم جدول به صورت زیر میشود:

method	DKBCGS	Chi 2*	MRMR*	Genetic Algorithm	Backward (svm)	PCA	NMF	Encoder-decoder
Accuracy	0.91	0.91	0.93	0.95	0.86	0.86	0.71	0.95
runtime	0.01896	0.02225	~1min	~10min	-	~5min	~1min	-

در این حالت دیگر برتر جامع متد های انتخاب ویژگی مشاهده نمیشود، (البته فقط encoder decoder بهتر است که ما حدس خود را از این بابت ذکر میکنیم) ولی سایر روش های کاهش ویژگی به همان دلیل که در قبل گفته ایم نسبت به انتخاب ویژگی نتایج بدتر هستند و در خود روش های انتخاب ویژگی نیز همان حالت همیشگی که بهترین روش ها ژنتیک و MRMR هستند و سپس DKBCGS و سپس chi2.

در رابطه با اینکه چرا شبکه encoder decoder در این حالت بهترین جواب را میدهد، از نظر ما دلیل این است که به صورت ذاتی خود داده ها را بازسازی میکند، همچنین چون رفتار داده بیمار و سالم متفاوت است به صورت ذاتی منیفلد داده های مپ شده این دو بخش از هم فاصله دارند بنابراین برای طبقه بندی نیز مناسب تر است و در این بخش نتیجه مناسب میدهد.

4-4 جمع بندی

در مقایسه روش ها به صورت کلی اگر داده گم شده نداشته باشیم، روش های انتخاب ویژگی روش های برتری هستند، روش جستجو ژنتیک بهترین نتیجه را همواره دارد و بعد از آن MRMR، ولی نسبت به روش ها DKBCGS و chi2 از نظر زمانی بهینه نیستند، و روش DKBCGS از chi2 بهتر است، ولی در حالتی که missing data داشته باشیم روش انکودر دیکودر انتظار داریم بهترین نتیجه را داشته باشد.

5 مراجع

[1] Liu, S., Xu, C., Zhang, Y., Liu, J., Yu, B., Liu, X., & Dehmer, M. (2018). Feature selection of gene expression data for Cancer classification using double RBF-kernels. *BMC Bioinformatics*, *19*(1).

[2] Lu, H., Chen, J., Yan, K., Jin, Q., Xue, Y., & Gao, Z. (2017). A hybrid feature selection algorithm for gene expression data classification. *Neurocomputing*, *256*, 56–62.

[3] Gao, W., Kannan, S., Oh, S., & Viswanath, P. (2017). Estimating mutual information for discrete-continuous mixtures. *Advances in Neural Information Processing Systems*, *2017-December*, 5987–5998. http://thuijskens.github.io/2017/10/07/feature-selection/