به نام خدا

تمرین سری دوم
درس یادگیری ماشین
دکتر علی شریفی زارچی

فرزان رحمانی
۴۰۳۲۱۰۷۲۵

سوال اول

از این الگوریتم استفاده میکنیم:

Algorithm 1 Sample Covariance Matrix

Input: $X \in \mathbb{R}^{N \times d}$ (data matrix with N data points and d dimensions)
Compute the mean of each feature: $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
Subtract the mean from each data point (center the data): $\tilde{X} \leftarrow X - \bar{x}^T$
Compute the covariance matrix: $\Sigma = \frac{1}{N-1} \tilde{X}^T \tilde{X}$
Compute the eigenvalues and eigenvectors of Σ: $[\lambda_1, \lambda_2, \ldots, \lambda_d], [v_1, v_2, \ldots, v_d] = \text{eig}(\Sigma)$
Select the top k eigenvectors corresponding to the largest eigenvalues: $A \leftarrow [v_1, v_2, \ldots, v_k]$
Transform the data into the new subspace: $X' \leftarrow X \cdot A$
Output: $X' \in \mathbb{R}^{N \times k}$ (transformed data with reduced dimensions)

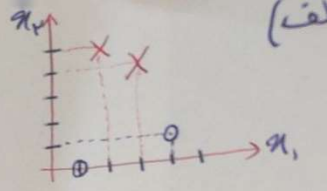
ادامه جواب در صفحه بعد آمده است:

۱. الف)

$$X = \begin{bmatrix} ۴ & ۱ \\ ۱ & ۰ \\ ۲ & ۵ \\ ۳ & ۴ \end{bmatrix}$$

ابتدا کل مجموعه داده را به صورت ماتریسی می‌نویسیم.

حال برای محاسبه بردار میانگین داریم:



$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{۴}\sum_{i=1}^{۴} x_i = \frac{1}{۴}\left([۴,۱]+[۱,۰]+[۲,۵]+[۳,۴]\right) = \frac{1}{۴}[۱۰,۱۰] = [۲,۵,\ ۲,۵]$$

$$\mu = [۲,۵\ \ ۲,۵] \xrightarrow{\text{حال با استفاده از}} \tilde{X} = X - \mu = \begin{bmatrix} ۱,۵ & -۱,۵ \\ -۱,۵ & -۲,۵ \\ -۰,۵ & ۲,۵ \\ ۰,۵ & ۱,۵ \end{bmatrix}$$

میانگین داده‌ها را center می‌کنیم. (centered data)

حال ماتریس Covariance را با استفاده از $n=4$ محاسبه می‌کنیم:

چون فرض کردیم کل مجموعه داده را داریم.

$$\Sigma = \frac{1}{n}\tilde{X}^T\tilde{X} = \frac{1}{۴}\tilde{X}^T\tilde{X}$$

$$\Sigma = \frac{1}{۴}\begin{bmatrix} ۱,۵ & -۱,۵ & -۰,۵ & ۰,۵ \\ -۱,۵ & -۲,۵ & ۲,۵ & ۱,۵ \end{bmatrix}\begin{bmatrix} ۱,۵ & -۱,۵ \\ -۱,۵ & -۲,۵ \\ -۰,۵ & ۲,۵ \\ ۰,۵ & ۱,۵ \end{bmatrix} = \frac{1}{۴}\begin{bmatrix} ۵ & ۱ \\ ۱ & ۱۷ \end{bmatrix} = \begin{bmatrix} ۱,۲۵ & ۰,۲۵ \\ ۰,۲۵ & ۴,۲۵ \end{bmatrix}$$

سپس مقادیر ویژه و بردارهای ویژه ماتریس Σ را پیدا می‌کنیم.

$$\det(\Sigma - \lambda I) = ۰$$

$$\det\left(\begin{bmatrix} ۱,۲۵-\lambda & ۰,۲۵ \\ ۰,۲۵ & ۴,۲۵-\lambda \end{bmatrix}\right) = \begin{vmatrix} ۱,۲۵-\lambda & ۰,۲۵ \\ ۰,۲۵ & ۴,۲۵-\lambda \end{vmatrix} = (۱,۲۵-\lambda)(۴,۲۵-\lambda) - (۰,۲۵)^۲ = ۰$$

$$\lambda^۲ - ۵,۵\lambda + ۵,۳۱۲۵ - ۰,۰۶۲۵ = ۰$$

$$\lambda^۲ - ۵,۵\lambda + ۵,۲۵ = ۰$$

بالگ sklearn ← معادله را حل کردم.

مقدارهای ویژه

$$\lambda_۱ \simeq ۴,۲۷۰۷$$
$$\lambda_۲ \simeq ۱,۲۲۹۳$$

چون $\lambda_۱$ بزرگتر از $\lambda_۲$ است.

$PC1 \rightarrow \lambda_۱, V_۱$

$PC2 \rightarrow \lambda_۲, V_۲$

بردارهای ویژه

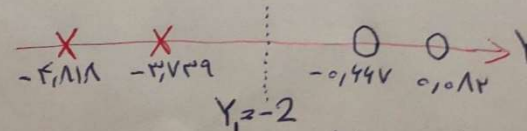
$\lambda_۱ = ۴,۲۷۰۷ \xrightarrow{\text{sklearn}} V_۱ \simeq [۰,۰۸۲۵ \quad -۰,۹۹۶۶]$

$\lambda_۲ = ۱,۲۲۹۳ \xrightarrow{\text{sklearn}} V_۲ \simeq [-۰,۹۹۶۶ \quad -۰,۰۸۲۵]$

ابتدا داده‌ها را با PC1 ($V_۱$) project می‌کنیم.

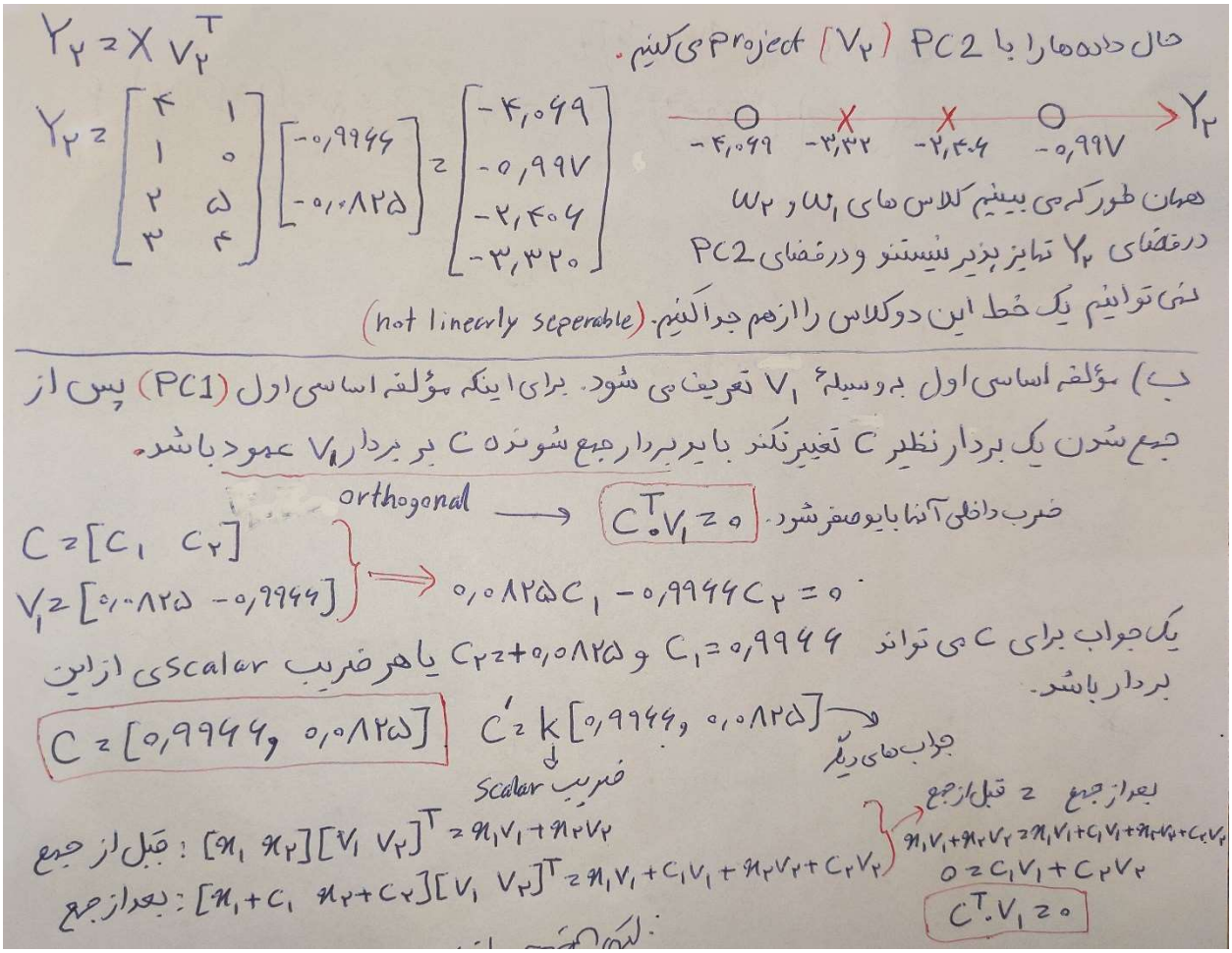
$$Y_۱ = X\,V_۱^T$$

$$Y_۱ = \begin{bmatrix} ۴ & ۱ \\ ۱ & ۰ \\ ۲ & ۵ \\ ۳ & ۴ \end{bmatrix}\begin{bmatrix} ۰,۰۸۲۵ \\ -۰,۹۹۶۶ \end{bmatrix} = \begin{bmatrix} -۰,۶۶۷ \\ ۰,۰۸۲ \\ -۴,۸۱۸ \\ -۳,۷۳۹ \end{bmatrix}$$



$Y_۱ = -2$

همان طور که می‌بینیم کلاس‌های $w_۱$ و $w_۲$ تمایز پذیر هستند و در فضای $Y_۱$ (همان PC1) می‌توانیم با خطی نظیر $Y_۱ = -2$ این دو کلاس را جدا کنیم. (linearly seperable)



برای اینکه first principal component بدون تغییر بماند، برداری که اضافه می کنیم باید متعامد به مؤلفه اصلی اول باشد. دلیل آن این است که وقتی نقاط داده را بر روی project PC1 می کنیم، projection یک عملیات خطی است. افزودن بردار متعامد به PC1، نمایش نقاط داده را به PC1 تغییر نمی دهد، زیرا حاصلضرب نقطه ای PC1 و بردار متعامد صفر خواهد بود. افزودن این بردار به تمام نقاط داده، جهت اولین جزء اصلی را تغییر نمی دهد.

سوال دوم

الف) مدل‌های ضعیف مستقل

- M مدل ضعیف داریم: $(f_1(X), f_2(X), \dots, f_M(X))$
- ensemble model:

$$f_{\text{ensemble}}(X) = \frac{1}{M} \sum_{i=1}^{M} f_i(X)$$

- هر مدل ضعیف دارای سوگیری (b) و واریانس (v) است.
- مدل های ضعیف مستقل از یکدیگر هستند.

هدف: بایاس و واریانس $f_{\text{ensemble}}(X)$ را محاسبه کنیم.

محاسبه بایاس

بایاس یک مدل به عنوان تفاوت بین پیش‌بینی مورد انتظار مدل و خروجی واقعی (Y) تعریف می‌شود:

$$\text{Bias}\big(f_{\text{ensemble}}(X)\big) = E[f_{\text{ensemble}}(X)] - Y$$

بیایید پیش‌بینی مورد انتظار $f_{\text{ensemble}}(X)$ را محاسبه کنیم:

$$E[f_{\text{ensemble}}(X)] = E\left[\frac{1}{M}\sum_{i=1}^{M} f_i(X)\right]$$

از آنجایی که امید ریاضی یک عملگر خطی است:

$$E[f_{\text{ensemble}}(X)] = \frac{1}{M}\sum_{i=1}^{M} E[f_i(X)]$$

برای هر مدل ضعیف ($f_i(X)$)، پیش‌بینی مورد انتظار این است:

$$E[f_i(X)] = Y + b$$

بدین ترتیب:

$$E[f_{\text{ensemble}}(X)] = \frac{1}{M}\sum_{i=1}^{M}(Y + b) = \frac{1}{M}.M.(Y + b) = Y + b$$

بنابراین، بایاس ensemble model این است:

$$\text{Bias}\big(f_{\text{ensemble}}(X)\big) = (Y + b) - Y = b$$

نتیجه‌گیری: بایاس ($f_{\text{ensemble}}(X)$) برابر با بایاس مدل‌های ضعیف (b) است.

محاسبه واریانس

واریانس ($f_{\text{ensemble}}(X)$) به صورت زیر تعریف می‌شود:

$$\text{Var}\big(f_{\text{ensemble}}(X)\big) = E[(f_{\text{ensemble}}(X) - E[f_{\text{ensemble}}(X)])^2]$$

جایگزین کردن ($f_{\text{ensemble}}(X)$):

$$\text{Var}\big(f_{\text{ensemble}}(X)\big) = \text{Var}\left(\frac{1}{M}\sum_{i=1}^{M} f_i(X)\right)$$

استفاده از ویژگی واریانس برای متغیرهای مستقل:

$$\text{Var}\big(f_{\text{ensemble}}(X)\big) = \frac{1}{M^2}\sum_{i=1}^{M} \text{Var}\big(f_i(X)\big)$$

با توجه به اینکه واریانس هر مدل ضعیف (v) است:

$$\mathrm{Var}\big(f_{\text{ensemble}}(X)\big) = \frac{1}{M^2}\sum_{i=1}^{M} v = \frac{Mv}{M^2} = \frac{v}{M}$$

نتیجه گیری:

- بایاس ensemble model (b) است.
- واریانس ensemble model ($\frac{v}{M}$) است.

اثر افزایش (M):

- با افزایش(M) بایاس بدون تغییر باقی می ماند.
- با افزایش (M) ، واریانس کاهش می یابد زیرا ($\frac{v}{M}$) کوچکتر می شود.

ب) مدل‌های ضعیف همبسته

- M مدل ضعیف داریم: $(f_1(X), f_2(X), \dots, f_M(X))$
- ensemble model:

$$f_{\text{ensemble}}(X) = \frac{1}{M}\sum_{i=1}^{M} f_i(X)$$

- هر مدل ضعیف دارای سوگیری (b) و واریانس (v) است.
- مدل های ضعیف یک همبستگی (ρ) بین هر جفت $(f_i(X))$ و $(f_j(X))$ که در آن ($i \neq j$) است.

می‌خواهیم بایاس و واریانس ensemble model را محاسبه کنیم و تأثیر همبستگی (ρ) را روی این معیارها ببینیم.

مرحله ۱: محاسبه بایاس

بایاس ensemble model دقیقا مانند قسمت (الف) است زیرا سوگیری به همبستگی بین مدل‌ها بستگی ندارد.

برای هر مدل ضعیف:

$$E[f_i(X)] = Y + b$$

برای ensemble model:

$$E[f_{\text{ensemble}}(X)] = E\left[\frac{1}{M}\sum_{i=1}^{M} f_i(X)\right]$$

استفاده از خطی بودن امید ریاضی:

$$E[f_{\text{ensemble}}(X)] = \frac{1}{M}\sum_{i=1}^{M} E[f_i(X)] = \frac{1}{M}\sum_{i=1}^{M} (Y + b) = Y + b$$

بایاس میشود:

$$\mathrm{Bias}\big(f_{\text{ensemble}}(X)\big) = E[f_{\text{ensemble}}(X)] - Y = (Y + b) - Y = b$$

مرحله ۲: محاسبه واریانس با همبستگی

برای محاسبه واریانس ($f_{\text{ensemble}}(X)$) با تعریف آن شروع می کنیم:

$$\text{Var}\left(f_{\text{ensemble}}(X)\right) = \text{Var}\left(\frac{1}{M}\sum_{i=1}^{M} f_i(X)\right)$$

فرمول واریانس را گسترش میدهیم:

برای هر متغیر تصادفی (X)و (Y):

$$\text{Var}(aX + bY) = a^2\text{Var}(X) + b^2\text{Var}(Y) + 2ab \cdot \text{Cov}(X, Y)$$

با استفاده از این، می توانیم واریانس مجموع را گسترش دهیم:

$$\text{Var}\left(\frac{1}{M}\sum_{i=1}^{M} f_i(X)\right) = \frac{1}{M^2}\text{Var}\left(\sum_{i=1}^{M} f_i(X)\right)$$

$$\text{Var}\left(\sum_{i=1}^{M} f_i(X)\right) = \sum_{i=1}^{M} \text{Var}\left(f_i(X)\right) + \sum_{i\neq j} \text{Cov}\left(f_i(X)،f_j(X)\right)$$

مقادیر واریانس و کوواریانس را جایگزین میکنیم.

- $\text{Var}\left(f_i(X)\right) = v$) برای هر (i)
- $\text{Cov}\left(f_i(X)،f_j(X)\right) = \rho.\sqrt{Var(X).Var(Y)} = \rho.\sqrt{v.v} = \rho v$) برای ($i \neq j$)

اکنون هر قسمت را محاسبه می کنیم:

۱. مجموع واریانس ها:

$$\sum_{i=1}^{M} \text{Var}\left(f_i(X)\right) = \sum_{i=1}^{M} v = Mv$$

۲. مجموع کوواریانس ها:

- $(M(M-1))$ جفت های متمایز $((i,j))$ وجود دارد که در آن ($i \neq j$)
- $\sum_{i\neq j} \text{Cov}\left(f_i(X), f_j(X)\right) = \rho v \cdot M(M-1)$

مجموع واریانس را محاسبه میکنیم.

$$\text{Var}\left(\sum_{i=1}^{M} f_i(X)\right) = Mv + \rho v \cdot M(M-1)$$

اکنون، بر (M^2) تقسیم کنید تا واریانس $f_{\text{ensemble}}(X)$ را بدست آوریم:

$$\text{Var}\left(f_{\text{ensemble}}(X)\right) = \frac{Mv + \rho v \cdot M(M-1)}{M^2}$$

عبارت را ساده میکنیم:

$$\mathrm{Var}\big(f_{\text{ensemble}}(X)\big) = \frac{v}{M} + \rho v \cdot \frac{M(M-1)}{M^2}$$

$$\mathrm{Var}\big(f_{\text{ensemble}}(X)\big) = \frac{v}{M} + \rho v \cdot \frac{M-1}{M}$$

نتیجه گیری

- بایس ensemble model (b) است.
- واریانس مدل مجموعه عبارت است از:

$$\mathrm{Var}\big(f_{\text{ensemble}}(X)\big) = \frac{v}{M} + \rho v \cdot \frac{M-1}{M}$$

تجزیه و تحلیل: اثر افزایش (M) و همبستگی (ρ)

۱. اثر افزایش (M) :

- اولین عبارت ($\frac{v}{M}$)با افزایش (M) کاهش می یابد.
- ترم دوم ($\rho v \cdot \frac{M-1}{M}$) با افزایش (M) به (ρv)نزدیک می شود.
- زمانی که ($M \to \infty$) :

$$\mathrm{Var}\big(f_{\text{ensemble}}(X)\big) \to \rho v$$

- بنابراین، اثر کاهش واریانس با افزایش تعداد مدل‌های ضعیف (M) در صورت وجود همبستگی(ρ) بالا کاهش می‌یابد.

۲. اثر همبستگی (ρ) :

- اگر ($\rho = 0$) (مدل‌های مستقل)، واریانس ($\frac{v}{M}$) است که با افزایش (M) کاهش می‌یابد.
- اگر ($\rho > 0$)، واریانس به دلیل عبارت اضافی ($\rho v \cdot \frac{M-1}{M}$) افزایش می‌یابد.
- همبستگی بالاتر (ρ) منجر به کاهش کمتری در واریانس با افزایش (M) می شود.

خلاصه:

- برای مدل‌های مستقل ($\rho = 0$)، واریانس به صورت $\frac{v}{M}$ کاهش می‌یابد.
- برای مدل های همبسته (($\rho > 0$))، کاهش واریانس کمتر موثر است زیرا عبارت ($\rho v \cdot \frac{M-1}{M}$) قابل توجه است.
- با افزایش (ρ) ، مزیت افزایش (M) کاهش می یابد و واریانس به (ρv) برای (M) بزرگ نزدیک می شود.

پ)

- نه، یادگیرنده های ضعیف در AdaBoost نیازی به مشتق پذیر بودن ندارند. AdaBoost یک الگوریتم یادگیری گروهی است که چندین دسته بند ضعیف (یا «یادگیرندگان ضعیف») را برای ایجاد یک طبقه‌بندی قوی ترکیب می‌کند. یادگیرندگان ضعیف می توانند به هر شکلی باشند، به شرطی که بهتر از حدس زدن تصادفی عمل کنند. مشتق پذیری یک الزام برای AdaBoost نیست، زیرا الگوریتم به جای بهینه سازی یک تابع ضرر مشتق پذیر، بر تنظیم مکرر وزن نمونه های آموزشی تمرکز دارد. در واقع، AdaBoost با تنظیم وزن نمونه‌های طبقه‌بندی اشتباه کار می‌کند و به بهینه‌سازی مبتنی بر گرادیان متکی نیست (برخلاف روش‌های gradient boosting)، بنابراین نیازی به مشتق پذیری نیست.

- Boosting معمولاً از نظر محاسباتی گران‌تر از bagging است. دلایل:
  - آموزش متوالی: Boosting یادگیرندگان ضعیف را به طور متوالی آموزش می دهد و هر مدل جدید بر روی خطاهای قبلی تمرکز می کند. این فرآیند iterative است و نمی توان آن را به طور موثر موازی کرد. این فرآیند مکرر متوالی می‌تواند از نظر محاسباتی گران باشد، به ویژه با افزایش تعداد یادگیرندگان ضعیف.
  - وزن دهی مجدد داده ها: Boosting وزن نمونه ها را در هر تکرار تنظیم می کند و در هر مرحله محاسبات اضافی را اضافه می کند.
  - در مقابل، Bagging (به عنوان مثال، Random Forest) چندین مدل را به طور مستقل بر روی bootstrap samples های مختلف آموزش می دهد، که امکان آموزش موازی را فراهم می کند. در واقع، یادگیرندگان ضعیف در Bagging به طور مستقل آموزش می بینند و پیش بینی نهایی با جمع آوری خروجی های تک تک یادگیرندگان (مثلاً از طریق رأی اکثریت یا میانگین گیری) به دست می آید. که این کار که میتواند موازی هم باشد از نظر محاسباتی کارآمدتر است.
  - هزینه محاسباتی در boosting با تعداد یادگیرندگان ضعیف scale میشود، زیرا هر یادگیرنده جدید باید در وزن های به روز شده نمونه های آموزشی آموزش ببیند. از سوی دیگر، Bagging می تواند از محاسبات موازی برای آموزش همزمان یادگیرندگان ضعیف استفاده کند، که باعث می شود هزینه محاسباتی نسبتاً کمتری داشته باشد.

سوال سوم

در این سوال فرض میکنیم هر نقطه همسایه خودش هم هست. (در بحث های کوئرا بیان شد.)

الف) k=1 و مقدار خطا = 0 است.

تعیین مقدار k که خطا را به حداقل می رساند:

- برای k=1: طبقه بندی کننده KNN تنها نزدیک ترین همسایه را برای طبقه بندی هر نقطه در نظر می گیرد. از آنجایی که هر نقطه نزدیکترین همسایه خود است، خطای طبقه بندی ۰ خواهد بود زیرا هر نقطه بر اساس خودش به درستی طبقه بندی می شود. (فاصله L2 هر نقطه از خودش ۰ است و ما هم فقط نزدیک ترین همسایه را که خود نقطه هست در نظر میگیریم.)
- برای k>1: با افزایش k، طبقه بندی کننده همسایگان بیشتری را در نظر می گیرد. اگر نقاط همسایه متعلق به کلاس‌های مختلف باشند، ممکن است منجر به طبقه‌بندی اشتباه و افزایش شود.

در حالت کلی، برای یافتن مقدار دقیق k که خطا را به حداقل می‌رساند، باید مقادیر k مختلف را روی مجموعه داده cross validation آزمایش کنیم. در این سوال، زمانی که k از ۱ بالاتر می‌رود، شروع به مشاهده خطاهای طبقه‌بندی می‌کنیم.

برای مجموعه داده های کوچکی مانند این سوال، k=1 معمولاً خطا را به حداقل می رساند زیرا:

- هر نقطه همسایه خود است، بنابراین به درستی طبقه بندی شده است.
- مقدار خطا(نسبت دسته‌بندی‌های نادرست به کل دسته‌بندی‌ها) در k=1 برابر با 0/14=0 است (۱۴ نقطه داده داریم).

ب) مقادیر k بسیار کوچک:

- طبقه بندی کننده به نویز و outliers های موجود در دیتاست بسیار حساس می شود زیرا فقط نزدیکترین همسایه را در نظر می گیرد. که این نکته منجر به poor generalization to new data می شود.
- ممکن است منجر overfitting و high variance شود زیرا مرز تصمیم می تواند بسیار پیچیده شود و با نقاط داده individual سازگار شود و نویز ها را نیز در برگیرد.

مقادیر k بزرگ:

- طبقه‌بندی کننده همسایه‌های زیادی را در نظر می‌گیرد، احتمالاً شامل نقاطی از کلاس‌های مختلف است که می‌تواند منجر به طبقه‌بندی اشتباه شود.
- به عنوان مثال، اگر k نزدیک به تعداد کل نقاط باشد، تصمیم دسته بند به سمت کلاس اکثریت(majority class) در کل مجموعه داده متمایل می شود و الگوهای محلی را نادیده می گیرد و منجر underfitting و low variance می شود.
- دسته بند بسیار ساده میشود و نمی‌تواند الگوهای اساسی در داده‌ها را ثبت کند و منجر به عملکرد ضعیف می‌شود.

پ) k=5 مقدار خطای دسته بند را کمینه میکند. مقدار این خطا 0.2857142857142857 = 4/14 می‌شود.

```
Optimal k: 5, Error: 0.2857142857142857
```

Leave One Out Cross-Validation (LOOCV) روشی است که در آن هر نمونه یک بار به عنوان نمونه آزمایشی استفاده می شود در حالی که بقیه مجموعه داده به عنوان مجموعه آموزشی عمل می کند.

Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the function approximator is trained on all the data except for one point and a prediction is made for that point. (https://www.cs.cmu.edu/~schneide/tut5/node42.html)

k بهینه را می توان با استفاده از کتابخانه ای مانند scikit-learn در پایتون پیدا کرد. این کار شامل fit کردن مدل KNN برای مقادیر مختلف k و بررسی اینکه کدام k خطای طبقه بندی را به حداقل می رساند.

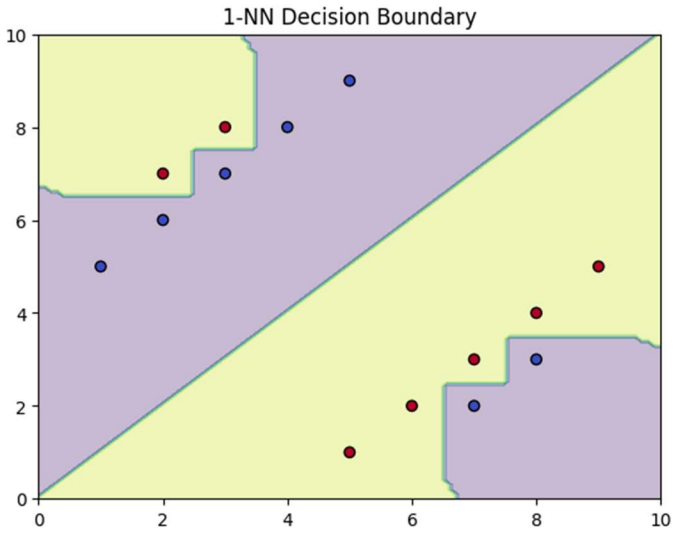
این کد به شما k بهینه را با استفاده از LOOCV می دهد:

```python
from sklearn.model_selection import LeaveOneOut
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import numpy as np
# Example input data (x, y) and labels (Replace with actual data if available)
X = np.array([[5, 1], [6, 2], [7, 2], [7, 3], [8, 3], [8, 4], [9, 5],
              [1, 5], [2, 6], [2, 7], [3, 7], [3, 8], [4, 8], [5, 9]])  #
Coordinates from the image
y = np.array([1, 1, 0, 1, 0, 1, 1,
            0, 0, 1, 0, 1, 0, 0])  # Labels (1 for positive, 0 for negative)
loo = LeaveOneOut()
best_k = 1
best_score = 0
# Trying out different k values
for k in range(1, len(X)):
    # knn = KNeighborsClassifier(n_neighbors=k)
    knn = KNeighborsClassifier(n_neighbors=k, metric='euclidean')
    scores = []
    # Perform LOOCV
    for train_index, test_index in loo.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        knn.fit(X_train, y_train)
```

```
        y_pred = knn.predict(X_test)
        scores.append(accuracy_score(y_test, y_pred))
    # Calculate mean accuracy for current k
    mean_score = np.mean(scores)
    if mean_score > best_score:
        best_score = mean_score
        best_k = k
print(f"Optimal k: {best_k}, Error: {1 - best_score}")
```

ت) برای دسته بند ۱-NN:

- مرز تصمیم با نقاطی که از نزدیکترین نمونه های مثبت و منفی فاصله دارند، تعریف می شود.
- برای این مجموعه داده، مرزهای خطی تکه تکه‌ای بین نقاط کلاس‌های مختلف تشکیل می‌دهد، و در فضای ویژگی‌های دوبعدی، شکلی زیر را ایجاد می‌کند.
- این مرز بر اساس تراکم و مکان نقاط در هر کلاس تغییر خواهد کرد.



این کد به شما مرز تصمیم گیری 1-NN را می دهد:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
# Fitting 1-NN classifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X, y)
# Plotting decision boundary
x_min, x_max = 0, 10
y_min, y_max = 0, 10
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3)
```

```
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', cmap='coolwarm')
plt.title("1-NN Decision Boundary")
plt.show()
```

سوال چهارم

الف) به روز رسانی مراکز خوشه ها برای به حداقل رساندن تابع هزینه

تابع هزینه برای خوشه بندی k-means به صورت زیر تعریف می شود:

$$L = \sum_{j=1}^{k} \sum_{x_i \in S_j} |x_i - \mu_j|^2$$

هدف:

نشان دهیم که برای به حداقل رساندن تابع هزینه، مقدار بهینه (μ_j) میانگین تمام نقاط در (S_j) است.

اثبات:

۱. برچسب های (y_j) را ثابت در نظر میگیریم:

در این مرحله فرض می کنیم که تخصیص نقاط به خوشه ها (یعنی y_j) ثابت است.

۲. بازنویسی تابع هزینه برای یک خوشه واحد (j):

$$L_j = \sum_{x_i \in S_j} |x_i - \mu_j|^2$$

۳. گرفتن مشتق با توجه به (μ_j):

برای کمینه کردن (L_j)، باید مشتق (L_j) را نسبت به (μ_j) پیدا کرده و آن را صفر کنیم.

$$\frac{\partial L_j}{\partial \mu_j} = \frac{\partial}{\partial \mu_j} \left(\sum_{x_i \in S_j} |x_i - \mu_j|^2 \right)$$

از آنجایی که $(|x_i - \mu_j|^2 = (x_i - \mu_j) \cdot (x_i - \mu_j))$، این را گسترش داده و مشتق میگیریم:

$$\frac{\partial L_j}{\partial \mu_j} = \frac{\partial}{\partial \mu_j} \left(\sum_{x_i \in S_j} (x_i - \mu_j) \cdot (x_i - \mu_j) \right) = \sum_{x_i \in S_j} 2 \,.\, -1 \,.\, (x_i - \mu_j) = \sum_{x_i \in S_j} 2\,(\mu_j - x_i)$$

۴. برابر قرار دادن مشتق با صفر:

$$\sum_{x_i \in S_j} 2\,(\mu_j - x_i) = 0$$

ساده کردن:

$$\sum_{x_i \in S_j} (\mu_j - x_i) = 0$$

$$\sum_{x_i \in S_j} (\mu_j) \quad - \quad \sum_{x_i \in S_j} (x_i) = 0$$

$$\sum_{x_i \in S_j} (\mu_j) \quad = \quad \sum_{x_i \in S_j} (x_i)$$

$$|S_j| \mu_j = \sum_{x_i \in S_j} x_i$$

$$\mu_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$$

نتیجه گیری:

مرکز خوشه (μ_j) که تابع هزینه را به حداقل می رساند میانگین نقاط موجود در خوشه (S_j) است.

ب) **حساسیت به مقداردهی اولیه:**

بله، k-means به مقداردهی اولیه حساس است:

- مقداردهی اولیه متفاوت مراکز خوشه می تواند منجر به خوشه های نهایی متفاوت شود.
- مقداردهی اولیه ضعیف ممکن است باعث شود که الگوریتم به کمینه های محلی همگرا شود و در k-means لزوما به global minimum نمیرسیم. در نتیجه ممکن است به suboptimal clustering solutions بدی برسیم.
- مقداردهی اولیه ضعیف می تواند منجر به خوشه های خالی یا راه حل های غیربهینه شود.
- برای کاهش این امر، روش هایی مانند K-means++ برای انتخاب مراکز خوشه اولیه بهتر استفاده می شود. همچنین میتوانیم چند بار الگوریتم را اجرا کنیم.

همگرایی:

بله، k-means همگرا می شود، اما نه لزوما به global minimum:

- الگوریتم همگرایی را تضمین می کند زیرا تابع هزینه L در هر مرحله(reassignment and center update) افزایش نمی یابد. (non-increasing at each step)
- تابع هزینه bounded below by 0 است و در هر step کاهش می یابد. تنها تعداد محدودی(possibly many) از تخصیص نقاط به خوشه ها وجود دارد بنابراین، الگوریتم در نهایت به یک حالت پایدار میرسد.
- با این حال، k-means ممکن است بسته به محل قرارگیری اولیه مراکز خوشه، به local minimum همگرا شود.
- الگوریتم زمانی متوقف می شود که هیچ تغییری در مراکز خوشه وجود نداشته باشد یا زمانی که برچسب نقاط تغییر نمی کند.

پ) سناریو:

- فرض کنید یک نقطه x_j از چندین مرکز خوشه در مرحله به روز رسانی برچسب، فاصله یکسانی دارد.
- اگر یکی از گزینه ها این است که x_j را در همان خوشه ای که در تکرار قبلی بوده باقی بماند، بهتر است این گزینه را انتخاب کنید.

دلیل:

۱. تثبیت همگرایی(Stabilizing Convergence):

- اگر x_j را در خوشه قبلی خود در فاصله مساوی نگه داریم، با اجتناب از تغییرات غیرضروری در انتساب های خوشه، به تثبیت الگوریتم کمک می کند.
- جابجایی مکرر x_j بین خوشه‌های هم‌فاصله ممکن است باعث نوسانات در مراکز خوشه‌ای شود و همگرایی را به تاخیر بیاندازد یا حتی از آن جلوگیری کند.
- نقاط ممکن است به جابجایی بین خوشه‌ها ادامه دهند، حتی زمانی که مراکز تغییر نمیکنند.

۲. معیارهای توقف الگوریتم (Stopping Criteria):

- الگوریتم k-means معمولاً زمانی متوقف می شود که مراکز خوشه دیگر پس از یک تکرار تغییر نکنند یا تغییر خیلی ناچیزی داشته باشند.
- اگر نقاطی مانند x_j مکرر بین خوشه‌ها جابه‌جا می‌شوند، این شرط ممکن است برآورده نشود و باعث شود الگوریتم برای تکرارهای بیشتر اجرا شود.
- این روش کمک می کند تا از حلقه های بی نهایت جلوگیری شود و الگوریتم را قطعی تر می کند.

مشکل بدون این اصل:

مسئله نوسان:

- نقطه x_j ممکن است بین دو یا چند خوشه در تکرارهای متوالی در نوسان پیوسته باشد و از همگرا شدن الگوریتم جلوگیری کند. در واقع، نقاط می توانند بین خوشه ها جابجا شوند حتی زمانی که مراکز خوشه ها ثابت هستند. که ممکن است منجر به ایجاد چرخه ای که هرگز ثابت نمی شود بشود.
- این مسئله می تواند منجر به افزایش زمان محاسباتی شود و ممکن است قبل از رسیدن به حالت پایدار به تکرارهای بیشتری نیاز داشته باشد.

با انتخاب این روش برای نگه داشتن نقاط با فاصله مساوی در خوشه قبلی، احتمال این مسائل را کاهش می دهیم، بنابراین از همگرایی سریعتر و پایدارتر الگوریتم k-means اطمینان حاصل می کنیم.

سوال پنجم

الف) برای حل این مسئله، می خواهیم نشان دهیم که بردار یکه u که میانگین مربعات خطا (MSE) را بین نقاط و تصویر آنها به حداقل می رساند، در واقع اولین مؤلفه اصلی (PC1) است.

داریم:

- مجموعه ای از m نقاط داده نرمال شده: $(x^{(1)}, x^{(2)}, \dots, x^{(m)} \in R^n)$
- داده ها نرمال شده اند، یعنی میانگین هر بعد صفر و واریانس هر بعد یک است.
- تصویر x بر روی بردار u:

$$f_u(x) = \arg\min_{v \in V} |x - v|^2 \text{ where } V = \{au : a \in R\}$$

- هدف این است که:

$$\arg\min_{u:u^\top u=1} \frac{1}{m} \sum_{i=1}^{m} |x^{(i)} - f_u(x^{(i)})|^2$$

مرحله ۱: بیان خطای تصویر کردن

برای بیان $f_u(x)$، توجه داشته باشید که تصویر x بر روی u به صورت زیر داده می شود:

$$f_u(x) = (x^\top u)u$$

که $x^\top u$ ضرب داخلی است و $(x^\top u)u$ برداری در جهت بردار u است.

بنابراین، بردار خطا برای یک نقطه $(x - f_u(x))$ به صورت زیر است:

$$x - f_u(x) = x - (x^\top u)u$$

پس مربع خطای یک نقطه به صورت زیر است:

$$|x - (x^\top u)u|^2$$

مرحله ۲: خطای مربع را گسترش می‌دهیم.

بیایید این عبارت را گسترش دهیم:

$$|x - (x^\top u)u|^2 = (x - (x^\top u)u)^\top (x - (x^\top u)u)$$

گسترش ضرب نقطه ای:

$$|x - (x^\top u)u|^2 = x^\top x - 2(x^\top u)(u^\top x) + (x^\top u)^2(u^\top u)$$

از آنجایی که $(u^\top u = 1)$

$$|x - (x^\top u)u|^2 = x^\top x - 2(x^\top u)^2 + (x^\top u)^2 = x^\top x - (x^\top u)^2$$

مرحله ۳: مجموع خطا را محاسبه میکنیم.

حالا بیایید این خطا را روی تمام نقاط داده جمع کنیم و میانگین مربعات خطا را به حداقل برسانیم:

$$\text{Total Error} = \text{MSE} = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)\top}x^{(i)} - \left(x^{(i)\top}u\right)^2\right)$$

از آنجایی که داده ها نرمال شده اند، $(x^{(i)\top}x^{(i)} = |x^{(i)}|^2)$ برابر واریانس است (چون میانگین صفر است) که برابر با ۱ است:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)\top}x^{(i)}\right) - \frac{1}{m}\sum_{i=1}^{m}\left(\left(x^{(i)\top}u\right)^2\right) = Var(X) - \frac{1}{m}\sum_{i=1}^{m}\left(\left(x^{(i)\top}u\right)^2\right)$$

این معادله ساده می شود:

$$\text{MSE} = 1 - \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)\top}u\right)^2$$

مرحله ۴: به حداکثر رساندن $\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)\top}u\right)^2$

کمینه کردن خطای کل معادل به حداکثر رساندن عبارت دوم در معادله بالاست (چرا که ضریب آن منفی است عبارت اول ثابت است) است:

$$\arg\max_{u:u^\top u=1}\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)\top}u\right)^2=\arg\max_{u:u^\top u=1}\frac{1}{m}\sum_{i=1}^{m}\left(u^\top x^{(i)}\right)\left(x^{(i)\top}u\right)=\arg\max_{u:u^\top u=1}\frac{1}{m}\sum_{i=1}^{m}u^\top\left(x^{(i)}x^{(i)\top}\right)u$$

مرحله ۵: بازنویسی با ماتریس کوواریانس

Covariance Matrix یا همان ماتریس کوواریانس (Σ) را به صورت زیر تعریف میکنیم: (فرض میکنیم کل داده های را داریم اگر فقط نمونه ها را داشتیم باید در مخرج از m-1 استفاده میکردیم که البته تفاوت زیادی در محاسبه ایجاد نمیکند.)

$$\Sigma=\frac{1}{m}\sum_{i=1}^{m}x^{(i)}x^{(i)\top}$$

تابع هدف تبدیل می شود به:

$$\arg\max_{u:u^\top u=1}u^\top\Sigma u$$

مرحله ۶: ضرایب لاگرانژ (Lagrange Multipliers)

برای حل این مشکل بهینه سازی مشروط شده، از Lagrange Multipliers استفاده می کنیم. لاگرانژ را به صورت زیر تعریف میکنیم:

$$\mathcal{L}(u,\lambda)=f(u)+\lambda g(u)=u^\top\Sigma u-\lambda(u^\top u-1)$$

گرفتن مشتق با توجه به u و صفر قرار دادن آن:

$$\frac{\partial\mathcal{L}}{\partial u}=2\Sigma u-2\lambda u+0=0$$

این عبارت ساده می شود:

$$\Sigma u-\lambda u=0$$
$$\Sigma u\ =\ \lambda u$$

این دقیقا همان معادله مقدار ویژه برای ماتریس کوواریانس (Σ) است! جواب u باید بردار ویژه Σ باشد و λ مقدار ویژه مربوطه است. از آنجایی که می خواهیم $u^\top\Sigma u$ را به حداکثر برسانیم، باید بردار ویژه مربوط به بزرگترین مقدار ویژه را انتخاب کنیم که دقیقاً تعریف اولین مولفه اساسی (PC1) است.

مرحله ۷: نتیجه گیری

بردار u که explained variance را به حداکثر می رساند، بردار ویژه مربوط به بزرگترین مقدار ویژه (Σ) است که طبق تعریف همان اولین مؤلفه اصلی (PC1) است.

بدین ترتیب نشان دادیم:

$$\arg \min_{u:u^\top u=1} \frac{1}{m} \sum_{i=1}^{m} |x^{(i)} - f_u\left(x^{(i)}\right)|^2 = \text{PC1}$$

که این اثبات را کامل می کند.

همچنین این اثبات نشان می دهد که:

۱. تصویری(projected) که خطای بازسازی(reconstruction error) را به حداقل می رساند.
۲. جهتی که واریانس را به حداکثر می رساند.
۳. اولین مولفه اصلی (PC1).

همه معادل هستند!

پایان