تمرین سوم

یادگیری ماشین

پرهام رهنمایی

400108547

## مسئله ۱

• الف)

ابتدا یک تصحیح، ترکیب خطی آمین شرط $\alpha_x \geq 0$ را ندارد.

در نظر می‌گیریم $C = \{ t \mid t = \sum_{x \in X} \alpha_x x ,\ \alpha_x \geq 0 ,\ \sum \alpha_x = 1 \}$

این $C$ = کانوکس هال نقاط (درس affine hull همینش منتها شرط $\alpha_x \geq 0$ اومده)

کانوکس هال نقاط $X$ را $C_X$ و ۲ را $C_Y$ می‌نامیم. طبق فرض $C_X$ و $C_Y$ اشتراک ندارند.

* دقت می‌کنیم که $C$ هم subset closed از $\mathbb{R}^n$ هستش هم compact عه دیس

لم ۱) وجود دارد $a \in C_X$ و $b \in C_Y$ که فاصله $\|b-a\|$ که $a \in C_X$، $b \in C_Y$ مینیمایز می‌کند.

* البته نیازی به لم نیست هردو کامپکت انه :دی

یک مثبت نقطه می‌گیریم $a \in C_X$، $b \in C_Y$ و $r_1 = \|b-a\|$. چون $C_X$ کامپکته پس یک کره حول $a$ وجود داره که کل $C_X$ درش جا می‌شه ← شعاعش می‌ذاریم $r_2$.

$S$ را اشتراک $C_Y$ و کره بسته به شعاع $r_1+r_2$ حول $a$ می‌نامیم. حال $S$ کامپکته و خالی هم نیست چون معادل $b$ توشه. چون تابع فاصله پیوسته است پس $a_0$ و $b_0$ ای هست که $\|a_0-b_0\|$ مینیمم در کل $A \times S$. ادعا می‌کنم این $a_0, b_0$ برای $A \times B$ هم مینیممه. وگرنه

اگه $a', b'$ ای باشه که $\|a'-b'\| < \|a_0-b_0\|$ اون وقت یعنی $\|a'-b'\| < r_1$ (چون از طرفی هم فاصله $a_0 b_0$ کمتر مساوی $r_1$)

و داریم که $\|a-b'\| \le \|a'-b'\| + \|a-a'\| < r_1 + r_2$ پس $b'$ توی $S$ هستش

که اینا با نحوه انتخاب $a_0, b_0$ توی $S$ در تناقضه ( سوپریمم بودن ) • چون قطعاً $\sup_{x \ne y}$ اینا رو شامل می‌شود

★ ازونی که فاصله تابع پیوسته هستش و فضای ما کامپکته (و بسته هستش) داریم که min و max شو اتخاذ می‌کنه برا همین که گفتمش

حال داریم که $x_0 \in C_x$ و $y_0 \in C_y$ با فاصله مینیمم داریم . ( $C_x \cap C_y = \emptyset$ پس $x_0 \ne y_0$ )

حالا پاره‌خط $x_0, y_0$ را در نظر بگیریم و دو ابرصفحه عمود بهش در $x_0$ ، $l_x$ و $l_y$ در $y_0$ رسم نماییم.

ادعا می‌کنیم $C_x$ و $C_y$ با فضای بین $l_x$ و $l_y$ اشتراکی ندارند. اگه اینو بگیم پس ابرصفحه‌ها عمود بر پاره‌خط بجز صورت در شرایط مسئله صدق می‌کنند و در بخش کردند.

داریم که طبق تعریف ابرصفحه اگر $v := y_0 - x_0$ بردار و $\langle v, x_0 \rangle = t_x < t_y = \langle v, y_0 \rangle$

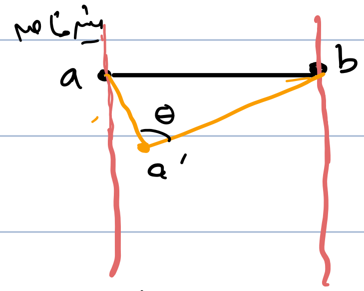
پس $l_x = \{x \mid \langle v, x \rangle = t_x\}$ $l_y = \{x \mid \langle v, x \rangle = t_y\}$

ادعام اینه که $\forall a \in C_x \quad \langle v, a \rangle \le t_x$

$\forall b \in C_y \quad \langle v, b \rangle \ge t_y$

فرض خلف که $\exists a \in C_x$ که $\langle v, a \rangle > t_x$ . پس $a'$ که عمود وارده از $y_0$ هستش روی پاره‌خط $[x_0, a]$

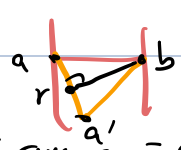
چون $C_x$ کانوکسه و $a$ و $x_0$ توشن پس $a'$ هم توشه . اما الان $a'$ از $x_0$ به $y_0$ نزدیک‌تر شد . که با انتخاب $x_0$ در تناقضه ← ثابت برای $y$ هم همینو می‌شه گفت.




شکلش تصور اینطوریه:

اگر $\theta > 90$ باشه که $a'b$ از $ab$ کوتاه‌تره.

وگرنه عمودش داخلشه.




الان $|br| < |ab|$

( $a, a'$ توی $C_x$ پس $r$ هم توی $C_x$ ع )

ب.

چون یونیفرم رندوم انتخاب کردیم.

از آنجایی که به احتمال ۱ نقاط در general position اند و داریم که VC لینیر کلسیفایر در فضای d بعدی d+1 است یعنی d+1 نقطه را shatter می‌کند. بنابراین به احتمال 1 همه حالت‌های رنگ‌بندی نقاطمان جدایی پذیرند. ∎

روش بازگشتی:

حالت کلی را برای p+1 نقطه در فضا $\mathbb{R}^n$ در نظر می‌گیریم که رندوم قرار گرفته‌اند، پس به احتمال 1 در جنرال پوزیشن اند. حال داریم که یک نقطه خاص را در نظر بگیریم و جدا کنیم. برای k نقطه در فضای m بعدی $F(m,k)$ صفحات جدا کننده حالات مختلف رنگ‌بندی p نقطه دیگر را دو دسته می‌کنیم.

۱. آنهایی که می‌توانند از نقطه جدا شده بگذرند. ۲. آنهایی که نقطه p+1 ام در یک طرفش است.

$\alpha$ ← 2 حالت رنگ‌بندی درست — $\beta$ ← 1 حالت رنگ‌بندی درست

★ چون جنرال پوزیشن اند پس $p \leqslant N$ دیگر نمی‌شد.

$$\rightarrow \underset{(\text{معادل یک بعد کمتر})}{2\alpha+\beta} = \alpha + (\alpha+\beta)$$

$\alpha$ : یعنی فضای صفحات جدا کننده از نقطه که چون در general position اند یعنی ← $F(n-1,p)$

$\alpha+\beta$ : کل صفحات جدا کننده در فضای n بعدی ← $F(n,p)$

* زیرفضاها هم جنرال پوزیشن اند و گرنه در فضای اصلی نبودند در نتیجه شرایط در الگوریتم حفظ می‌شد.

$$\rightarrow F(n,p+1) = F(n-1,p) + F(n,p)$$

در حالت کلی برای ۲ یا ۳ نقطه در فضای ۳ بعدی همه حالات جدا می‌شد ← $F(2,2)=F(2,3)=2^3$

ادعا کنیم $\underset{p\leqslant n+1}{F(n,p)} = 2^p$. فرض کنید برای تا $n=k$ داشته باشیم $\underset{p\leqslant n+1}{F(n,p)} = 2^p$

حال برای k+1 داریم که $F(k+1,p) = F(k-1,p-1) + F(k,p-1)$

از $p=1$ داریم که 1 نقطه حتماً ۲ حالتش جدا می‌شد ← $F(k,1) = 2^1 = 2$

پس اگر تا $p=p'$ درست باشد

$$\underset{p'\le k+1}{F(k+1,\, p'+1)} = \overset{p'\le k}{F(k-1,\, p')} + \overset{p'\le k+1}{F(k,\, p')}$$

$$\underset{\underset{\le k}{\text{حالت‌های قبلی}}}{=} 2^{p'} + 2^{p'} = 2^{p'+1}$$ ✓

پس طبق استقرا داریم که $F(n,p)=2^p$ که همه حالات قابل رنگ‌بندی است به احتمال ۱.

ج صفحه جدید

ج) مینیمم ۲ است.

در مسئله SVC داریم که یکی از شروط $\sum_{i=1}^{N} \alpha_i y_i = 0$ است.

که $\alpha_i$ ها ضرایب لاگرانژ در مسئله دوگان اند.

همچنین $\alpha_i$ های ناصفر همان ساپورت وکتور ها اند.

حال تعداد نقاط ساپورت وکتور ۱ نمی‌تواند باشد چون اگر مینیمم فاصله نقطه margin ها $\varepsilon$ باشد ما با افزایش مارجین به اندازه $\frac{\varepsilon}{2}$ مارجین بزرگتری داریم که متناقض ماکس مارجین یافته شده است.

پس حداقل ۲ هست. حال هم هندسی می‌توان گفت ما خط کلاسیفایر را در جهت مارجین خلاف این ساپورت وکتور گسترش بدهیم داریم که اندازه مارجین زیاد می‌شود.

نشانه دیگر این است که $\sum_{i=1}^{N} \alpha_i y_i = 0$

این ساپورت وکتور فرض کنید wlog $y_1 = 1$ آنگاه و اندیس ۱ $\alpha_1 y_1 + \sum_{i=2}^{N} \alpha_i y_i = 0$

$\underset{>0}{\alpha_1} + \sum_{i=2}^{N} \alpha_i y_i = 0$

پس نمی‌تواند تنها ساپورت وکتور باشه چون همه $\alpha_i$ های دیگه ۰ اند پس $\alpha_1 + 0 = 0$ که $\alpha_1 = 0$

اما می‌دانیم چون یه ساپورت وکتوره $\alpha_1 > 0$ . ×

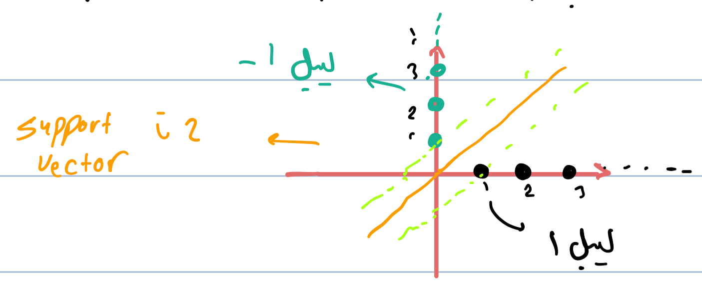
wlog اندیس ۱ و ۲

مینیمم مقدار ۲ است و طبق $\sum \alpha_i y_i = 0$ باید داشته باشیم $\alpha_1 y_1 + \alpha_2 y_2 = 0 \Leftarrow$ چون $\alpha_i > 0$

پس $y_i$ ها مختلف العلامت اند $y_1 = 1, y_2 = -1$ wlog $\alpha_1 - \alpha_2 = 0 \Rightarrow \alpha_1 = \alpha_2$ پس باید

دو ساپورت وکتورها مختلف لیبل باشند و ضریب لاگرانژشان برابر می‌شود.

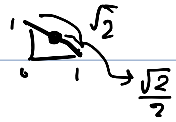
برای این مثال می‌آورم:




چرا اینجا ۲؟ چونکه حداقل ۲ و اگر بخواد بیشتر شود، می‌دانیم کاملاً جای بپذیره

حال اگر حداقل ۲ نقطه روی مارجین باشند پس مثلاً ۲ نقطه سایه روی مارجین اند

پس یک مارجین کل فقط $y=0$ و خط دیگر باید $y=1$ باشد. پس اندازه مارجین $\frac{(0-1)}{2}$

پس $\frac{1}{2}$ است. اما در خط مثال نمایش داده شده اندازه برابر 


که $\frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$ است اما $2 > \sqrt{2}$ پس $\frac{1}{\sqrt{2}} > \frac{1}{2}$ در نتیجه بهترین خط همین 2 ساپورت وکتورها.

## مسئله 2

### Entropy

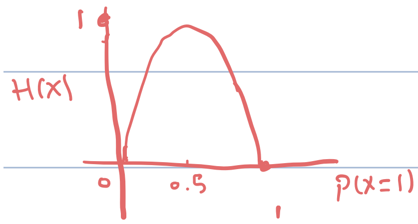
این معیار جهت اندازه‌گیری میزان information یا uncertainty مربوط به خروجی های یک متغیره.

در واقع داره از $H(X) = E(I(X)) = E(-\log P(X))$ می آید که $I(X) =$ information content می باشد.

حال در درخت تصمیم گیری به ما در جهت انتخاب بهترین split کمک می‌کنند، به این صورت که به ما معیاری برای parity بخش ها می دهد. $H(S) = -\sum P(x_i) \log P(x_i)$

* یک نشانه information theory هم می‌شه داشت که $H(X)$ امید ریاضی تعداد بیت‌های لازم در بهینه‌ترین کدینگ برای اینکد کردن

یک مقدار رندوم از X است ← $-\log P(X=i)$ بیت برای اینکود $x=i$




### Information Gain

$$Gain(S, A) = H_S(Y) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H_{S_v}(Y)$$

سمپل‌ها ← S، متغیر split ← A، متغیر تارگت ← Y

این معیار در واقع نوع تصمیم گیری‌ها بر حسب entropy در درخت تصمیم می‌باشه، یعنی مقدار کاهش انتروپی در یک split را به ما می دهد.

$$IG(T,a) = \underbrace{H(T)}_{\text{انتروپی پدر}} - \underbrace{H(T|a)}_{\text{جمع انتروپی فرزندان}}$$

$$= -\sum p_i \log p_i - \sum_a p(a) \sum_{i=1}^{J} - Pr(i|a) \log_2 Pr(i|a)$$

در هر تقسیم ما بهترین انتخابمان max IG است ←

$i$ در remaining att.

$$j = \arg\max \; Gain(S, x_i)$$
$$= \arg\max \; H_S(Y) - H_S(Y|x_i)$$
$$= \arg\min \; H_S(Y|x_i)$$

مینی‌مم رابطه با انتروپی دارد فقط تنها دلیلی که ما سراغ IG می‌رویم این است که بین

تقسیم بندی در برگ های مختلف در یک مرحله به ما اولویت می دهد چرا که حاصل از همین انتروپی به ما بیشترین کاهش را می دهد

* رابطه نزدیک (اکسپکتد) با mutual information هم دارد. $I(X,Y)=H(Y)-H(Y|X)$ اگر X نسبت به Y مستقل باشد

$P(X=i,Y=j)=P(X=i)P(Y=j)$ می شود پس $I=0$ و ما دنبال اینیم که information با لیبل ها صورت در هر مرحله به دست بیاریم پس

بیشتر یعنی برخلاف استقلال را می خواهیم.

## Gini

این اندکس هم به ما میزان پراکندگی در داده ها روی دهد و برای سنجش impurity کاربرد داره.

درواقع به ما این مقدار رو می ده که چه احتمالی یک عضو رندوم از یه مجموعه لیبل غلطی خوره اگه بصورت رندوم و مستقل طبق توزیع لیبل ها لیبل زده شده باشد. اگه همه هر نود تری یک کلاس بشوند 0 می شه که مینیمم است

$$I_G(p)=\sum_{i=1}^{J}\Big(p_i\sum_{k\neq i}p_k\Big)=\sum_{i=1}^{J}p_i(1-p_i)=\sum_{i=1}^{J}p_i-\sum_{i=1}^{J}p_i^2=1-\sum_{i=1}^{J}p_i^2$$

* ارتباط مستقیم هم با Tsallis Entropy داره.

$$\hat{p}_{mk}=\frac{1}{\underbrace{N_m}_{\text{سایز}}}\sum_{\underbrace{x_i\in R_m}_{\text{منطقه تقسیم شده توسط درخت}}} I(\overset{\text{لیبل}}{y_i}=k)$$

دو نکته جالب به این اندکس هست. اول تعریف می کنم که

$\rightarrow$ Gini $\quad \sum_{k\neq k'}\hat{p}_{mk}\hat{p}_{mk'}=\sum_{k\neq k'}\hat{p}_{mk}(1-\hat{p}_{mk})$

نکته اول: بیایم جای کلسیفای به کلاس عمده درک نود، به کلاس k با احتمال $\hat{p}_{mk}$ اختصاص بدیم. اکسپکتد ارور دیتای ترین می شه gini index (همونی که قبلاً گفتم) نکته دوم: برای کلاس k هر آبزرویشن رو 1 کد کنیم وگرنه 0. واریانسش می شه $\hat{p}_{mk}(1-\hat{p}_{mk})$ ، حالا برای کل k ها ببینیم همون gini هستش.

## ب

مینیمم gini همانطور که گفتم اگر مربوط به یک کلاس باشند. $1\times 0=0$ می باشد.

اما بدترین حالت این است که بیشترین impurity را داریم یعنی از هر کلاس به اندازه یکسان باشند پس اگر c کلاس باشه

$$\sum p_i(1-p_i)=\sum_{c}\frac{1}{c}\Big(1-\frac{1}{c}\Big)=\sum_{c}\frac{1}{c}-\sum_{c}\frac{1}{c^2}=c\Big(\frac{1}{c}-\frac{1}{c^2}\Big)=1-\frac{1}{c}$$

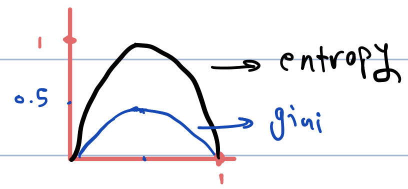
پس برای c کلاس بازه gini بصورت $[0, 1-\frac{1}{c}]$ است و درکل اگر $c\to\infty$ ، $[0,1)$

برای دو کلاس $[0,\frac{1}{2}]$

بازه Entropy : بهترین و pure ترین همه یک کلاس باشند ← $-(1\times \overset{0}{\log 1}) = 0$

و بدترین و impure ترین همه برابر داریم که $-\sum \frac{1}{c}\log_2\frac{1}{c} = -\frac{1}{c}\sum -\log_2 c = \frac{1}{c}\, c\log_2 c = \log c$

پس $[0, \log_2 c]$ می‌باشد. که c تعداد کلاس هاست.




درحالت دو کلاسه :

c تعداد کلاس

از لحاظ محاسباتی چون log دارد entropy ($O(c\log c)$) دشوار تر است ولی gini ($O(c)$) صرفاً درجه ۲ هم دارد.

مثل زمانشون انتروپی نسبت به gini روبابست تره و کمتر تحت تأثیر نویز قرار می‌گیره.

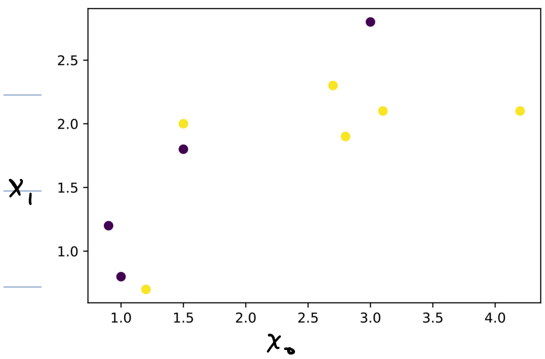
و به نسبه انتروپی به تعداد کلاس ها از gini حساس تره و gini به توزیع کلاس ها ، درواقع جاهایی که خیلی imbalanced هستند gini عملکردش افت داره. انتروپی بایاس سمت تقسیمی که uncertainty در بیشتر کاهش بده داره ولی gini به سمت توزیع بالانس تر.

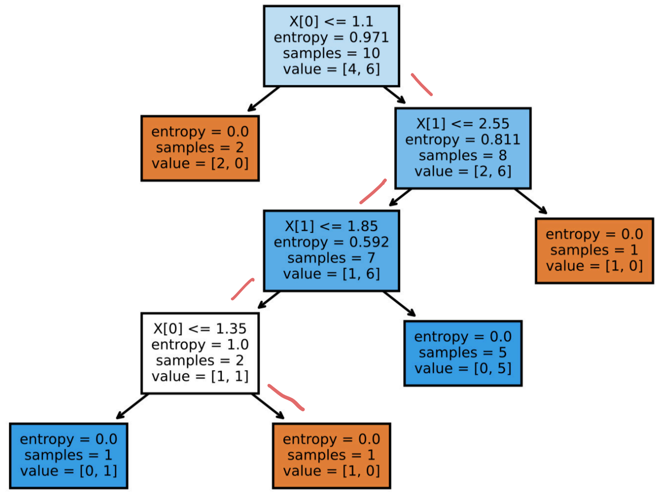
• ج

برای مقایسه درخت با entropy از تکه کد زیر بهره می‌برم.

```python
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
x1 = np.array([1.5,3,1.2,4.2,1.5,0.9,3.1,2.7,1.0,2.8])
x2 = np.array([2,2.8,0.7,2.1,1.8,1.2,2.1,2.3,0.8,1.9])
y= np.array([1,0,1,1,0,0,1,1,0,1])
data = np.array([x1,x2])
data = data.T
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(data,y)
plt.scatter(x1,x2,c=y)
plt.savefig('scatterEntropy.pdf')
plt.show()
tree.plot_tree(clf, filled=True,fontsize=6)
plt.savefig('treeEntropy.pdf')
```

همچنین کلاس بندی درخت بصورت زیر است. درخت بدست آمده به شرح زیر است.



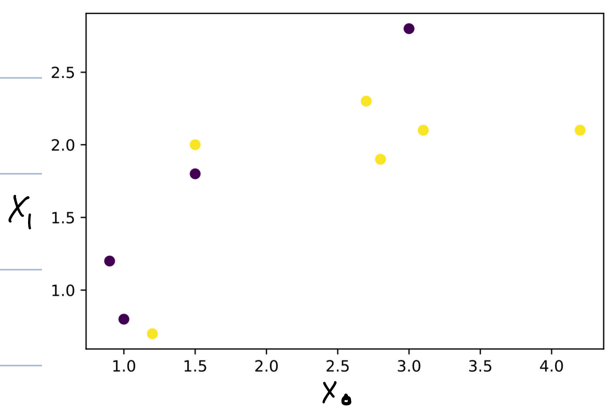



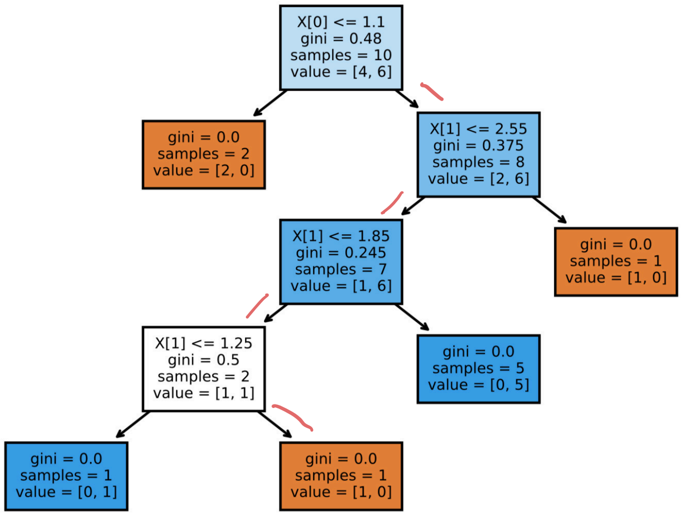

برای $x_1=x_2=1.5$ داریم که طبق مسیر قرمز است ← بنفش یا $y=0$ می باشد.

برای gini از کد زیر بهره میبرم.

```
clf = DecisionTreeClassifier(criterion='gini')
clf.fit(data,y)
plt.scatter(x1,x2,c=y)
plt.savefig('scatterGini.pdf')
plt.show()
tree.plot_tree(clf, filled=True,fontsize=6)
plt.savefig('treeGini.pdf')
```

حال درخت تصمیم گیری و نتیجه تصمیم متناظر به شرح زیر اند.




مسیر $x_1=x_2=1.5$ را با رنگ قرمز نشان دادم و بنفش یا $y=0$ است.

مشاهده کردیم که با انتخاب entropy یا gini فرقی در تصمیم گیری نشد و هردو ۰ شدند.

برای نمایش نحوه تقسیم بندی نیز کد زیر را جهت پرینت entropy های تصمیم گیری زدم :

```python
class DecisionTree:

    def __init__(self, max_depth=None):
        self.max_depth = max_depth
        self.tree = {}

    def _compute_entropy(self, y):
        classes, counts = np.unique(y, return_counts=True)
        probabilities = counts / len(y)
        entropy = -np.sum(probabilities * np.log2(probabilities))
        return entropy

    def _split_data(self, X, y, feature_index, threshold):
        left_mask = X[:, feature_index] <= threshold
        right_mask = ~left_mask
        left_data, right_data = X[left_mask], X[right_mask]
        left_labels, right_labels = y[left_mask], y[right_mask]
        return left_data, right_data, left_labels, right_labels

    def _find_best_split(self, X, y):
        best_feature_index, best_threshold, best_entropy = None, None, float('inf')
        for feature_index in range(X.shape[1]):
            thresholds = np.unique(X[:, feature_index])
            for threshold in thresholds:
                left_data, right_data, left_labels, right_labels = self._split_data(X, y, feature_index, threshold)

                if len(left_data) == 0 or len(right_data) == 0:
                    print("one side 0")
                    continue
                total_entropy = len(left_labels) / len(y) * self._compute_entropy(left_labels) + \
                                len(right_labels) / len(y) * self._compute_entropy(right_labels)
                print(f"total entropy: {total_entropy}")
                if total_entropy < best_entropy:
                    best_feature_index, best_threshold, best_entropy = feature_index, threshold, total_entropy
            print(f"feature index: {feature_index}, best threshold: {best_threshold}, best entropy: {best_entropy}")
        print(f"len x = {len(X)} ,best feature index: {best_feature_index}, best threshold: {best_threshold}, best entropy: {best_entropy}")
        return best_feature_index, best_threshold

    def _build_tree(self, X, y, depth):
        if depth == 0 or len(np.unique(y)) == 1:
            return np.bincount(y).argmax()

        best_feature_index, best_threshold = self._find_best_split(X, y)
        if best_feature_index is None:
            return np.bincount(y).argmax()

        left_data, right_data, left_labels, right_labels = self._split_data(X, y, best_feature_index, best_threshold)
        if len(left_data) == 0 or len(right_data) == 0:
            return np.bincount(y).argmax()

        left_subtree = self._build_tree(left_data, left_labels, depth - 1)
        right_subtree = self._build_tree(right_data, right_labels, depth - 1)

        return {'feature_index': best_feature_index, 'threshold': best_threshold,
                'left': left_subtree, 'right': right_subtree}

    def fit(self, X, y):
        self.tree = self._build_tree(X, y, self.max_depth)

    def _predict_sample(self, x, tree):
        if not isinstance(tree, dict):
            return tree
        feature_index, threshold = tree['feature_index'], tree['threshold']
        if x[feature_index] <= threshold:
            return self._predict_sample(x, tree['left'])
        else:
            return self._predict_sample(x, tree['right'])

    def predict(self, X):
        return np.array([self._predict_sample(x, self.tree) for x in X])
```

```
total entropy: 0.8264662506490407
total entropy: 0.6490224995673063
total entropy: 0.8796731482129885
total entropy: 0.8464393446710154
total entropy: 0.9245112497836532
total entropy: 0.965148445440323
total entropy: 0.8
total entropy: 0.8919684538544
one side 0
feature index: 0, best threshold: 1.0, best entropy: 0.6490224995673063
total entropy: 0.8919684538544
total entropy: 0.963547202339972
total entropy: 0.8796731482129885
total entropy: 0.7145247027726656
total entropy: 0.8464393446710154
total entropy: 0.9245112497836532
total entropy: 0.963547202339972
total entropy: 0.8264662506490407
one side 0
feature index: 1, best threshold: 1.0, best entropy: 0.6490224995673063
len x = 10 ,best feature index: 0, best threshold: 1.0, best entropy: 0.6490224995673063
total entropy: 0.7552304974958022
total entropy: 0.7955659970750351
total entropy: 0.8112781244591328
total entropy: 0.7955659970750351
total entropy: 0.6887218755408672
total entropy: 0.7552304974958022
one side 0
feature index: 0, best threshold: 3.0, best entropy: 0.6887218755408672
total entropy: 0.7552304974958022
total entropy: 0.7375168162362655
total entropy: 0.7955659970750351
total entropy: 0.8112781244591328
total entropy: 0.7375168162362655
total entropy: 0.5177136812595365
one side 0
feature index: 1, best threshold: 2.3, best entropy: 0.5177136812595365
len x = 8 ,best feature index: 1, best threshold: 2.3, best entropy: 0.5177136812595365
total entropy: 0.5571620756985892
total entropy: 0.39355535745192405
total entropy: 0.46358749969093305
total entropy: 0.5156629249195446
total entropy: 0.5571620756985892
one side 0
feature index: 0, best threshold: 1.5, best entropy: 0.39355535745192405
total entropy: 0.5571620756985892
total entropy: 0.2857142857142857
total entropy: 0.39355535745192405
total entropy: 0.46358749969093305
total entropy: 0.5571620756985892
one side 0
feature index: 1, best threshold: 1.8, best entropy: 0.2857142857142857
len x = 7 ,best feature index: 1, best threshold: 1.8, best entropy: 0.2857142857142857
total entropy: -0.0
one side 0
feature index: 0, best threshold: 1.2, best entropy: -0.0
total entropy: -0.0
one side 0
feature index: 1, best threshold: 1.2, best entropy: -0.0
len x = 2 ,best feature index: 0, best threshold: 1.2, best entropy: -0.0
```

مسئله 3

• الف)

cubic spline →

هر چندجمله‌ای درجه 3، 4 ضریب دارد پس 4 درجه آزادی.

به تعداد knot ها شرط برابری نقطه و مشتق اول و دوم داریم ← 3 تا.

⇒ $k$ knot → $k+1$ polynomial

$$\Rightarrow \quad 4(k+1) - 3k = 4k+4-3k = k+4$$

پس $df = k+4$

* برای داشتن basis آن به شکل

$h_j(x) = x^{j-1} \quad j = 1, \dots, M$

spline

$h_{m+\ell}(x) = (x-\xi_\ell)_+^{M-1} \quad \ell = 1, \dots, k$

$\xi_\ell$ ↳ knot

→ مشتق پیوسته تا $M-2$

پس برای cubic داریم $M=4$

$$\Rightarrow \quad x^0, x^1, x^2, x^3, \{(x-\xi_i)^3 \mid i = 1, \dots, k\} \Rightarrow \text{ تا } k+4$$

* در حالت کلی لزومی که knot ها مرتب باشند نیست.

natural :

فرض ثابت برای قبل و بعد knot ها داریم.

بنابراین از دو spline اول و آخر دو درجه آزادی کاسته می‌شود (درجه 3 ← 1 ، 4 پارامتر ← 2)

پس $df = k+4-4 = k$

basis $\quad N_1(x) = 1 \quad N_2(x) = x \quad N_{k+2}(x) = d_k(x) - d_{k-1}(x) \quad d_k(x) = \dfrac{(x-\xi_k)_+^3 - (x-\xi_K)_+^3}{\xi_K - \xi_k}$

basis ها هم $k$ پارامتر می‌شه ← ☺

• ب

پس اینجا smoothing spline داریم و $RSS(\lambda, f) = \sum_{i=1}^{N}\{y_i - f(x_i)\}^2 + \lambda\int\{f''(t)\}^2 dt$

$\lambda$ به سمت ∞ برود، مشتق دوم را نشان ندهد tolerate کرد پس جواب به سمت خطی بودن

و دو وا مع همان جواب کمینه مربعات عادی می رود.

تحلیل ریاضی طور تر:

$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j \xrightarrow{} n \text{ dim natural spline}$

$N_{ij} = N_j(x_i)$

ماتریس $(\Omega_N)_{jk} = \int N_j''(x) N_k''(x)\, dx$

$\Rightarrow RSS(\theta, \lambda) = (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega_N\theta$

مشتق بر

$\Rightarrow \hat{\theta} = (N^TN + \lambda\Omega_N)^{-1}N^Ty \Rightarrow \hat{f}(x) = \sum N_j(x)\hat{\theta}_j = N\hat{\theta} = S_\lambda y$

$S_\lambda = N(N^TN + \lambda\Omega_N)^{-1}N^T$

$* \; S_\lambda = N(N^T(I + N^{-T}\lambda\Omega_N N^{-1})N)^{-1}N^T = (I + \lambda \underbrace{N^{-T}\Omega_N N^{-1}}_{K})^{-1} = (I + \lambda K)^{-1}$

می‌شه نوشت

$\rightarrow \hat{f} = S_\lambda y = \min_f (y-f)^T(y-f) + \lambda f^T K f$

فرم svd:

$S_\lambda = \sum_{k=1}^{N} P_k(\lambda) u_k u_k^T$

داریم که

s.t $P_k(\lambda) = \dfrac{1}{1 + \lambda d_k}$

مقدار ویژه k هستش

* چون مشتق دوم بیس ها صفرن ، پس $d_1 = d_2 = 0$ و shrink اینجا رخ نمی‌ده ولی بقیه $|d_i| > 0$

$\Rightarrow \quad P_0(\lambda) = \frac{1}{1+\lambda 0} = 1 \quad , \quad P_1(\lambda) = \frac{1}{1+\lambda 0} = 1$

حال اگر $\lambda \to \infty$ داریم که $P_{k>0}(\lambda) = \frac{1}{1+\lambda d_k} \to \frac{1}{\infty} = 0$ پس همه ۰ می‌شوند بجز دوتایی که $u_1$ و $u_2$ بودند و تقریبه خطی‌مان بودن ← رفتن $\lambda \to \infty$ داریم به جواب خطی کمینه رگرسیون میل می‌کنه.

• ج) ۱

روش اول: n بعدی را به n متغیر می‌نگاریم. حال برای هر یک از متغیرها (بعدها) متغیر iام مانند روش تک‌بعدی spline را مشخص می‌کنیم و basis function های آن را $h_i$ در نظر می‌گیریم.

یعنی برای spline $M_i$ پارامتری اش: $h_{i1}(X_i), \dots, h_{iM_i}(X_i)$

← بعد iام X.

basis ها اند.

برای تبدیل به مثلاً دو بعد تعریف می‌کنیم

$$g_{jk}(X) = h_{1j}(X_1)\, h_{2k}(X_2)$$

$$j = 1, \dots, M_1 \quad k = 1, \dots, M_2$$

و داریم که

$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk}\, g_{jk}(X)$$

این روش مشکلش اینه که dim بصورت exponential داره زیاد می‌شه.

مثلاً برای N بعدی ما $g_{i_1 \dots i_N}(X) = h_{1 i_1}(X_1) \dots h_{N i_N}(X_N)$ داریم و $df = M_1 \times M_2 \times \dots \times M_N$

اما در عمل نسبت به روش دوم (additive) منعطف‌تره و ممکنه ساختارهای پیچیده‌تر را بسازه.

## روش دوم ( Additive

یه حالت خاصی از روش بالا وقتی مثل smoothing براش پنالتی بیاریم.

$$f(X) = \alpha + f_1(X_1) + \dots + f_d(X_d)$$

به شکل جمع d تا spline تک بعدی می‌نویسیمش در این صورت اندازه روش یکی ساده‌ترنه ساختار complex میباشه ولی

منهای درجه آزادی ثابت ← ترم     عرض از مبدا

$$df = \sum (M_i - 1) + 1$$

$$\sum_{i=1}^{d} M_i - (d-1)$$

* $h_i$ ها که گرفتیم را به روش های مختلف می‌شه گرفت یکی ⊛ دو صفحه قبله

یه روش بهتر اینه که از B-spline استفاده کنیم. $\xi_0 < \xi_1$ و $\xi_k < \xi_{k+1}$ bound ها knot ها بصورت

میگیریم. حال $\tau_1 \le \dots \le \tau_M \le \xi_0$     $\xi_i$ همون قبلیا knot ها

$$\tau_{j+M} = \xi_j \quad j = 1, \dots, k$$

$$\xi_{k+1} \le \tau_{k+M+1} \le \dots \le \tau_{k+2M}$$

basis order $m$

$$B_{i,m}(x)$$

$$B_{i,1}(x) = \begin{cases} 1 & \tau_i \le x < \tau_{i+1} \\ 0 & o.w \end{cases} \qquad \forall\, i = 1, \dots, k+2M-1$$

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x)$$

$i = 1, \dots, k+2M-m$

خود $\xi$ قبلی‌ها هم برحسب توزیع یا مینیمم یا ویتر پراکندگی مثل quantile ها

میزاریم. ⟵ که ضرب این knot ها کل فضای $B_i$ و

درنتیجه ضرب ها یعنی n تا knot ها را در multi-dimensional داریم.

## مسئله 4

• الف) E

در بین حالات ارائه شده تنها D و E هستند که کلسیفیکیشن درستی برای تمای نقاط دیتای ترین دارند
در بین این دوتا نیز D بسیار نقاط بیشتری به عنوان support vector دارد که این به ما خطر اورفیت
شدن و عملکرد نامناسب روی کل توزیع (دیتای تست) را می دهد بنابراین بهترین انتخاب E است.

ب)

1. radial basis 0.08 — D
2. radial basis 0.5 — E
3. radial basis 2 — B
4. linear — A
5. second order polynomial — C

علت:

دو انتخاب خطی و درجه دو نسبتاً واضح اند. A مرز خطی ای دارد بنابراین انتخابش با linear است.
همینطور C نیز ساختار مارجینش به توابع درجه ۲ نزدیک است و ساختار جمع گاوسی radial دیده نمیشود بنابراین
انتخاب second order polynomial است.
حال تنها ساختار جمع گاوسی ها فربیار ساپورت وکتور ها را دارند بعضی smooth تر و بعضی نیز smooth تر
پس سه گزینه دیگر radial اند.
حال برای انتخاب ها داریم که هرچه σ به سمت صفر برود به سمت اورور ترینینگ ۰ و اورفیتینگ با
ساپورت وکتور شدن همه نقاط میرویم.

همچنین هرچه $\sigma$ کوچک می‌شود با افزایش ساپورت وکتورها و دخیل شدن کارسی‌های بیشتر مرز پیچیده (wiggly) تری داریم.

علاوه بر اینها یک نکته این است که با فرض ثابت بودن C داریم که با کوچک شدن $\sigma$ که در نتیجه ساپورت وکتورها زیاد می‌شوند به‌ندی مارجین گسترده‌تری داریم.

حال در شکل داریم که تعداد ساپورت وکتورها $\underset{9}{D} > \underset{5}{E} > \underset{4}{B}$ و همچنین از لحاظ پیچیدگی مرز (همچنین گستردگی مارجین داخل شکل) D بسیار پیچیده‌تر از دوتای دیگه است.

در نتیجه برداشت این است که $\sigma_B > \sigma_E > \sigma_D$ باشند و طبق این انتخاب کردم که

radial sigma 2 ← B , radial sigma 0.5 ← E , radial sigma 0.8 ← D

ج •

$\underline{*}$ $w = \sum_{n=1}^{N} \alpha_n y_n x_n$ , $\underline{0}$ $\sum_{n=1}^{N} \alpha_n y_n = 0$

$\underline{*}$ non-support-vector $\rightarrow \alpha_i = 0 \Rightarrow w = \alpha_1 y_1 x_1 + \alpha_2 y_2 x_2$

$\Rightarrow w = \alpha_1 (+1) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 (-1) \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -\alpha_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ -\alpha_2 \end{bmatrix}$

$\underline{0}$ $\alpha_1 y_1 + \alpha_2 y_2 = 0 \Rightarrow \alpha_1 \times 1 + \alpha_2 \times (-1) = 0 \Rightarrow \alpha_1 = \alpha_2 \rightarrow w = \begin{bmatrix} \alpha_1 \\ -\alpha_1 \end{bmatrix}$

سمپل های که روی مارجین اند $(\alpha_s > 0)$ داریم که: $y^{(s)} (w^T x^{(s)} + w_0) = 1$

$$\Rightarrow \begin{cases} w^T x_1 + w_0 = 1 \\ w^T x_2 + w_0 = -1 \end{cases} \rightarrow \begin{cases} [\alpha_1 \ \ -\alpha_1] \begin{bmatrix} 1 \\ 0 \end{bmatrix} + w_0 = 1 \\ [\alpha_1 \ \ -\alpha_1] \begin{bmatrix} 0 \\ 1 \end{bmatrix} + w_0 = -1 \end{cases} \Rightarrow \begin{cases} \alpha_1 + w_0 = 1 \\ -\alpha_1 + w_0 = -1 \end{cases}$$

جمع : $2w_0 = 0 \Rightarrow w_0 = 0$

با جاگذاری $\alpha_1 + 0 = 1 \Rightarrow \alpha_1 = 1$

$\Rightarrow w = \begin{bmatrix} \alpha_1 \\ -\alpha_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ , $w_0 = 0$ $\quad w = \alpha_1 y_1 x_1 + \alpha_2 y_2 x_2 = 1 \times 1 \times x_1 + 1 \times (-1) \times x_2 = x_1 - x_2$

$$\Rightarrow \boxed{\begin{aligned} w &= x_1 - x_2 \\ b &= w_0 = 0 \end{aligned}}$$

مسئله 5

• الف)

1- T یا تعداد boosting round ها است. در مثال الگوریتم ها یپر پارامترهایی که باید تعیین شوند

بوستینگ درخت تصمیم تعداد درخت هایی که می سازیم.

2- پارامتر d که vc-dim کلاسیفایرهای weak T که می سازیم. به نوعی در الگوریتم بوستینگ درخت تصمیم

همان d که برای تعداد split ها انتخاب می‌شود.

3- λ که shrinkage parameter است و ریت یادگیری الگوریتم را مشخص می کند.

پارامتر T که تعداد boosting round ها یا در درخت تصمیم تعداد درخت ها باشد با زیاد شدن (از حد زیاد) در

در روش boosting موجب اورفیت می شود ولی این پارامتر که در bagging هم هست موجب چنین

امری نمی‌شود و خطا ارور ریتمان با افزایش T استیبل می‌شود.

در واقع در روش bagging کاربری این T با روش boosting فرق دارد. در bagging ما هر مولفه‌ها

bias کم و واریانس زیاد دارد و با استفاده از T قصد داریم تا حدودی همانند

$$Var\left(\frac{X_1+\dots+X_n}{n}\right)=\frac{\sigma^2}{n}$$

$Var(X): \sigma^2$

با میانگین گیری واریانس را کاهش دهیم. پس به ست عبارت بر bias کم کاهش واریانس می‌بریم

ولی در boosting ما هر مولفه بایاس زیاد و واریانس کمی دارد و به نوعی اورفیت رو عکس حالت ازمفها

اند و قصد داریم با T بیشتر بایاس را کاهش ببریم.

حالت اورفیت زمانیه که ما بایاس کمی داریم و واریانس زیاد و در روش bagging ما به سمت واریانس کم می‌ریم

ولی برخلاف آن در boosting هرچه T بیشتر داریم بایاس را کم می کنیم ولی از جایی به بعد کاهش بایاسمان همراه افزایش

واریانسه که موجب overfit می‌شه.

* همچنین در boosting bound که از Freund, Schapire '95 داریم برای ارور تست $E_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{N}}\right)$

با احتمال بالا هستش که به ما نشانه‌هایی از افزایش اختلاف ارورها و اورفیت درصورت خیلی زیاد شدن T می دهد.

ب •

مقایسه متودی : در Random forest ما همانند bagging از تعداد زیاد مثلاً T درخت استفاده می کنیم و میانگین می گیریم (یا در کلسیفیکیشن رای عمده) و برای بهبود کاهش واریانسمان بخصوص در کاهش کورلیشن درخت ها در هر split هر درخت m تا رندم (مثلاً $\sqrt{p}$ ← # فیچرها) از فیچرها انتخاب و بین آنها بهینه را می یابیم.

یک عامل برتری درخت ساده : درخت ساده interpretable تر است و به آسانی می توان نحوه تصمیم گیری را مشاهده کرد درحالیکه RF چنین نیست و برای درک جامع تاثیر هر فیچر در کل مدل را بررسی کنیم و قابلیت درک مستقیم برای حالاتی مثل کاربردهای پزشکی مدل کمتر دارد.

عامل ضعف : درخت تصمیم احتمال اورفیت شدن بیشتری دارد چون می تواند به آسانی زیادی complex شود و واریانس زیادی دارد. اما در RF با کمک از روش bagging روی درخت‌ها با کورلیشن کم این واریانس را کاهش داده و از اورفیت جلوگیری می کنیم.

و در کل عموماً نسبت به درخت تصمیم پرفورمنس بهتری را داشته باشد (به علت استفاده از درخت‌ها متعدد و کاهش واریانس)

* درخت تصمیم معمولاً بایاس کمی دارد که ما در RF از تعداد زیاد از همین مولفه‌ها با بایاس کم بهره می بریم)