Topic 1 Discriminant Function

So far, we are not strange to classification. In "Probability" chapter, we have know MLE and MAP. They can be used to do classfication. We will introduce another method which uses linear model.

1. Definition of Discriminant Function.

Discriminant is a function which turns input $\underline{x}$ to one of K classes, denoted C_k. Let's limit ourselves to linear discriminants at present.

Our model is

$$y(\underline{x}) = f(\underline{w}^T\underline{x} + b)$$

f: active function, can be non-linear

$\underline{w}^T\underline{x} + b$: linear discriminat function.

It is true that, because of "$f(\cdot)$", our model is non-linear. But if we see the $\underline{x}$ that

$f(\underline{w}^T\underline{x} + b) = \text{const}$, all $\underline{x}$ satisfying the condition will form up a linear hyperplane.

f: monotonous function

@ Two classes.

Aka, $K=2$.

Let's start from the simplest model.. $\underline{x} \in \mathbb{R}^D$

$$y(\underline{x}) = \underline{w}^T \underline{x} + b$$

Our classification strategy is

$$\begin{cases} y(\underline{x}) \leq 0 \rightarrow C_1 \\ y(\underline{x}) > 0 \rightarrow C_2 \end{cases}$$

$\{\underline{x} : y(\underline{x}) = 0\}$ will form up a decision hyperplane of $D-1$ dimension

Suppose $\underline{x}_A, \underline{x}_B$ are points $y(\underline{x}_A) = y(\underline{x}_B) = 0$

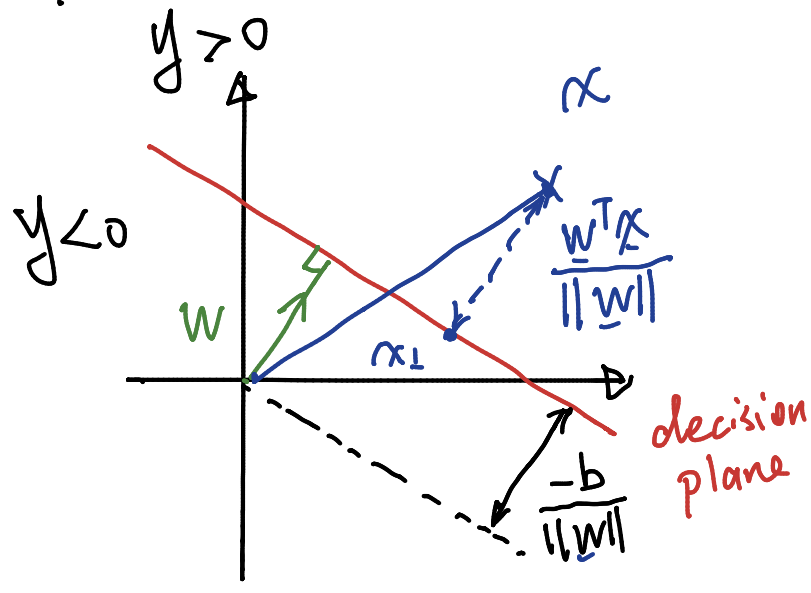
$\underline{w}^T(\underline{x}_A - \underline{x}_B) = 0 \Leftrightarrow \underline{w} \perp \underline{x}_A - \underline{x}_B$

$\vec{w}$ is orthogonal to any vector within decision plane.

$\Leftrightarrow \vec{w}$ is the normal vector of decision plane.

Also, if $\underline{x}$ is on decision plane

$$\frac{\underline{w}^T \underline{x}}{\|\underline{w}\|} = \frac{-b}{\|\underline{w}\|}$$



If the projection of $\underline{x}$ onto decision plane is $\underline{x}_\perp$, then

$$\underline{x} = \underline{x}_\perp + \underline{r} \cdot \frac{\underline{w}}{\|\underline{w}\|}$$

$$y(\underline{x}) = \underline{w}^T \underline{x} + b$$

$$y(\underline{x}_\perp) = \underline{w}^T \underline{x}_\perp + b$$

$$\underline{r} = \frac{y(\underline{x})}{\|\underline{w}\|}$$

$\underline{r}$: The distance between $\underline{x}$ and decision plane.

Let's use homogenious coordinate the linear function

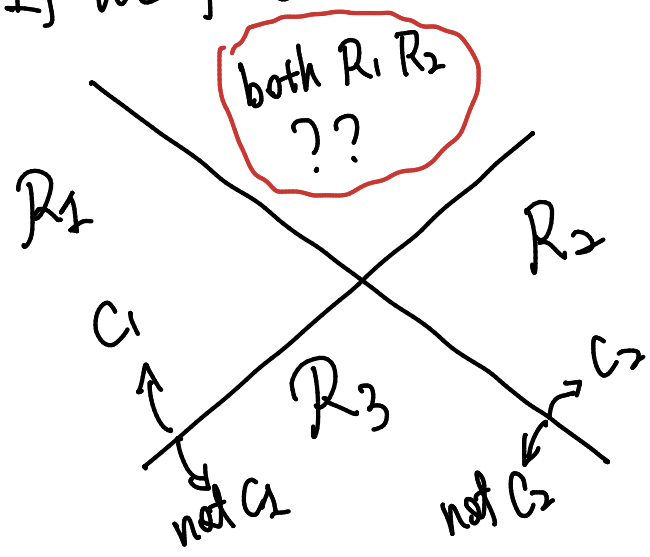
$$y(\underline{x}) = [\underline{w}, b] \begin{bmatrix} \underline{x} \\ 1 \end{bmatrix}$$

---

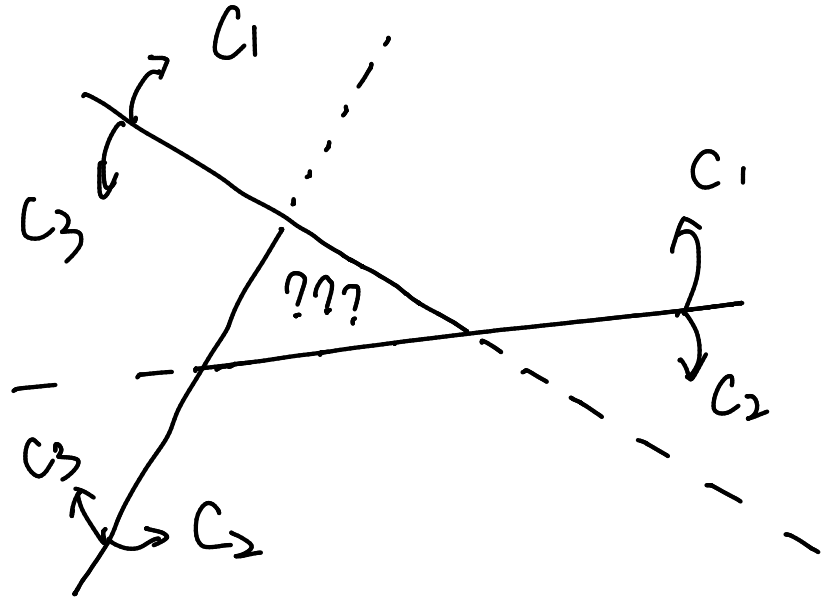
2. Multiple Class.

When $K > 2$, things turn to be complicated

If we follow our method in section 1, we may have



We have a region can be both C_1 & C_2.



If we create decision boundary for every pair of class, problems still happen, but in another mode.

We need a compond descriminant function!

We have a series of discriminant function

$$y_k(\underline{x}) = \underline{w}_k^T \underline{x} + b_k \qquad k = 1, \cdots, K$$

we assign $\underline{x}$ to class C_k iff $y_k(\underline{x}) > y_j(\underline{x}) \quad j \neq k$.

Let's investigate the decision boundary.

for C_k & C_j

$$(\underline{w}_k - \underline{w}_j)^T \underline{x} + (b_k - b_j) = 0$$

$$\hookrightarrow \underline{w}'^T \underline{x} + b' = 0 \quad \text{same form as 2-class case.}$$

According to knowledge from Convex set, the decision region is a union of several half plane, so it is convex.

We can prove this by defination

Proof:

Suppose we heve $\underline{x}_A, \underline{x}_B$ in R_k. According to defination,

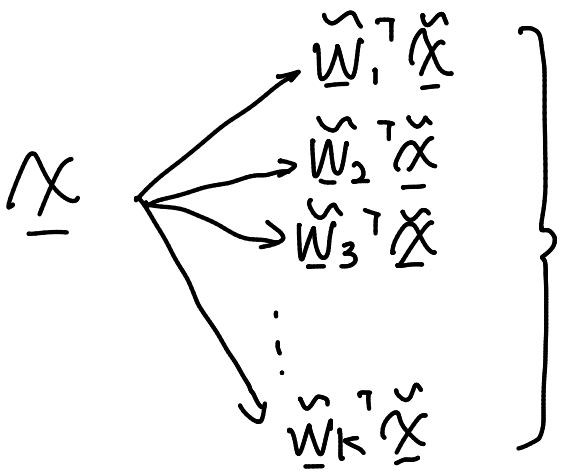
$$\hat{\underline{x}} = \lambda \hat{\underline{x}}_A + (1-\lambda) \hat{\underline{x}}_B \qquad \lambda \in [0,1]$$

From linearity

$$y(\hat{\underline{x}}) = \lambda y(\hat{\underline{x}}_A) + (1-\lambda) y(\hat{\underline{x}}_B) \Rightarrow y_k(\hat{\underline{x}}) < y_j(\hat{\underline{x}}) \Rightarrow \hat{\underline{x}} \text{ still in } R_k$$

3. Least Square for classification

Let's claim our strategy.



Suppose we have a dataset $\{\underline{x}_n, \underline{t}_n\}$ $n = 1, \cdots N$.

$$\tilde{X} = \begin{bmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_N^T \end{bmatrix} \quad T = \begin{bmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_N^T \end{bmatrix} \quad \tilde{W} = \begin{bmatrix} w_1 & w_2 & \cdots & w_K \\ & & & \vdots \\ b_1 & b_2 & & b_K \end{bmatrix}$$

Let's choose the following sum-of-squares error function

$$E_D(\tilde{W}) = \frac{1}{2} Tr\{(\tilde{X}\tilde{W} - T)^T(\tilde{X}\tilde{W} - T)\}$$

$$= \frac{1}{2}\{Tr\{\tilde{W}^T\tilde{X}^T\tilde{X}\tilde{W}\} - Tr\{\tilde{W}^T\tilde{X}^T T\} - Tr(T^T\tilde{X}\tilde{W}) + \underbrace{Tr(T^T T)}_{\text{No } \tilde{W}}\}$$

Same

$$\frac{\partial}{\partial \tilde{W}} E_D(\tilde{W}) = \frac{1}{2}\frac{\partial}{\partial \tilde{W}} Tr\{\tilde{W}^T\tilde{X}^T\tilde{X}\tilde{W}\} - \frac{\partial}{\partial \tilde{W}} Tr\{T^T\tilde{X}\tilde{W}\}$$

$$= \tilde{X}^T\tilde{X}\tilde{W} - \tilde{X}^T T = 0 \Rightarrow \tilde{W}_{LS} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T T$$

$$= \tilde{X}^\dagger T$$

$\therefore y(\underline{x}) = T^T(\tilde{X}^\dagger)^T\tilde{x}$

An Interesting property is that.

if we can find $\underline{a}, b$ that

$$\underline{a}^T \underline{t}_j + b = 0 \quad j = 1, \cdots, N \text{ then}$$

$$\underline{a}^T \underline{y}(\underline{x}) + b = \underline{a}^T T^T (\tilde{X}^+)^T \tilde{\underline{x}} + b$$

Recal that

$T^T = [t_1, t_2, \cdots t_N]$, so $a^T T^T = -b \mathbb{1}_{1 \times N}$

$$(\tilde{X}^+)^T \tilde{\underline{x}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

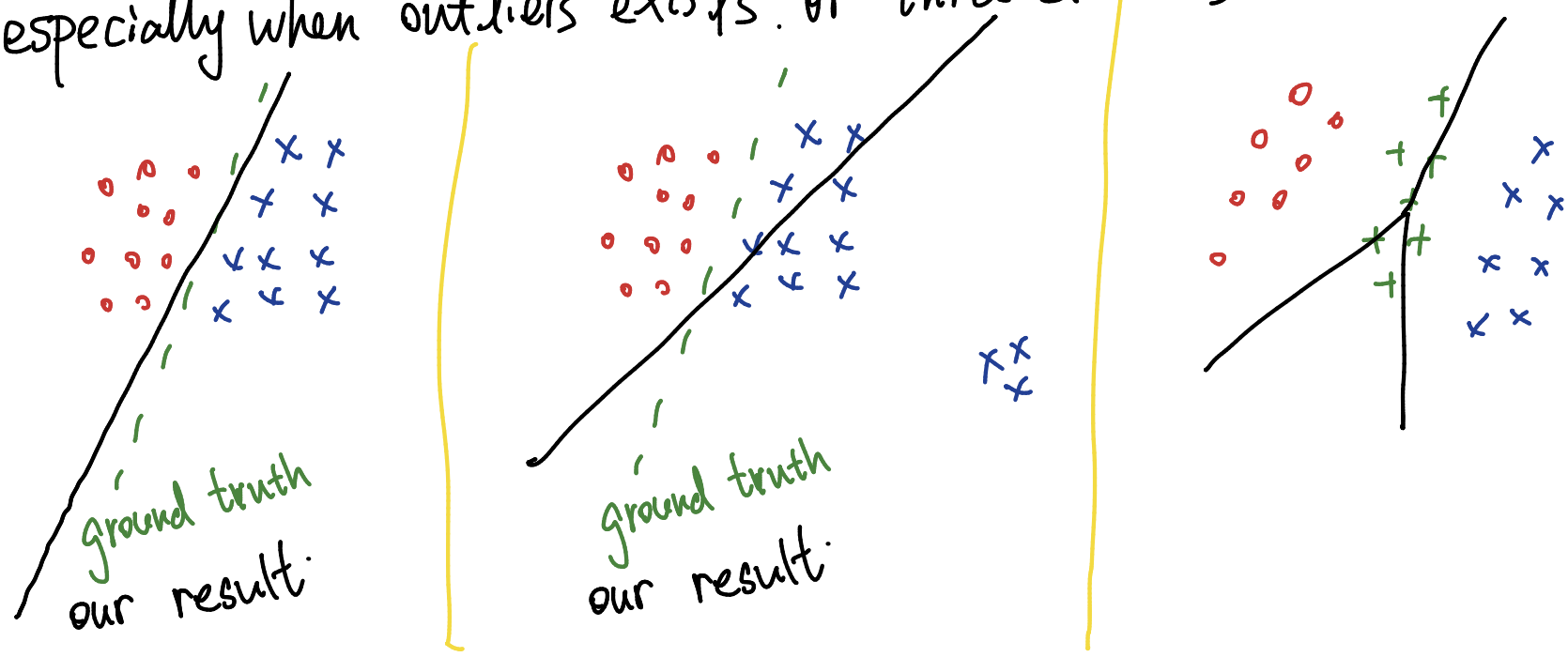
$\Rightarrow a^T \underline{y}(\underline{x}) + b = 0$

So, if we use 1-bit hotkey, $a = \mathbb{1}$, then

$\mathbb{1}^T \cdot \underline{t}_n = 1$.

So, $\underline{y}(\underline{x})$'s sumation of all elements is 1. But it may not be interpreted as probability, because the element may not guaranteed to be in $[0, 1]$

We should be clear that least-square method lacks robustness, especially when outliers exists. or three classes show in 2-dimension



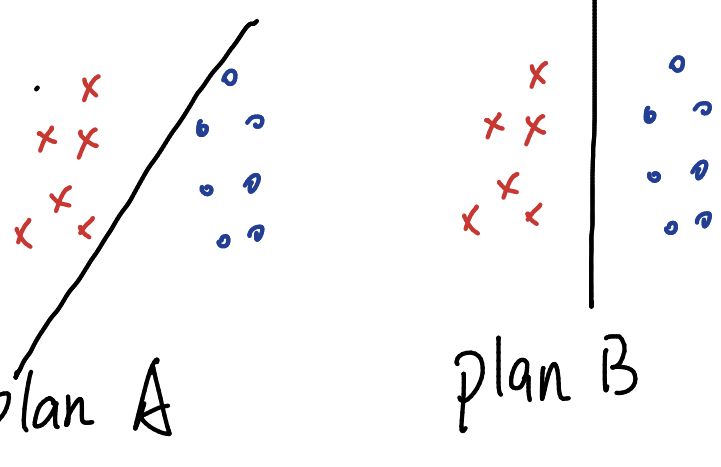
4. Fisher Linear Discriminant

In previous section, our method is project $\underline{x}$ onto one dimension by $y = \underline{w}^T \underline{x}$

and find out whether y is >0 or <0.

But we didn't make use of points fully

E.g.



Obviously, plan B is better. because the gap between points & boundary is larger.

Our target in this section is to find a good direction of boundary by adjust w to maximize the gap.

Suppose we have N_1 points for C_1, N_2 points for C_2. The mean vector for two classes are

$$\underline{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \underline{x}_n \qquad \underline{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \underline{x}_n$$

One simple strategy is to maximize the projection of $\underline{m}_1, \underline{m}_2$

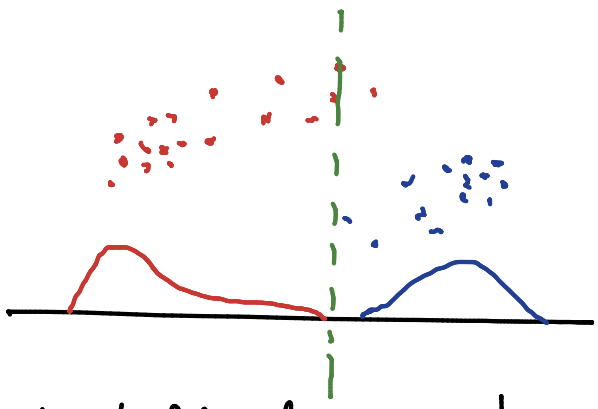
$$\arg\max_{\underline{w}} \underline{w}^T (\underline{m}_2 - \underline{m}_1)$$

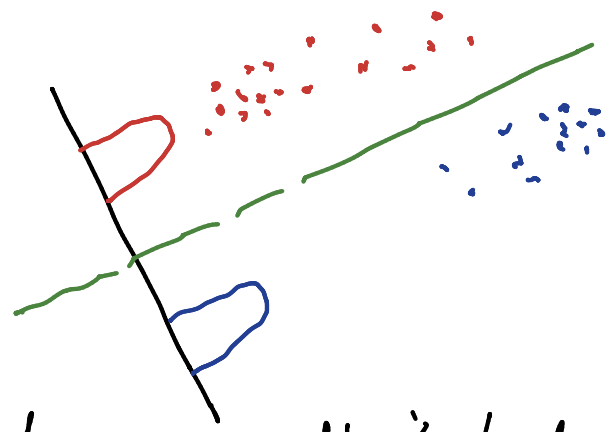
As you may feel, we need to constrain the magnitude of $\underline{W}$.

The overall expression is

$$\underset{\underline{W},\|\underline{W}\|=1}{\arg\max}\ \underline{W}^T(\underline{m}_1-\underline{m}_2)$$

Besides from separation of class's center, we want projected points have low variance.





The left decision has large "center separation" but high inner-class variance and not that good performance. The right decision is obviously better. So, let's give out a new objective function

$$J(\underline{W}) = \frac{\left[\underline{W}^T(\underline{m}_1-\underline{m}_2)\right]^2}{\underbrace{\sum_{n\in C_1}\left[\underline{W}^T(\underline{m}_1-\underline{x}_n)\right]^2}_{\text{projected points' var}} + \sum_{n\in C_2}\left[\underline{W}^T(\underline{m}_2-\underline{x}_n)\right]^2}$$

$$= \frac{\underline{W}^T S_B \underline{W}}{\underline{W}^T S_W \underline{W}}$$

$$S_B = (\underline{m}_2-\underline{m}_1)(\underline{m}_2-\underline{m}_1)^T$$

$$S_W = \sum_{n\in C_1}(\underline{m}_1-\underline{x}_n)(\underline{m}_1-\underline{x}_n)^T + \sum_{n\in C_2}(\underline{m}_2-\underline{x}_n)(\underline{m}_2-\underline{x}_n)^T$$

Our next target is to find out $\underline{W}^*$

$$\frac{\partial}{\partial \underline{W}} J(\underline{W}) = \frac{2\underline{W}^T S_W \underline{W} S_B \underline{W} - 2\underline{W}^T S_B \underline{W} S_W \underline{W}}{\underline{W}^T S_W \underline{W} \underline{W}^T S_W \underline{W}} = 0$$

$$\Rightarrow \underbrace{\underline{W}^T S_B \underline{W}}_{\text{Scalar}} S_W \underline{W} = \underbrace{\underline{W}^T S_W \underline{W}}_{\text{Scalar}} S_B \underline{W}$$

We don't care scalar, or the magnitude of $\underline{W}$, let's use

$$\begin{cases} \underline{W}^T S_B \underline{W} = a \\ \underline{W}^T S_W \underline{W} = b \end{cases} \qquad a S_W \underline{W} = b S_B \underline{W}$$

$$S_B \underline{W} = (\underline{m}_2 - \underline{m}_1)\underbrace{(\underline{m}_2 - \underline{m}_1)^T \underline{W}}_{\text{scalar} \leftarrow \text{use } C}$$

Then we have

$$a S_W \underline{W} = b \cdot C (\underline{m}_2 - \underline{m}_1) \Rightarrow \underline{W} = \alpha \cdot S_W^{-1} (\underline{m}_2 - \underline{m}_1)$$

Recall that $\|\underline{W}\| = 1$, the final answer is

$$\underline{W} = \frac{S_W^{-1}(\underline{m}_2 - \underline{m}_1)}{\| S_W^{-1}(\underline{m}_2 - \underline{m}_1) \|}$$

→ Fisher's Linear Discriminant

5. The perceptron algorithm

This is also a linear discriminant model but can work online. This algorithm can be viewed as the first generation of Gradient Descent learning.

The model is

$$y(x) = f(w^T \phi(x))$$

$\phi(x)$ is a fixed kernel function.

w is the parameter that we want to learn

$f(\cdot)$ is the active function given by a step function

$$f(x) = \begin{cases} +1 & x \geq 0 \quad \cdots \text{ Class } C_1 \\ -1 & x < 0 \quad --- \text{ Class } C_2 \end{cases}$$

We need to sequentially update w to make the classifier more and more accurate.

The update is made based on minimizing error

Error function, also known as perceptron criterion is

$$E_p(w) = -\sum_{n \in M} w^T \phi_n(x) t_n$$

M ← Miss labled data.

Note that. if the algorithm miss-lables sample i, then

$$w^T \phi(x_i) t_i < 0$$

So the definition above is reasonable

Then the update turns to be

$$w^+ = w - \eta \nabla_w E_p(w) = w + \eta \phi(x) t$$