Review of Linear Discrimination

Mengting Chen

October 27, 2022

1 Three Approaches to Decision Problems

(a) First solve the inference problem of determining the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ for each class $\mathcal{C}_k$ individually. Also separately infer the prior class probabilities $p(\mathcal{C}_k)$. Then use Bayes' theorem in the form

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} \tag{1.82}$$

to find the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$. As usual, the denominator in Bayes' theorem can be found in terms of the quantities appearing in the numerator, because

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k). \tag{1.83}$$

Equivalently, we can model the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ directly and then normalize to obtain the posterior probabilities. Having found the posterior probabilities, we use decision theory to determine class membership for each new input $\mathbf{x}$. Approaches that explicitly or implicitly model the distribution of inputs as well as outputs are known as *generative models*, because by sampling from them it is possible to generate synthetic data points in the input space.

(b) First solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$, and then subsequently use decision theory to assign each new $\mathbf{x}$ to one of the classes. Approaches that model the posterior probabilities directly are called *discriminative models*.

(c) Find a function $f(\mathbf{x})$, called a discriminant function, which maps each input $\mathbf{x}$ directly onto a class label. For instance, in the case of two-class problems, $f(\cdot)$ might be binary valued and such that $f = 0$ represents class $\mathcal{C}_1$ and $f = 1$ represents class $\mathcal{C}_2$. In this case, probabilities play no role.

2 Approach (c)

2.1 Two classes

Two classes: $\{C_1, C_2\}$.

Linear discriminant function: $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x} + w_0$.

Decision rules: $\begin{cases} y(\mathbf{x}) \geq 0 & \mathbf{x} \in C_1 \\ y(\mathbf{x}) < 0 & \mathbf{x} \in C_2 \end{cases}$.

Decision boundary: $y(\mathbf{x}) = 0$.

2.2 Multiple classes

Multiple classes: $\{C_1, \ldots, C_K\}$.

Linear discriminant function: $y_k(\mathbf{x}) = \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k,0}$.

Decision rules: $\begin{cases} y_k(\mathbf{x}) \geq y_j(\mathbf{x}) & \mathbf{x} \in C_k \\ y_k(\mathbf{x}) < y_j(\mathbf{x}) & \mathbf{x} \in C_j \end{cases}$.

Decision boundary: $y_k(\mathbf{x}) = y_j(\mathbf{x})$ or $y_k(\mathbf{x}) - y_j(\mathbf{x}) = (\mathbf{w}_k - \mathbf{w}_j)^\mathsf{T}\mathbf{x} + (w_{k,0} - w_{j,0}) = 0$.

3 Probabilistic Generative Models

By **Bayes' theorem**: $P(C_k|\mathbf{x}) \propto P(\mathbf{x}|C_k)P(C_k)$.

Decision rules: Choose the class with largest posterior.

3.1 Two classes

The posterior probability for class C_1 can be written as

$$\begin{aligned} P(C_1|\mathbf{x}) &= \frac{P(C_1|\mathbf{x})}{P(C_1|\mathbf{x}) + P(C_2|\mathbf{x})} \\ &= \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x}|C_1)P(C_1) + P(\mathbf{x}|C_2)P(C_2)} \\ &= \frac{1}{1 + \frac{P(\mathbf{x}|C_2)P(C_2)}{P(\mathbf{x}|C_1)P(C_1)}} \\ \sigma(a) &:= \frac{1}{1 + e^{-a}}, \end{aligned} \tag{1}$$

where

$$a(P(C_1|\mathbf{x})) := \ln \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x}|C_2)P(C_2)}. \tag{2}$$

We call $\sigma(\cdot)$ as the **sigmoid** function, and $a(\cdot)$ as the **logit** function.

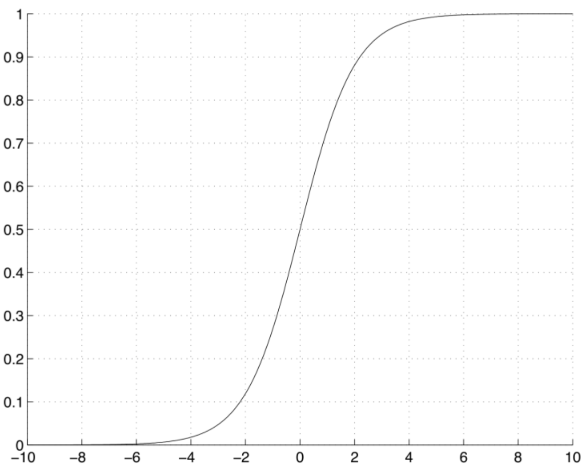


Figure 1: Sigmoid

Decision rules via posterior: Choose C_1, if $P(C_1|\mathbf{x}) \geq 0.5$.

In order to classify an input $\mathbf{x}$, we need to compute posterior for each class, which is uniquely identified by the logit function a. So a is defined as **discriminant** here.

Decision rules via discriminant: Choose C_1, if $a(P(C_1|\mathbf{x})) \geq 0$. ($\Leftrightarrow P(C_1|\mathbf{x}) \geq 0.5$)

3.1.1 Gaussian $p(\mathbf{x}|C_k)$ with the same $\boldsymbol{\Sigma}$

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}, k = 1, 2. \tag{3}$$

Plug (3) into (1), we have

$$\begin{aligned}
a(P(C_1|\mathbf{x})) &= \ln \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x}|C_2)P(C_2)} \\
&= \ln \frac{P(\mathbf{x}|C_1)}{P(\mathbf{x}|C_2)} + \ln \frac{P(C_1)}{P(C_2)} \\
&= \ln \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)\right\}}{\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)\right\}} + \ln \frac{P(C_1)}{P(C_2)} \\
&= \boldsymbol{\Sigma}^{-1}\left[(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\mathsf{T}\mathbf{x} + (-\frac{1}{2}\left\|\boldsymbol{\mu}_1\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\mu}_2\right\|^2)\right] + \ln \frac{P(C_1)}{P(C_2)} \\
&= \mathbf{w}^\mathsf{T}\mathbf{x} + w_0.
\end{aligned}$$

So if class-conditional densities are Gaussian with the same covariance matrix, the discriminant is **linear** over $\mathbf{x}$.

Decision rules via discriminant: Choose C_1, if $a(P(C_1|\mathbf{x})) = \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 \geq 0$.

3.2 Multiple classes

The posterior probability for class C_k can be written as

$$\begin{aligned} P(C_k|\mathbf{x}) &= \frac{P(C_k|\mathbf{x})}{\sum_{j=1}^{K} P(C_j|\mathbf{x})} \\ &= \frac{P(\mathbf{x}|C_k)P(C_k)}{\sum_{j=1}^{K} P(\mathbf{x}|C_j)P(C_j)} \\ \mathrm{softmax}(a_k) &:= \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \end{aligned} \tag{4}$$

where

$$a_k(\mathbf{x}) := \ln P(\mathbf{x}|C_k)P(C_k).$$

As posterior is identified by all a_k's, they are defined as discriminants.

Decision rules via posterior: Choose C_k, if $P(C_k|\mathbf{x})$ is the largest.

Decision rules via discriminant: Choose C_k, if $a_k(\mathbf{x})$ is the largest.

Similarly, if class-conditional densities are Gaussian with the same covariance matrix, the discriminant is linear over $\mathbf{x}$.

Decision rules via discriminant: Choose C_k, if $a_k(\mathbf{x}) = \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k,0} \geq a_j(\mathbf{x}) = \mathbf{w}_j^\mathsf{T}\mathbf{x} + w_{j,0}$ for all $j \neq k$.

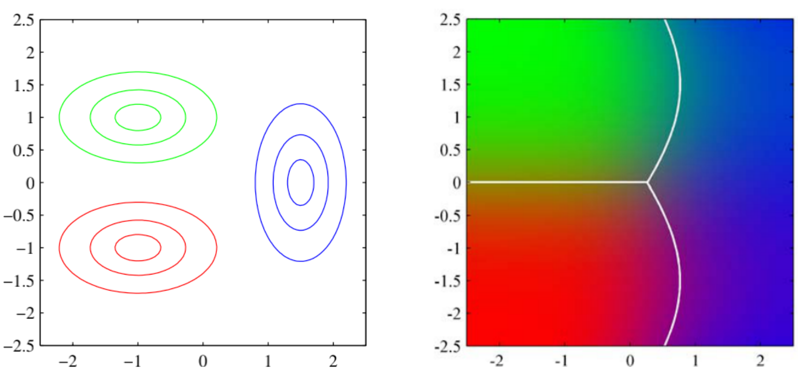


Figure 2: Three Gaussian models, the first two share the same covariance matrix

4 Probabilistic Discriminative Models

Directly **maximize a likelihood function** defined through the conditional distribution $p(C_k|\mathbf{x})$. Or equivalently, **minimize negative log-likelihood** (cross-entropy error).

4.1 Two classes

As in (1), posterior of class C_1 is in the form of sigmoid.
1) Build error function
2) Use gradient descent method to estimate parameters (the chain rule)
3) Compute $P(C_1|x) = \sigma(a)$, choose C_1 if $P(C_1|x) > 0.5$.

4.2 Multiple classes

As in (4), posterior of class C_k is in the form of softmax.
1) Build error function
2) Use gradient descent method to estimate parameters
3) Compute $P(C_k|x) = \text{sigmoid}(a_k)$, choose C_k if $P(C_k|x)$ is the largest.

Two-class Classification

Perceptron

Input vector $\mathbf{x} = [x_1, \ldots, x_n]^T$, want to find

$$f(\mathbf{x}) = \text{sign}\left(\left(\sum_{j=1}^{n} w_j x_j\right) + w_0\right)$$

The "bias weight" w_0 corresponds to the threshold when the neuron is triggered.
We have defined a Hypothesis set $\mathcal{H}$ (dummy variable $x_0 \equiv 1$)

$$\mathcal{H} = \{f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})\}$$

called the perceptron or linear separator

A perceptron fits the data by using a line to separate the $+1$ from -1 data

A simple learning model

- Input vector $\mathbf{x} = [x_1, \ldots, x_d]^T$
- Given importance weights to the different inputs and compute a "Credit Score"
  "Credit Score" $= \sum_{i=1}^{d} w_i x_i$.
- Approve credit if the "Credit Score" is acceptable
  "Approve Score" $= \sum_{i=1}^{d} w_i x_i >$ threshold. ("credit" is good)
  "Deny Score" $= \sum_{i=1}^{d} w_i x_i <$ threshold. ("credit" is bad)
- How to choose the importance weights w_i

input x_i is important	$\rightarrow$	large weight $\lvert w_i \rvert$
input x_i beneficial for credit	$\rightarrow$	positive weight $w_i > 0$
input x_i detrimental for credit	$\rightarrow$	negative weight $w_i < 0$

$$\text{“Approve Score”} = \sum_{i=1}^{d} w_i x_i > \text{threshold. (“credit” is good)}$$
$$\text{“Deny Score”} = \sum_{i=1}^{d} w_i x_i < \text{threshold. (“credit” is bad)}$$

can be written formally as

$$f(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + w_0\right)$$

The “bias weight” w_0 corresponds to the threshold when the neuron is triggered

$\mathbf{x} = [x_1, \ldots, x_d], \mathbf{w}' = [w_1, \ldots, w_d]$

(1) $\mathbf{w}'^T\mathbf{x} > \text{threshold}, \ \text{Y};$

(2) $\mathbf{w}'^T\mathbf{x} \leq \text{threshold}, \ \text{N};$

(1) can be rewritten as $\mathbf{w}'^T\mathbf{x} - \text{threshold} = \mathbf{w}'^T\mathbf{x} + w_0 = \mathbf{w}^T\mathbf{x} > 0,$ where

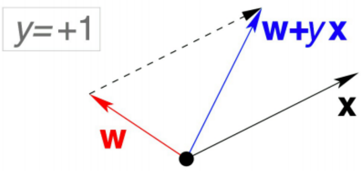
$w_0 = -\text{threshold}, \quad \mathbf{w} = [w_1, \ldots, w_d, w_0].$

The perceptron learning algorithm (PLA)

NOT REQUIRED!

The perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^\intercal \mathbf{x})$$

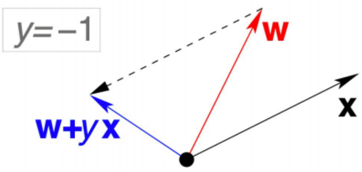


Given the training set:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

pick a misclassified point:

$$\text{sign}(\mathbf{w}^\intercal \mathbf{x}_n) \neq y_n$$



and update the weight vector:

$$\boxed{\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n}$$

Why adding $\pm\mathbf{x}_i$ to $\mathbf{w}$?

PLA implements our idea: start at some weights and try to improve it

“Incremental learning” on a single example at a time

$\mathbf{a} \cdot \mathbf{b} = \cos\langle \mathbf{a}, \mathbf{b} \rangle \, ||\mathbf{a}|| \, ||\mathbf{b}||$

Logistic Regression

Finding loss functions

A third linear prediction model

$$s = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$

linear classification

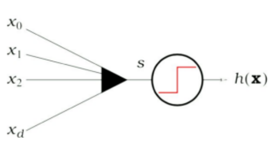
$h(\mathbf{x}) = \text{sign}(s)$

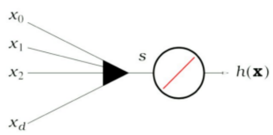
linear regression

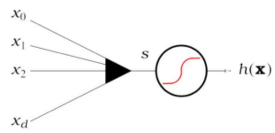
$h(\mathbf{x}) = s$

logistic regression

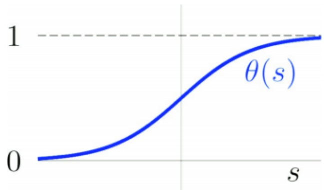
$h(\mathbf{x}) = \theta(s)$







The logistic function



$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

Properties about θ:

$$\theta(-s) = 1 - \theta(s), \quad \theta'(s) = \frac{e^s}{(1+e^s)^2} = \theta(s)(1-\theta(s))$$

Error Measure: likelihood

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

$$P(y \mid \mathbf{x}) = \theta(y\ \mathbf{w}^{\mathsf{T}}\mathbf{x})$$

Properties about θ:

For an input $\mathbf{x}$, it has two possibilities: being labelled as $+1$, or -1.

Compute $score = \mathbf{w}^T\mathbf{x}$,

1. if $score > 0$,
   a. then it is more likely to be classified into $y = +1$,
      $P(y = +1|\mathbf{x}) = h(\mathbf{x}) = \theta(score) \in (0.5, 1)$.
   b. and is less likely to be classified into $y = -1$,
      $P(y = -1|\mathbf{x}) = 1 - h(\mathbf{x}) = \theta(-score) \in (0, 0.5)$.
2. if $score < 0$,

a. then it is more likely to be classified into $y=-1$,

$P(y=-1|\mathbf{x}) = h(\mathbf{x}) = \theta(score) \in (0, 0.5)$.

b. and is less likely to be classified into $y=+1$,

$P(y=+1|\mathbf{x}) = 1 - h(\mathbf{x}) = \theta(-score) \in (0.5, 1)$.

So $P(y|\mathbf{x}) = \theta(y \cdot score)$.

Likelihood of $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ is

$$\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) = \prod_{n=1}^{N} \theta(y_n \mathbf{w}^\intercal \mathbf{x}_n)$$

MLE

Maximize the likelihood, is to minimize:

$$-\frac{1}{N} \ln \left(\prod_{n=1}^{N} \theta(y_n \mathbf{w}^\intercal \mathbf{x}_n) \right)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \ln \left(\frac{1}{\theta(y_n \mathbf{w}^\intercal \mathbf{x}_n)} \right) \qquad \left[\theta(s) = \frac{1}{1 + e^{-s}} \right]$$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \underbrace{\ln \left(1 + e^{-y_n \mathbf{w}^\intercal \mathbf{x}_n} \right)}_{\mathrm{e}\left(h(\mathbf{x}_n), y_n\right)} \qquad \text{“cross-entropy” error}$$

Summary:

- $score = \mathbf{w}^T \mathbf{x}_i$, where $\mathbf{w} = [\mathbf{w}', w_0]$
- Perceptron: $\text{sign}(score)$ for classification: $y_{pred} = \{+1, -1\}$
  - treat data <u>locally</u>, require the dataset to be linearly separable
  - if $\mathbf{x}_i$ is misclassified, then update weights: $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
- $\text{sigmoid}(score)$ for probability: $P(y|\mathbf{x}) \in (0, 1)$
  - treat data <u>globally</u>, be tolerable to noise
  - Loss: negative log-likelihood