08-Linear Classification (Discriminative)

Probabilistic assumption

- We would like to impose probabilistic assumptions to classification tasks, so that we could have a principled way of learning, e.g., MLE and MAP estimation.
- Bernoulli distribution

  A Bernoulli distribution, parametrized by p, is a two-point distribution

  $$(1-p) < 0 > +p < 1 >$$

  That is,

  $$\Pr\{t=1\} = p \quad \text{and} \quad \Pr\{t=0\} = 1-p$$

  The probability of t is $p^t \cdot (1-p)^{1-t}$

  - If $t = 1$, the probability is p.
  - If $t = 0$, the probability is $1 - p$.

- For a classification task, we can always assume $t|\boldsymbol{x} \sim \text{Bernoulli}(p)$
- And, we further assume

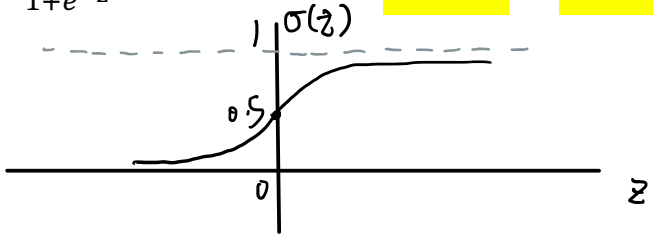
  $$t|\boldsymbol{x} \sim \text{Bernoulli}(f_{\boldsymbol{\theta}}(\boldsymbol{x}))$$

  for some unknown parameters $\boldsymbol{\theta}$.
- In linear regression, $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b \in \mathbb{R}$, which does not work for $p \in [0,1]$.
- For classification, we may squash the linear function into $(0,1)$ as

  $$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

  where

  $\sigma(z) = \frac{1}{1+e^{-z}}$ is known as the sigmoid or logistic function.

  The resulting model is known as logistic regression

- Why sigmoid function?
  - Exponential family
  - Other squashing functions are also acceptable

- E.g., the probit function $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(t; 0, 1)\, dt$

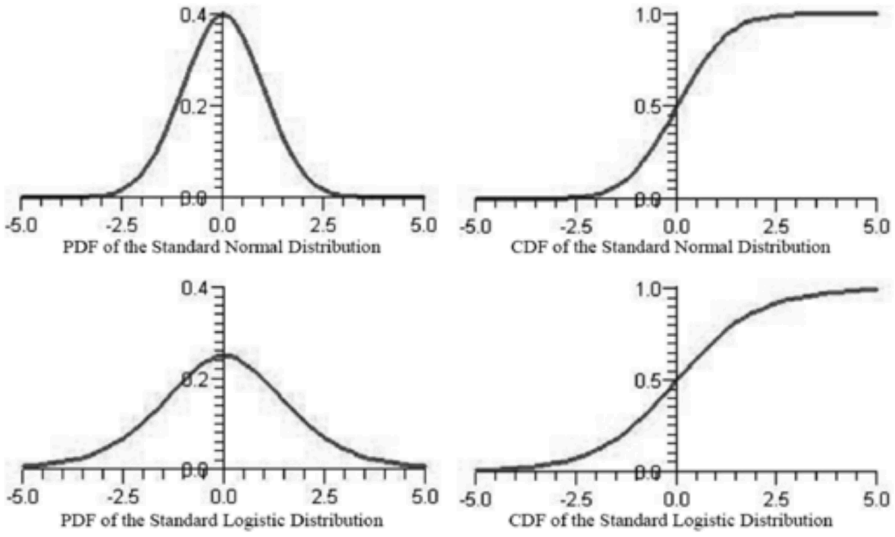


Figure 1. The Standard Normal and Standard Logistic Probability Distributions

Source : Park (2010)

Note: The different between a probit function or a sigmoid function is not merely scaling.

Maximum Likelihood Estimation (MLE) for Logistic Regression

Suppose we have a dataset $\{(\boldsymbol{x}^{(m)}, t^{(m)})\}_{m=1}^{M}$, and the above assumption says

$$t^{(m)} \sim \text{Bernoulli}\left(f_{\boldsymbol{w},b}(\boldsymbol{x})\right)$$

where $\boldsymbol{w}, b$ are true, but unknown, parameters.

The log-likelihood is

$$\log p(t^{(1)}, \cdots t^{(M)} \mid \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(M)})$$

$$= \log \prod_{m=1}^{M} p(t^{(m)} \mid \boldsymbol{x}^{(m)}) \qquad [\text{data iid, and } t^{(m)} \text{ only depends on } \boldsymbol{x}^{(m)}]$$

$$= \sum_{m=1}^{M} \log p(t^{(m)} \mid \boldsymbol{x}^{(m)}) \qquad [\log \Pi \Leftrightarrow \Sigma \log]$$

$$= \sum_{m=1}^{M} \log \left[f_{\boldsymbol{w},b}(\boldsymbol{x})\right]^{t^{(m)}} \cdot \left[1 - f_{\boldsymbol{w},b}(\boldsymbol{x})\right]^{1-t^{(m)}} \qquad [\text{Bernoulli}]$$

$$= \sum_{m=1}^{M} \left[t^{(m)} \log f_{\boldsymbol{w},b}(\boldsymbol{x}^{(m)}) + (1 - t^{(m)}) \cdot \log\left(1 - f_{\boldsymbol{w},b}(\boldsymbol{x}^{(m)})\right)\right]$$

The training objective is to minimize

$$J(\mathbf{w},b) = \frac{1}{M}\sum_{m=1}^{M}\left[-t^{(m)}\log f_{\mathbf{w},b}(\boldsymbol{x}) - (1-t^{(m)})\log(1-f_{\mathbf{w},b}(\boldsymbol{x}))\right]$$

$t \in \{0,1\}$ can be thought of as a distribution $\boldsymbol{t}$.

If $t = 0$, then $\boldsymbol{t} = \begin{pmatrix}1\\0\end{pmatrix}$ If $t = 1$, then $\boldsymbol{t} = \begin{pmatrix}0\\1\end{pmatrix}$.

t is the index representation, whereas $\boldsymbol{t}$ is the one-hot representation
They are equivalent.

It is noted that $\begin{pmatrix}1-f_{\mathbf{w},b}(\boldsymbol{x})\\ f_{\mathbf{w},b}(\boldsymbol{x})\end{pmatrix}$ is also a distribution

The above objective is the cross-entropy loss of $f_{w,b}(\boldsymbol{x})$ against $\boldsymbol{t}$.

The Kullback--Leibler (KL) divergence is an important measure of the dissimilarity between two distributions. The KL of two distributions $\boldsymbol{p}$ and $\boldsymbol{q}$, in the discrete case, is defined as

$$KL(\boldsymbol{p}\|\boldsymbol{q}) = \sum_{i=1}^{K} p_i \log\frac{p_i}{q_i}$$

[HW] Show that minimizing the cross-entropy loss is equivalent to minimizing the KL divergence between the predicted distribution and the one-hot target distribution. Either $KL(\boldsymbol{f}\|\boldsymbol{t})$ or $KL(\boldsymbol{t}\|\boldsymbol{f})$, but which and why?

Gradient for cross-entropy loss

We introduce an intermediate variable

We don't need the hat for y because we denote groundtruth label as t.

$$z^{(m)} = \mathbf{w}^T\boldsymbol{x}^{(m)} + b$$

$$y^{(m)} = f(z^{(m)}) = \frac{1}{1+e^{-z^{(m)}}}$$

$$J^{(m)}(\mathbf{w},b) = \left[-t^{(m)}\log y^{(m)} - (1-t^{(m)})\log(1-y^{(m)})\right]$$

$$\frac{\partial}{\partial z^{(m)}}J^{(m)}(\mathbf{w},b) = \frac{\partial}{\partial z^{(m)}}\left[t^{(m)}\log(1+e^{-z^{(m)}}) - (1-t^{(m)})\log\left(\frac{e^{-z^{(m)}}}{1+e^{-z^{(m)}}}\right)\right]$$

$$= t^{(m)}\frac{1}{1+e^{-z^{(m)}}}\cdot e^{-z^{(m)}}\cdot(-1) - (1-t^{(m)})(-1) + (1-t^{(m)})\cdot\frac{1}{1+e^{-z^{(m)}}}\cdot e^{-z^{(m)}}\cdot(-1)$$

$$= y^{(m)} - t^{(m)} \qquad \left(\frac{1}{1+e^{-z^{(m)}}} = y^{(m)},\ \frac{e^{-z^{(m)}}}{1+e^{-z^{(m)}}} = 1-y^{(m)}\right)$$

$$\frac{\partial J^{(m)}}{\partial w_i} = \frac{\partial J^{(m)}}{\partial z^{(m)}}\cdot\frac{\partial z^{(m)}}{\partial w_i} = \left(y^{(m)} - t^{(m)}\right)\cdot x_i$$

$$\frac{\partial J}{\partial b_i} = y^{(m)} - t^{(m)}$$

Since $J = \sum_{m=1}^{M} J^{(m)}$,

$$\frac{\partial J}{\partial w_i} = \sum_{m=1}^{M} \left(y^{(m)} - t^{(m)}\right) \cdot x_i^{(m)} \qquad \frac{\partial J}{\partial \boldsymbol{w}} = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{t})$$

$$\frac{\partial J}{\partial b} = \sum_{m=1}^{M} \left(y^{(m)} - t^{(m)}\right)$$

Multi-class classification

If $t \in \{1, 2, \ldots, K\}$, it is safe to assume $t|\boldsymbol{x} \sim$Multinomial($\boldsymbol{p}$), where $\boldsymbol{p}$ is a probability of a multinomial distribution with $p_k = \Pr\{t = k|\boldsymbol{x}\}$

Note:

- A Bernoulli trial $x \sim \text{Bernoulli}(p)$ is a two-point distribution
  $(1-p)\langle 0\rangle + p\langle 1\rangle$
- A binomial distribution $n \sim$ Binomial(p, N) is the sum of Bernoulli(p) repeated N times, i.e.,
  $n = \Sigma_{i=1}^{N} x_i$, where $x_i \sim \text{Bernoulli}(p)$
- In a general setting, a multinomial distribution is parametrized by Multinomial$(\boldsymbol{p}, N)$, and
  $n = \Sigma_{i=1}^{N} x_i$, where $\Pr\{x_i = k) = p_k$
  In classification problems, we usually have $N = 1$, which is omitted for simplicity. It is also called a multinoulli distribution, or a categorical distribution.

For classification problems, it suffices to model uncertainty by a multinomial distribution. However, we still need to design the machine learning model that computes $\boldsymbol{p}$ from $\boldsymbol{x}$.

Linear multi-class classification

We still hope to have a linear model like $W\boldsymbol{x} + \boldsymbol{b}$ and then normalize it as a probability. We make things work in a heuristic manner as follows.

- We define a scoring function $z_i = \boldsymbol{w}_i^T \boldsymbol{x} + b_i$
  i.e., $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{x} \in \mathbb{R}^d$ (number of input features)
  $\boldsymbol{z} \in \mathbb{R}^K$, $\boldsymbol{b} \in \mathbb{R}^K$ (number of categories)
  $\boldsymbol{W} \in \mathbb{R}^{K \times d}$
- We make the scoring function positive
  $\tilde{y}_i = \exp(z_i)$

- We make the scoring function normalized

$$y_i = \frac{\tilde{y}_i}{\sum_j \tilde{y}_j} = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Such computation is known as (k-way) softmax:

$$\boldsymbol{y} = \text{softmax}(\boldsymbol{z})$$

The terminology is said in comparison to (hard) argmax, which gives a one-hot index of the maximum element in the vector $\boldsymbol{z}$. Instead, softmax gives a soft distribution over all categories, and the maxinum entry in $\boldsymbol{z}$ is also the one in $\boldsymbol{y}$.

Maximum likelihood estimation for softmax multi-class classification

$$\begin{aligned}
&\log p(t^{(1)} \cdots t^{(M)} \mid \boldsymbol{x}^{(1)}, \cdots \boldsymbol{x}^{(M)}) \\
&= \log \prod_{m=1}^{M} p(t^{(m)} \mid \boldsymbol{x}^{(m)}) && [\text{Data iid, } t^{(m)} \text{ depends only on } \boldsymbol{x}^{(m)}] \\
&= \sum_{m=1}^{M} \log p(t^{(m)} \mid \boldsymbol{x}^{(m)}) && [\log \prod \Leftrightarrow \textstyle\sum \log] \\
&= \sum_{m=1}^{M} \log y^{(m)}_{t^{(m)}} && [\text{Index representation of } t^{(m)}] \\
&= \sum_{m=1}^{M} \sum_{k=1}^{K} t^{(m)}_k \log y^{(m)}_K && [\text{One-hot representation } \boldsymbol{t}^{(m)}]
\end{aligned}$$

Logistic regression vs. softmax

- Logistic regression is equivalent to a two-way softmax

For a two-way softmax with $t \in \{0, 1\}$.

$$y_0 = \hat{\Pr}[t=0 \mid \boldsymbol{x}] = \frac{\exp\{-(\boldsymbol{w}_0^T \boldsymbol{x} + b_0)\}}{\exp\{-(\boldsymbol{w}_0^T \boldsymbol{x} + b_0)\} + \exp\{-(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)\}}$$

$$y_1 = \hat{\Pr}[t=1 \mid \boldsymbol{x}] = \frac{\exp\{-(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)\}}{\exp\{-(\boldsymbol{w}_0^T \boldsymbol{x} + b_0)\} + \exp\{-(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)\}}$$

$$= \frac{1}{\exp\{-[\underbrace{(\boldsymbol{w}_0 - \boldsymbol{w}_1)}_{\boldsymbol{w}_0}{}^T \boldsymbol{x} + \underbrace{(b_0 - b_1)}_{b_0}]\} + 1}$$

$$= \sigma(\boldsymbol{w}_0^T \boldsymbol{x} + b_0)$$

- Most results for logistic regression still holds for softmax

[HW: derive the gradient of the cross-entropy loss for softmax]

Note that MLE for logistic regression is also equivalent to MLE for 2-

way softmax.

Also, MAP estimation/regularization can also be applied as in linear regression

Inference criteria (decision rule)

For a k-category classification problem, the hypothesis function is

$$h : \mathcal{X} \to \{1, \cdots, K\}$$

Suppose we have a probabilistic modeling of $\Pr\{t = k|\boldsymbol{x}\}$. Then,

Max a posteriori (MAP) inference <=> Maximizing accuracy

$$h(\boldsymbol{x}) = \operatorname{argmax} \Pr[t=k|\boldsymbol{x}] \qquad \text{maximize} \ \mathop{\mathbb{E}}_{t}[\mathbb{1}\{t = h(\boldsymbol{x})\}]$$

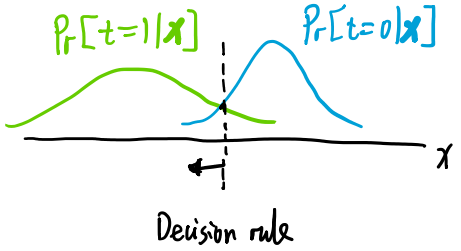
Proof:

$$\mathop{\mathbb{E}}_{t}[\mathbb{1}\{t = h(\boldsymbol{x})\}]$$

$$= \sum_{t=1}^{K} p(t|\boldsymbol{x})\,\mathbb{1}\{t = h(\boldsymbol{x})\}$$

$$= p(h(\boldsymbol{x})|\boldsymbol{x})$$

$$\underset{h(\boldsymbol{x})}{\text{maximize}} \ p(h(\boldsymbol{x})|\boldsymbol{x}) \iff h(\boldsymbol{x}) = \operatorname{argmax} p(\cdot|\boldsymbol{x})$$



This explains:

- For logistic regression, we predict $\hat{t} = \begin{cases} 1 & \text{if } y \geq 0.5 \\ 0 & \text{if } y < 0.5 \end{cases}$
- For softmax, we predict

$$\hat{t} = \operatorname{argmax} \mathbf{y}$$