Classification

Agenda

- Classification
- Logistic Regression

Classification

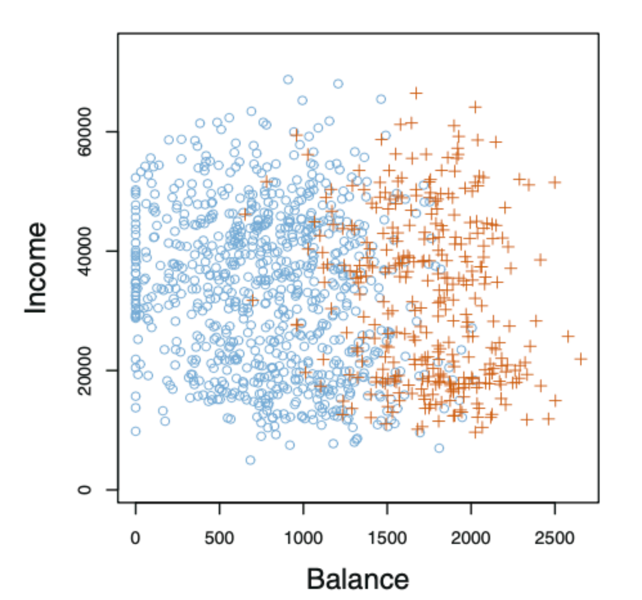
- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, based on the user's IP address, past transaction history.
- A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions {stroke, drug overdose, and epileptic seizure}
- Build a handwritten text recognition system

Qualitive response variable Y !

Y takes values in a predefined set $\mathcal{C}$

Motivation Example

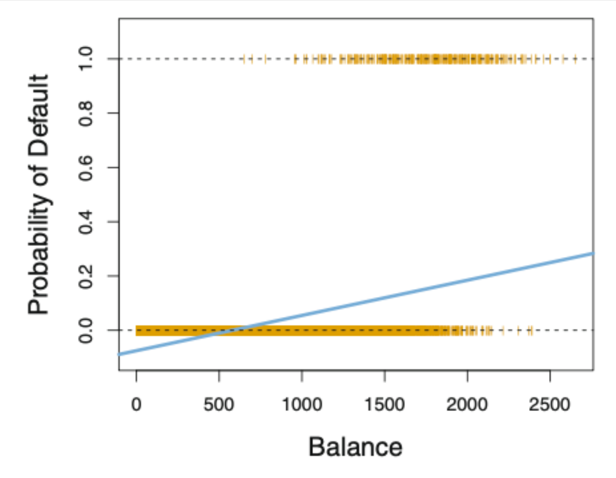
We are interested in predicting whether an individual will default on his or her credit card payment, on the basis of annual income and monthly credit card balance.



Can we use Linear Regression?

- Suppose for the Default classification task that we code

$$y = \begin{cases} 0 & if\ No \\ 1 & if\ Yes \end{cases}$$

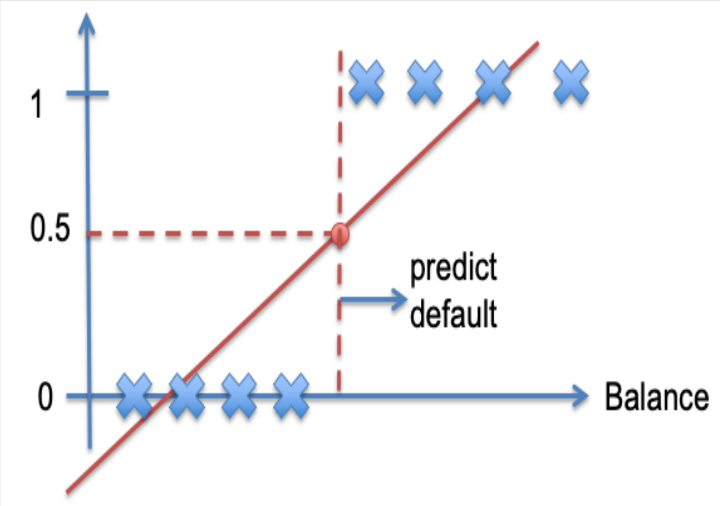


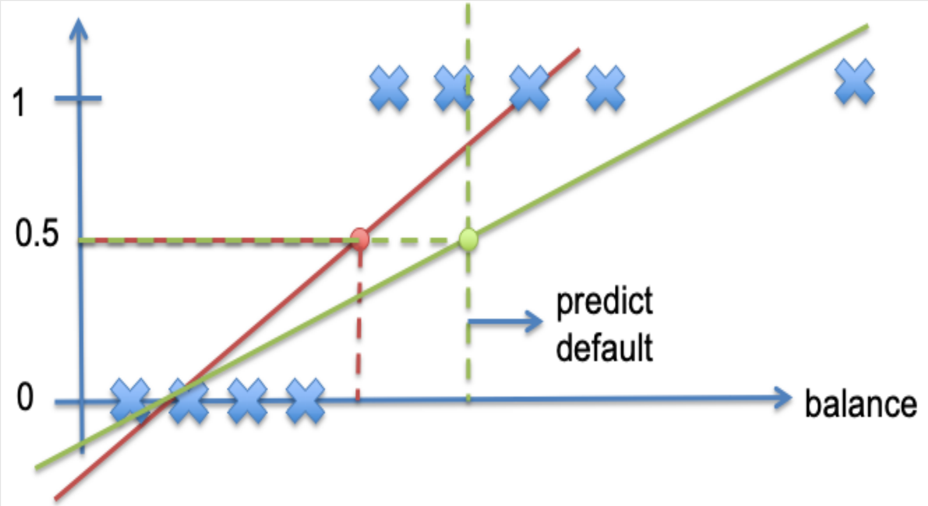
Can we simply perform a linear regression of Y on X and classify as Yes if $\hat{y}$> 0.5?

- Linear regression is estimating $P(Y|X)$
- Linear regression might produce probabilities less than zero or bigger than one. Logistic regression is more appropriate.

Can we use Linear Regression?

Even with Balanced data,
The cutoff value for LR is not 0.5





If $f(x) \geq 0.5$ predict y =1
If $f(x) < 0.5$ predict y =0

Linear regression with natural 0.5 threshold does not look good here.

Can we use Linear Regression?

Suppose for the emergence treatment example we code:

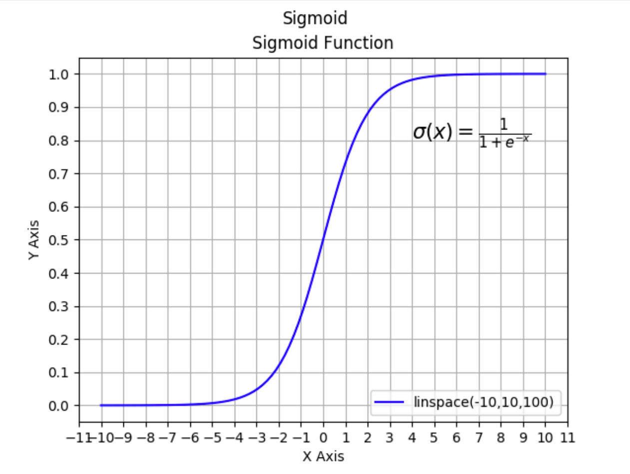
$$y = \begin{cases} 1 \ \ if\ stroke \\ 2 \ \ if\ drug\ overdose \\ 3 \ \ if\ eileptic\ seizure \end{cases}$$

This coding suggests an ordering, and in fact implies that the difference between stroke and drug overdose is the same as between drug overdose and epileptic seizure.

Logistic regression

Binary classification:

We want to predict $P(Y|X = x)$ where $y \in \{0, 1\}$. Let's write f$(x) = P(Y = 1|X)$



$$p(x) = w^T x \in (-\infty, +\infty) \quad p(x) \in [0, 1]$$

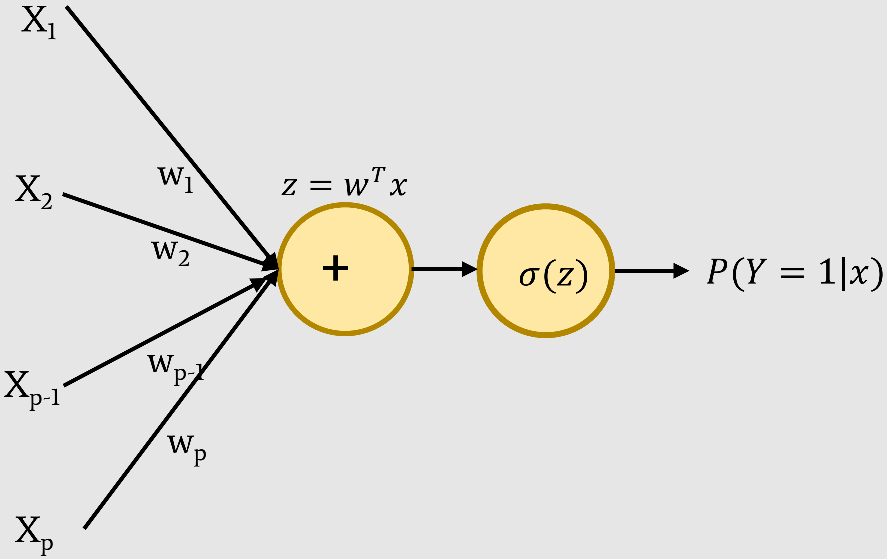
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$p(x) = \frac{1}{1 + \ e^{-w^T x}} = \frac{e^{w^T x}}{1 + \ e^{w^T x}} \quad p(x) = \sigma(w^T x)$$

Logistic Regression

Binary classification:

We want to predict $P(Y|X = x)$ where $y \in \{0, 1\}$. Let's write f(x) $= P(Y = 1|X)$



Logistic regression

Binary classification:

We want to predict $P(Y|X = x)$ where $y \in \{0, 1\}$. Let's write $\mathrm{f}(x) = P(Y = 1|X)$

$\Downarrow$

$P(Y=0|x) = 1 - P(Y=1|x)$

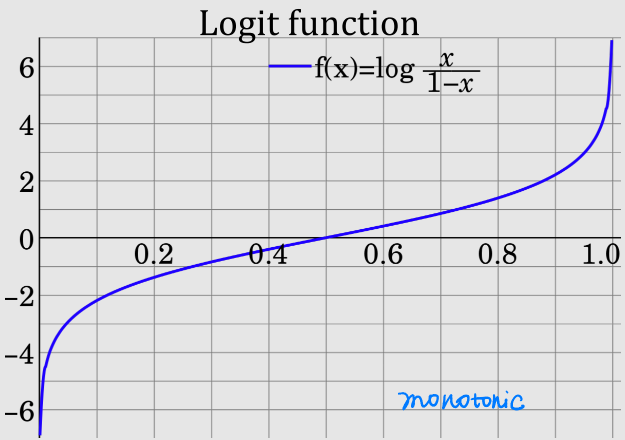
$$p(x) = \frac{e^{w^T x}}{1 + e^{w^T x}}$$

$$\frac{p(x)}{1 - p(x)} = e^{w^T x}$$

odds

$$log\ \frac{p(x)}{1 - p(x)} = w^T x$$

Log-odds/logit



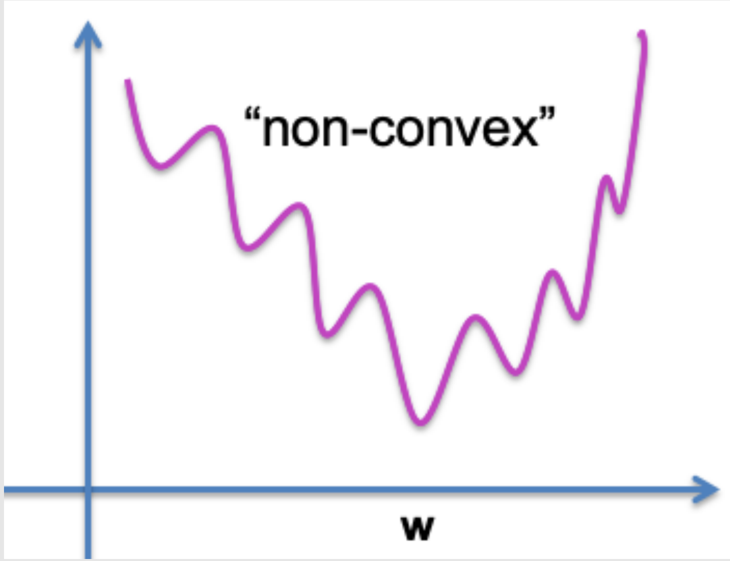
Logistic Regression – Parameter Learning

Learning the parameters: w.

Can we use MSE as loss function (as in Linear Regression)?

$$l(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - p(x_i))^2$$

$$p(x) = \frac{1}{1 + e^{-w^T x}}$$



Logistic Regression – Parameters Learning

Maximum Likelihood Estimation (MLE for short), is a probabilistic framework for estimating the parameters of a model.

In Maximum Likelihood Estimation, we wish to maximize the conditional probability of observing the data (X) given a specific probability distribution and its parameters.

Logistic Regression – parameters learning

In Logistic regression, we want to maximize the likelihood:

$$p_w(y_1, \dots y_N | x_1, \dots, x_N)$$

$$= \prod_{i=1}^{N} p_w(y_i | x_i)$$

It's equivalent to maximize the log-likelihood:

$$l(w) = \sum_{i=1}^{N} log p_w(y_i | x_i)$$

For binary classification, $y_i \in \{0, 1\}$, and we have

$$P(Y = 1|x) = p(x) = \frac{1}{1 + e^{-w^T x}}$$

$$P(Y = 0|x) = 1 - p(x)$$

Logistic Regression – parameters learning

It's equivalent to maximize the log-likelihood:

$$l(w) = \sum_{i=1}^{N} log p_w(y_i|x_i)$$

For binary classification, $y_i \in \{0, 1\}$, and we have

$$P(Y = 1|x) = p(x) = \frac{1}{1 + e^{-w^T x}}$$

$$P(Y = 0|x) = 1 - p(x)$$

Hence we have

$$P(Y = y|X = x) = p(x)^y [1 - p(x)]^{1-y}$$

Bernoulli (p(x))

Logistic Regression – parameters learning

For binary classification we can simplify log-likelihood as:

$$l(w) = \sum_{i=1}^{N} \{y_i \log p(x_i; w) + (1 - y_i) \log(1 - p(x_i; w))\}$$

Maximizing the log-likelihood is equivalent to minimize the cross entropy loss!

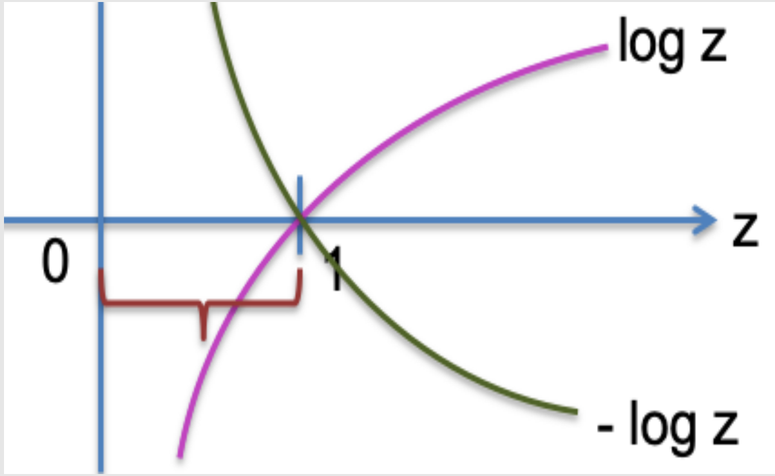
$$J(w) = -\sum_{i=1}^{N} \{y_i \log p(x_i; w) + (1 - y_i) \log(1 - p(x_i; w))\}$$

Logistic Regression – Intuition of loss function

The cross entropy loss:

$$\min Cost(x_i, y_i) = \begin{cases} -\log p(x_i; w) & if\ y_i = 1 \iff \max p(x_i, w) \\ -\log(1 - p(x_i; w)) & if\ y_i = 0 \iff \max(1 - p(x_i, w)) \end{cases}$$

- Consider y=1,
    - as $p(x_i; w)$→ 1, Cost → 0
    - as $p(x_i; w)$ →0, Cost →∞, we'll penalize learning algorithm by a very large cost.



Logistic Regression – Intuition of loss function

<u>Entropy:</u>

For a discrete event X: $H(X = x_0) = \overbrace{-log(p(x_0))}^{\text{How much suprise : Higher } p(x_0) \text{ less suprise}}$

Entropy for the random variable X: $\underset{\text{Expected suprise}}{H(X)} = -\sum_{i=1}^{n} p(x_i) log[p(x_i)]$

Note: The intuition behind quantifying information is the idea of measuring how much surprise there is in an event. Those events that are rare have more information than those events that are common.

<u>Cross Entropy:</u>

$$H(p,q) = -\sum_{i=1}^{n} \overbrace{p(x_i)}^{\text{ground truth prob}} log[\overbrace{q(x_i)}^{\text{predicted prob}}] \geq H(p)$$

Cross entropy is the expected surprisal of an observer with subjective probabilities Q upon seeing data that were actually generated according to probabilities P. Cross entropy $H(p,q)$ is minimized when $P = Q$

Logistic Regression – parameter learning

Cross Entropy Loss

$$J(w) = -\sum_{i=1}^{N}\{y_i \log p(x_i; w) + (1 - y_i)\log(1 - p(x_i; w))\}$$

Use gradient decent to minimize $J(w)$

Update w by $w_j := w_j - \lambda J(\boldsymbol{w})$

Logistic Regression – parameter learning

$$J(w) = -\sum_{i=1}^{N}\{y_i \boxed{\log p(x_i;w)} + (1-y_i)\log(1-p(x_i;w))\}$$

$$\boxed{\frac{\partial J(w)}{\partial w_l}}$$

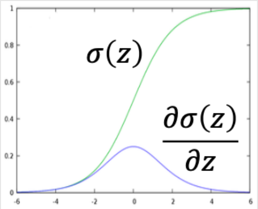
$$\frac{\partial \log p(x_i;w)}{\partial w_l}$$

$$= \frac{\partial \log \sigma(z)}{\partial z}\frac{\partial z}{\partial w_l}$$

$$= [1-\sigma(z)]\, x_l$$

$$p(x) = \frac{1}{1+e^{-z}} = \sigma(z), \quad z = w^T x = w_0 + \sum_{i=1}^{p} w_i \cdot x_i$$

$$\frac{\partial \log \sigma(z)}{\partial z} = \frac{1}{\sigma(z)}\frac{\partial \sigma(z)}{\partial z} = 1-\sigma(z), \quad \frac{\partial z}{\partial w_l} = x_l$$



$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

Logistic Regression – parameter learning

$$J(w) = -\sum_{i=1}^{N} \{y_i \log p(x_i; w) + (1 - y_i) \boxed{\log(1 - p(x_i; w))}\}$$

$$\boxed{\frac{\partial J(w)}{\partial w_l}}$$

$$\frac{\partial \log[1 - p(x_i; w)]}{\partial w_l}$$

$$= \frac{\partial \log[1 - \sigma(z)]}{\partial z} \frac{\partial z}{\partial w_l}$$

$$= \sigma(z) x_l$$

$$p(x) = \frac{1}{1 + e^{-z}} = \sigma(z), \quad z = w^T x = w_0 + \sum_{i=1}^{p} w_i x_i$$

$$\frac{\partial \log[1-\sigma(z)]}{\partial z} = -\frac{1}{1-\sigma(z)} \frac{\partial \sigma(z)}{\partial z} = \sigma(z), \quad \frac{\partial z}{\partial w_l} = x_l$$

Logistic Regression – parameter learning

$$\frac{\partial J(w)}{\partial w_l}$$

$$J(w) = -\sum_{i=1}^{N} \{y_i \log p(x_i; w) + (1 - y_i) \log(1 - p(x_i; w))\}$$

$$\frac{\partial J(w)}{\partial w_l} = -\sum_i \left[y_i \frac{\partial log p(x_i; w)}{\partial w_l} + (1 - y_i) \frac{\partial \log\big(1 - p(x_i; w)\big)}{\partial w_l} \right]$$

$(1 - \sigma(z))x_{il}$

$$= -\sum_i \big[y_i \underbrace{\big(1 - p(x_i; w)\big)x_{il}} - (1 - y_i)\underbrace{p(x_i; w)x_{il}}\big]$$

$$= -\sum_i [y_i - y_i p(x_i; w) - p(x_i; w) + y_i p(x_i; w)] x_{il}$$

$$= -\sum_i [y_i - p(x_i; w)] x_{il}$$

Larger difference
larger update

LR $\quad \frac{\partial J(w)}{\partial w_l} = -\sum_i [y_i - w^T x_i] x_{il}$

Logistic Regression – parameter learning

Cross Entropy Loss

$$J(w) = -\sum_{i=1}^{N} \{y_i \log p(x_i; w) + (1 - y_i) \log(1 - p(x_i; w))\}$$

Use gradient decent to minimize $J(w)$

Update w by $w_l := w_l - \lambda J(\boldsymbol{w})$

$$\frac{\partial J(w)}{\partial w_l} = -\sum_i [y_i - p(x_i; w)] x_{il}$$

$$w_l \leftarrow w_l - \lambda \sum_i -[y_i - p(x_i; w)] x_{il}$$

Logistic Regression – parameter learning (Newton)

The loss function is a convex function

$$J(w) = -\sum_{i=1}^{N}\{y_i \log p(x_i; w) + (1 - y_i)\log(1 - p(x_i; w))\}$$

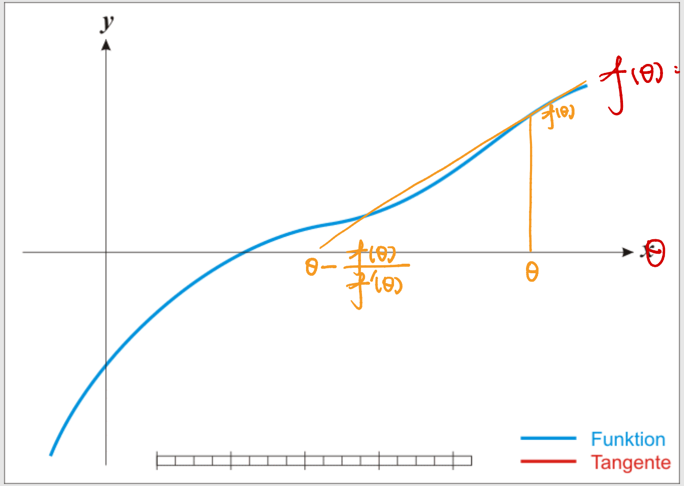
To minimize it, we set its derivative to be zero

$$\frac{\partial J(w)}{\partial w} = -\sum_i [y_i - p(x_i; w)]x_i = 0$$

Logistic Regression – parameter learning (Newton)

Given a function $f: R \to R$. We want to find a value of θ such that $f(\theta) = 0$.
Newton–Raphson algorithm perform the following update

$$\theta \leftarrow \theta - \frac{f(\theta)}{f'(\theta)}$$



$f(\theta)$: First order derivative

Logistic Regression – parameter learning (Newton)

To solve

$$\frac{\partial J(w)}{\partial w} = -\sum_i [y_i - p(x_i; w)]x_i = 0$$

We consider the Hessian matrix

$$\frac{\partial^2 J(w)}{\partial w \partial w^T} = -\sum_i x_i x_i^T p(x_i; w)[1 - p(x_i; w)]$$

Update date w using

$$w \leftarrow w - \left(\frac{\partial^2 J(w)}{\partial w \partial w^T}\right)^{-1} \frac{\partial J(w)}{\partial w}$$

Logistic Regression v.s. Linear Regression

Logistic Regression	Linear Regression
$f(x) = \sigma(w^T x) = \sigma\left(w_0 + \sum_{i=1}^{p} w_i x_i\right)$ Output between 0 and 1	$f(x) = w^T x = w_0 + \sum_{i=1}^{p} w_i x_i$ Output: any value
Label y_i: 1 for class 1, 0 for class 2 Cross entropy loss $-\sum_{i=1}^{N} \{y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i))\}$	Label y_i: a real number MSE loss $\frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2$
Gradient descent $w_l \leftarrow w_l - \lambda \sum_i -[y_i - p(x_i; w)] x_{il}$	Gradient descent $w_l \leftarrow w_l - \lambda \sum_i -[y_i - f(x_i; w)] x_{il}$

Logistic Regression – parameter regularization

Cross Entropy Loss : $J(w) = -\sum_{i=1}^{N}\{y_i \log p(x_i; w) + (1 - y_i)\log(1 - p(x_i; w))\}$

L2 regularization: $J(w) = -\sum_{i=1}^{N}\{y_i \log p(x_i; w) + (1 - y_i)\log\big(1 - p(x_i; w)\big)\} + \lambda \sum_{l=1}^{p} w_l^2$

L1 regularization: $J(w) = -\sum_{i=1}^{N}\{y_i \log p(x_i; w) + (1 - y_i)\log\big(1 - p(x_i; w)\big)\} + \lambda \sum_{l=1}^{p} |w_l|$

Logistic Regression – multiclass classification

Multi-class classification

$C1{:}\ z_1 = \beta_{10} + \beta_1^T x$

$C2{:}\ z_2 = \beta_{20} + \beta_2^T x$

$C3{:}\ z_3 = \beta_{30} + \beta_3^T x$

Probability

$P(Y = C_1)$

$P(Y = C_2)$

$P(Y = C_3)$

$$\sum_i P(Y = C_i) = 1$$

Softmax function

$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)} \text{ for } i = 1, \dots, K \text{ and } z = (z_1, z_2, \dots, z_K)^T$$

So we have $\sum_{i=1}^{K} \sigma(z)_i$=1

SoftMax Regression: $P(Y = k|X = x) = \frac{\exp(\beta_{k0}+\beta_k^T x)}{\sum_{l=1}^{K} \exp(\beta_{l0}+\beta_l^T x)} \quad k = 1, \dots . K$

Logistic Regression – multiclass classification

<u>Multi-class classification</u>

$C1:\ z_1 = \beta_{10} + \beta_1^T x$

$C2:\ z_2 = \beta_{20} + \beta_2^T x$

$C3:\ z_3 = \beta_{30} + \beta_3^T x$

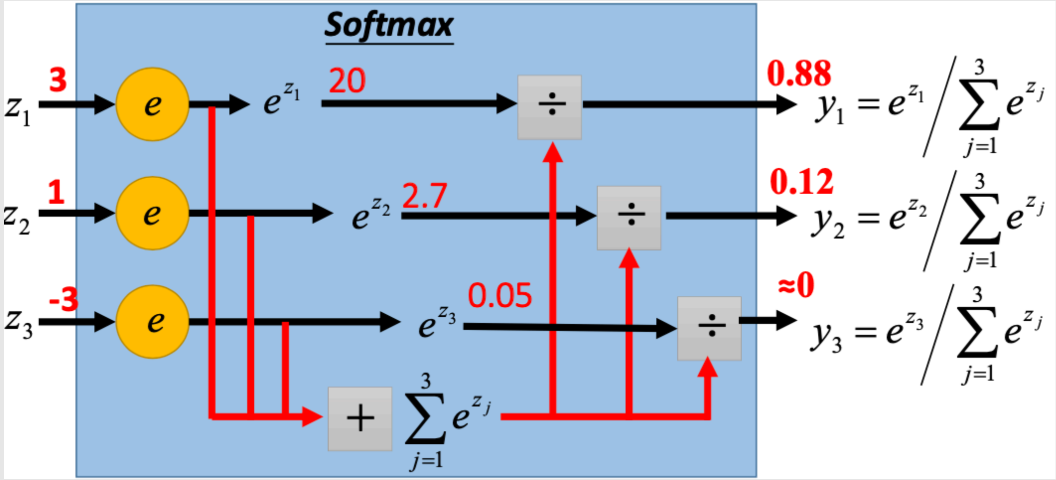
Probability

$P(Y = C_1)$

$P(Y = C_2)$

$P(Y = C_3)$

$$\sum_i P(Y = C_i) = 1$$

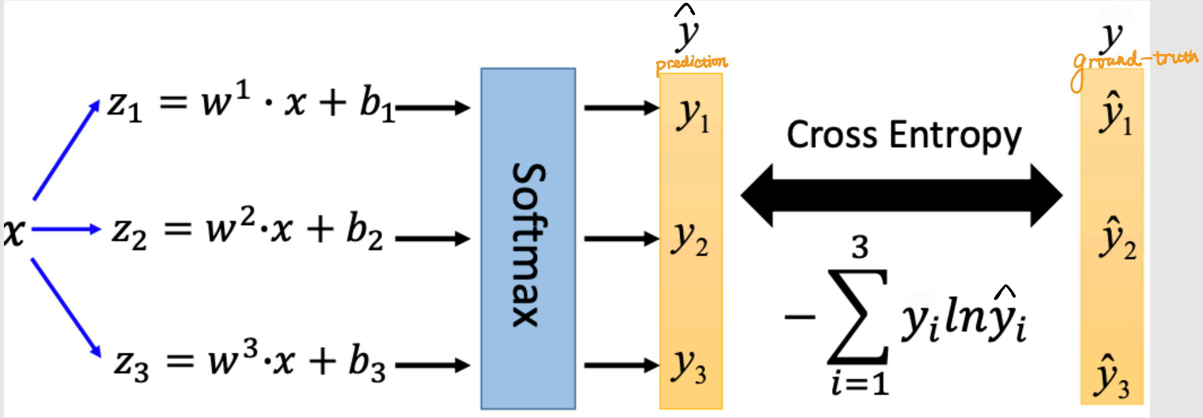


$P(Y=C_1|x)$

$P(Y=C_2|x)$

$P(Y=C_3|x)$

Logistic Regression – multiclass classification

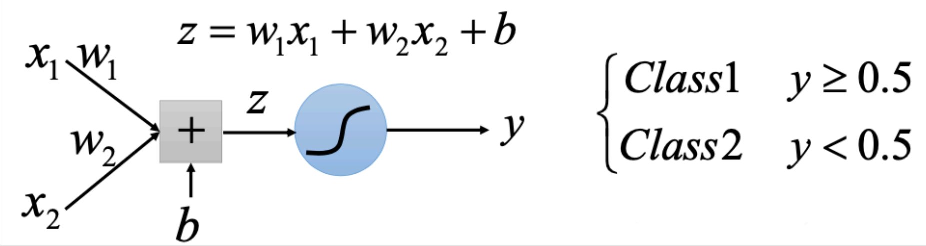


cross-entropy loss $-\sum_{k=1}^{3} y_k \log \hat{y}_k = -\sum_{k=1}^{3} I(Y=c_k) \log P(Y=c_k|x)$

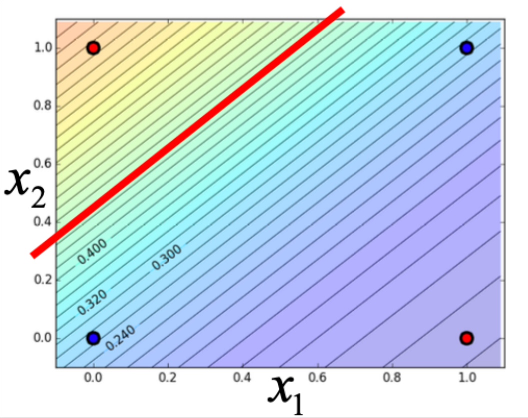
Data in class1: $y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ Data in class2: $y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ Data in class3: $y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Limitation of logistic regression

Logistic regression has a linear decision boundary



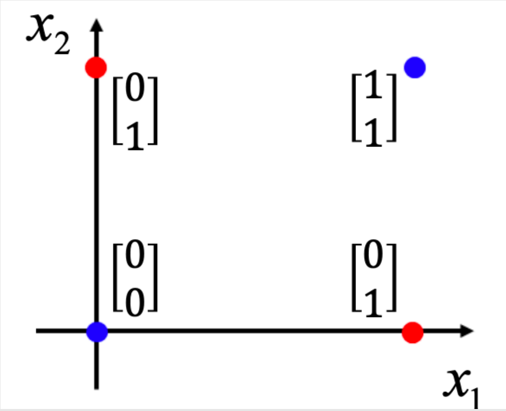
Input Feature		Label
x_1	x_2	
0	0	Class 2
0	1	Class 1
1	0	Class 1
1	1	Class 2

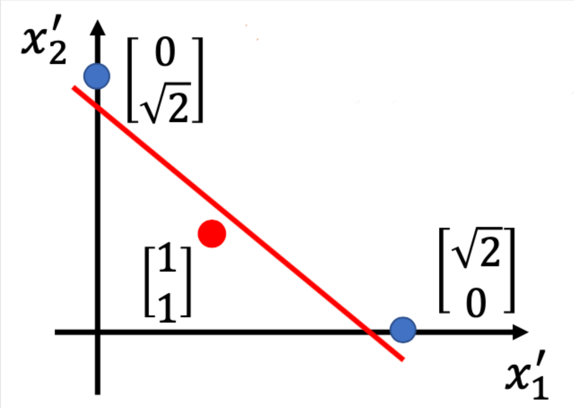


Limitation of logistic regression

Feature Transformation:

x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$, x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

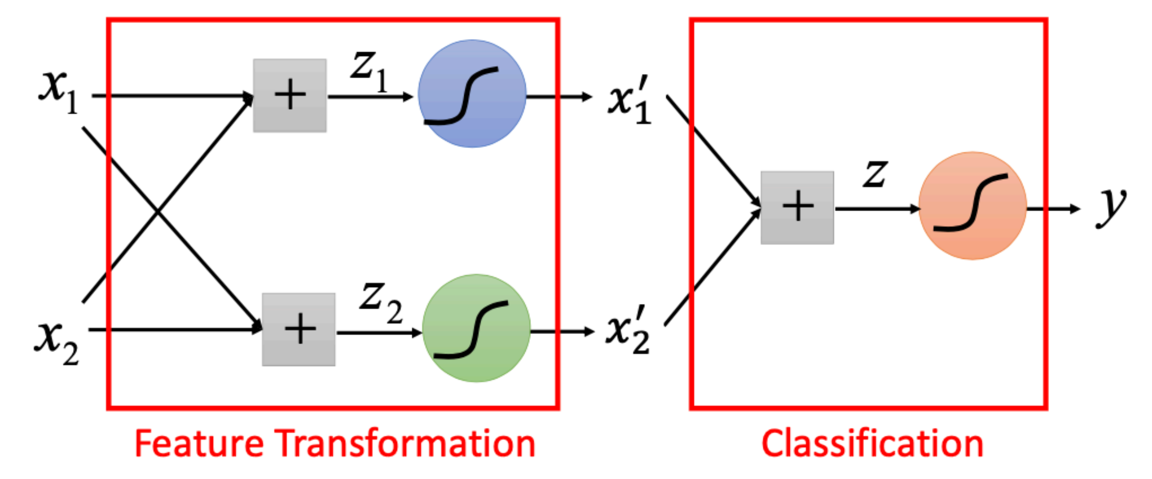




Not always easy to find a good transformation

Limitation of logistic regression

Cascading logistic regression models



Multilayer neural network!