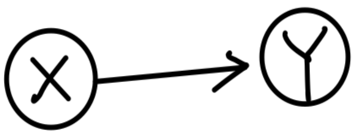
Logistic Regression

`Machine Learning`

Binary Classification



Given an x, we want to predict the probability

$$\hat{y} = P(y = 1|x)$$

where $x \in R^{n_x}$

Logistic Regression

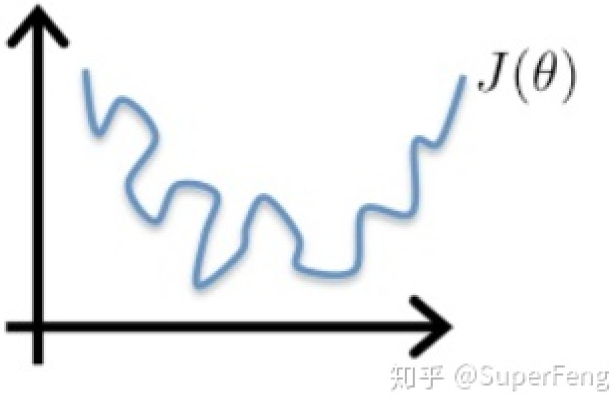
Logistic Regression tries to approximate the above probability using the following mathematic model:

$$\begin{aligned}\hat{y} &= P(y = 1|x, w, b) \\ &= \sigma(w^T x + b)\end{aligned}$$

where $w \in R^{n_x}$, and $b \in R$.

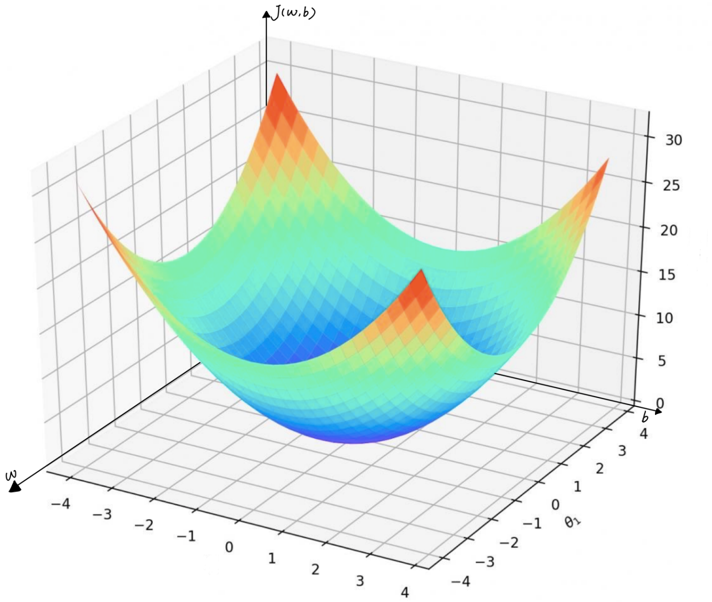
Loss Function

Given a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$. For logistic regression, a simple mean-squared error won't help the gradient descent work well. Because if we use the min-squared error, the curve of the loss function will be like this:



which is not convex.
Instead we use a loss fun:

$$\mathcal{L}(\hat{y}, y) = -\left(y \log \hat{y} + (1-y)\log(1-\hat{y})\right)$$



which is a convex function.##Cost Function
The cost function is the average loss for the whole training set:

$$\begin{aligned} \mathcal{J}(w, b) &= \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\ &= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + (1-y)\log(1-\hat{y}^{(i)})\right] \end{aligned}$$

Same Effect as MAP(Maximize a Posteriori Probability)

The probability function for y given x is

$$p(y|x) = \begin{cases} \hat{y} & y = 1 \\ 1 - \hat{y} & y = 0 \end{cases}$$

which can be written in this form

$$p(y|x) = \hat{y}^y (1 - \hat{y})^{(1-y)}$$

And to maximize the posteriori probability is equivalent to minimize the negative logarithm of this probability:

$$-\log p(y|x) = -\left(y \log \hat{y} + (1-y)\log(1-\hat{y})\right)$$

So **minimize $\mathcal{L}(\hat{y}, y)$ is equivalent to maximize** $p(y|x)$.

Gradient Descent for Logistic Regression

The cost function $\mathcal{J}(w, b)$ is convex, so we can get its global minimum using gradient descent regardless the initial values for w and b.
we update w and b repeatedly:

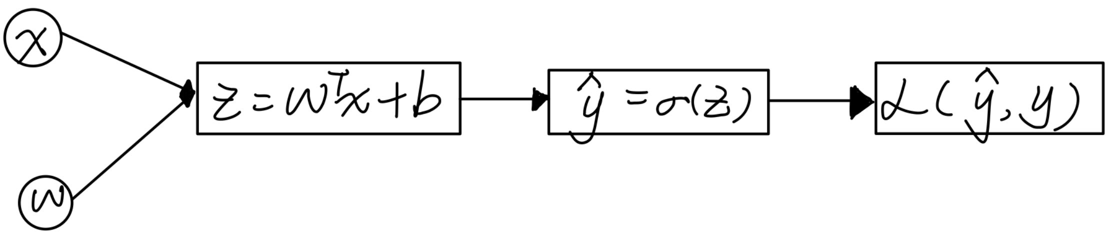
$$w := w - \alpha \frac{d\mathcal{J}(w, b)}{\partial w}$$

$$b := b - \alpha \frac{d\mathcal{J}(w, b)}{\partial b}$$

where α is the learning rate.

Gradient Descent for A Single Example

Giving the following computational graph:



We can get the derivative

$$dz = \hat{y} - y$$

Explanation:

$$
\begin{aligned}
dz = \frac{\mathcal{L}(\hat{y}, y)}{dz} &= \frac{\mathcal{L}(\hat{y}, y)}{d\hat{y}} \cdot \frac{d\hat{y}}{dz} \\
&= -\frac{ylog\hat{y} + (1-y)log(1-\hat{y})}{d\hat{y}} \cdot \frac{\sigma(z)}{dz} \\
&= -\left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \cdot \frac{\sigma(z)}{dz} \\
&= \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \frac{\sigma(z)}{dz}
\end{aligned}
$$

while

$$
\begin{aligned}
\frac{\sigma(z)}{dz} &= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{e^{-z}}{1+e^{-z}} \cdot \frac{1}{1+e^{-z}} \\
&= (1-\hat{y}) \cdot \hat{y}
\end{aligned}
$$

So we have

$$
dz = \hat{y} - y
$$

And from that we can get

$$
\begin{aligned}
dw &= \frac{d\mathcal{L}}{dz} \nabla_w z \\
&= (\hat{y} - y) \cdot x \\
db &= \frac{d\mathcal{L}}{dz} \frac{dz}{db} \\
&= dz = \hat{y} - y
\end{aligned}
$$

Gradient Descent for m Examples

As we already know the cost function for a m-size training set is

$$
\mathcal{J}(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})
$$

So we have

$$
\begin{aligned}
dw = \frac{d\mathcal{J}}{dw} &= \frac{1}{m} \sum_{i=1}^{m} \frac{\partial \mathcal{L}(\hat{y}^{(i)}, y^{(i)})}{dw} \\
&= \frac{1}{m} \sum_{i=1}^{\overset{(i)}{m}} \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot x^{(i)}
\end{aligned}
$$

Vectorizing Logistic Regression

The training set is represented by a $n_x \times m$ matrix:

$$
X = \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix}
$$

The weight matrix is represented by a vector:

$$
w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n_x} \end{bmatrix}
$$

The bias matrix is represented by a $1 \times m$ matrix:

$$
b = \begin{bmatrix} b & b & \cdots & b \end{bmatrix}
$$

The latent variable is represented by a $1 \times m$ matrix:

$$
Z = \begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix}
$$

.

So in all, the logistic regression in a matrix form is:

$$
Z = w^T X + b
$$

Vectorizing Logistic Regression Gradient descent

We use a $1 \times m$ matrix to represent the predicted ys:

$$
\hat{Y} = \begin{bmatrix} \hat{y}^{(1)} & \hat{y}^{(2)} & \cdots & \hat{y}^{(m)} \end{bmatrix}
$$

Similarly real ys are also represented by a $1 \times m$ matrix:

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$$

Then the vectorized dz for all zs are represented by a $1 \times m$ matrix:

$$dZ = \hat{Y} - Y$$

For the overall cost function, we need to add the gradient for each training example altogether.
So we get:

$$db = \frac{1}{m} np.sum(dZ)$$

$$\begin{aligned} dw &= \frac{1}{m} \left(dz^{(1)} x^{(1)} + dz^{(2)} x^{(2)} + \cdots + dz^{(m)} x^{(m)} \right) \\ &= \frac{1}{m} X dZ^T \end{aligned}$$

where db is a scale and dw is a $n \times 1$ vector.

Implementing Logistic Regression

$Z = w^T X + b$
$\hat{Y} = \sigma(Z)$
$dZ = \hat{Y} - Y$
$dw = \frac{1}{m} X dZ^T$
$db = \frac{1}{m} np.sum(dZ)$
$w := w - \alpha dw$
$b := b - \alpha db$