CH4: CLASSIFICATION

Lecturer: NGIN KIMLONG

OBJECTIVES

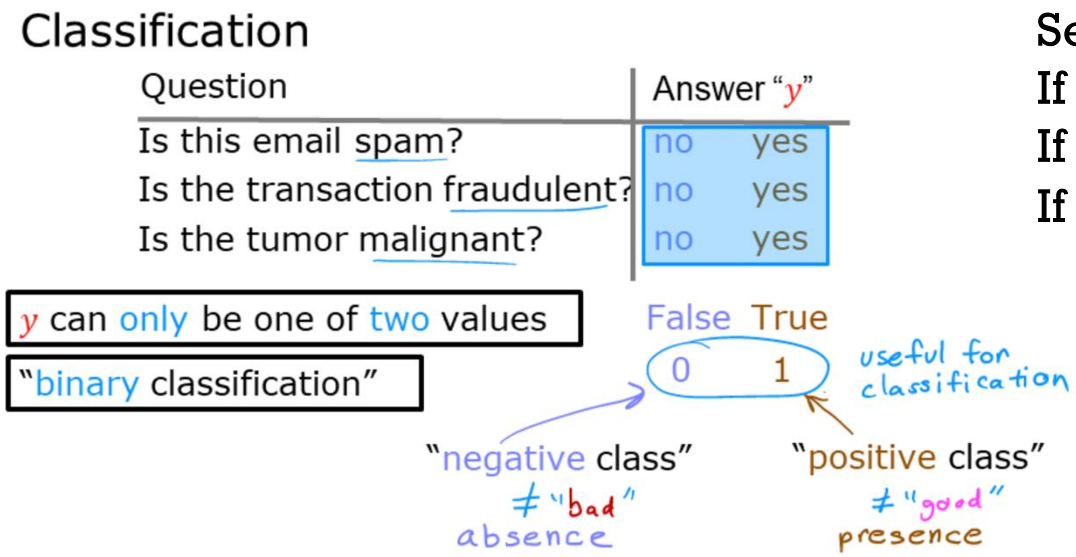
- Prediction by using logistic regression
- Problem of overfitting and solution with regularization
- Using logistic regression for binary classification

CONTENT

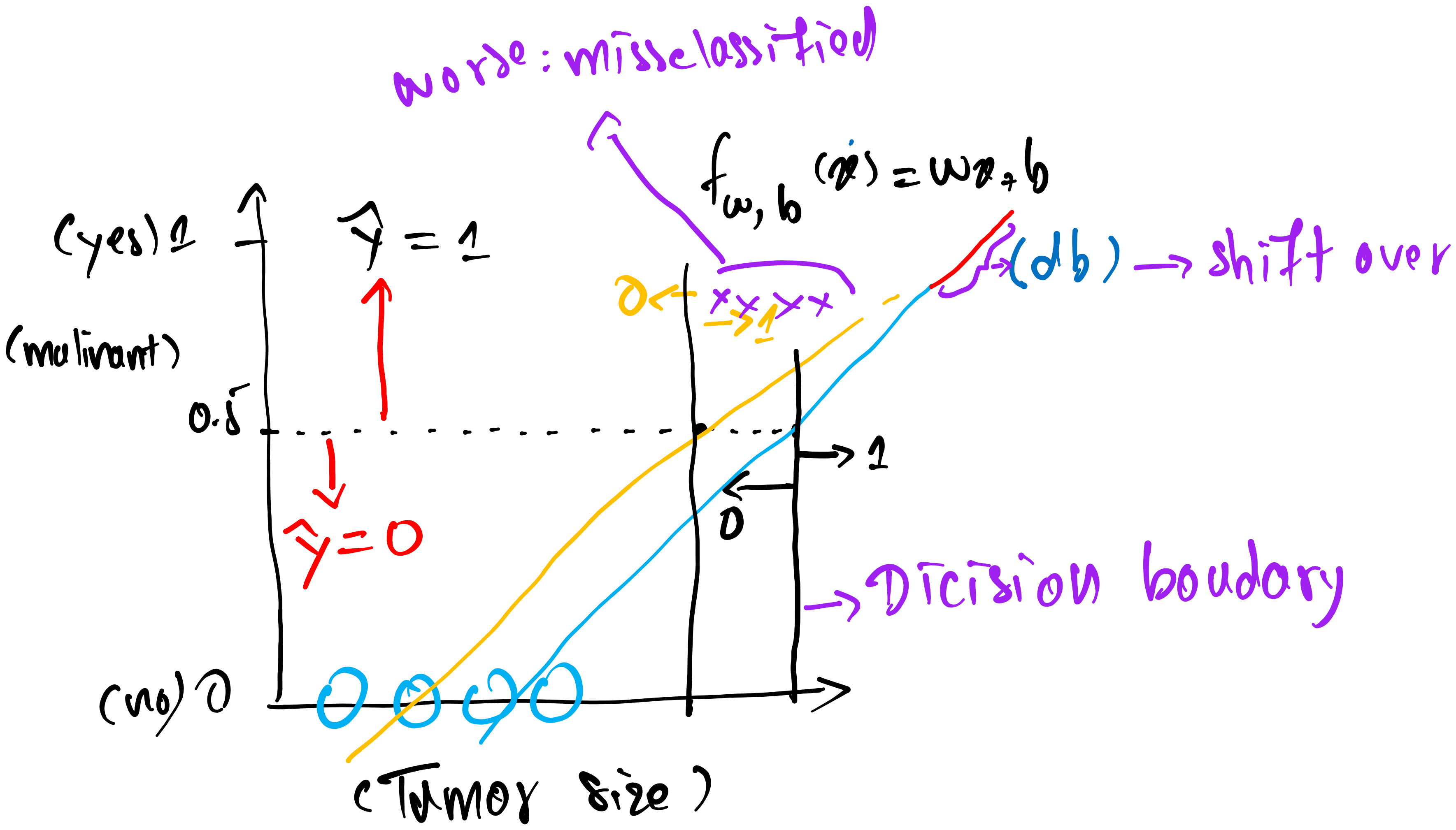
1. Classification with logistic regression
2. Logistic regression
3. Simplified cost function for logistic regression
4. Gradient descent for logistic regression
5. The problem of overfitting

1. CLASSIFICATION WITH LOGISTIC REGRESSION

- The classification with problems that result only two result "True" or "False" called binary classification



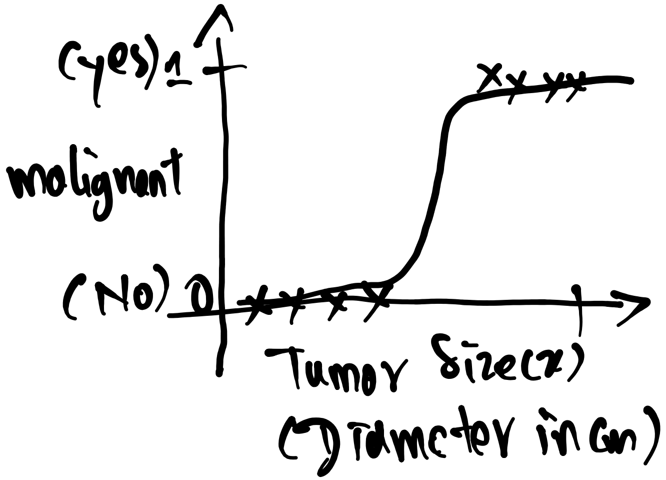
Set threshold = 0.5, if $\hat{y} \geq 0.5 \Rightarrow Malignant, but$
If $\hat{y} < 0.5 \Rightarrow not\ malignant$
If $(f_{w,b}(x) < 0.5 \Rightarrow \hat{y} = 0$
If $(f_{w,b}(x) \geq 0.5 \Rightarrow \hat{y} = 1$



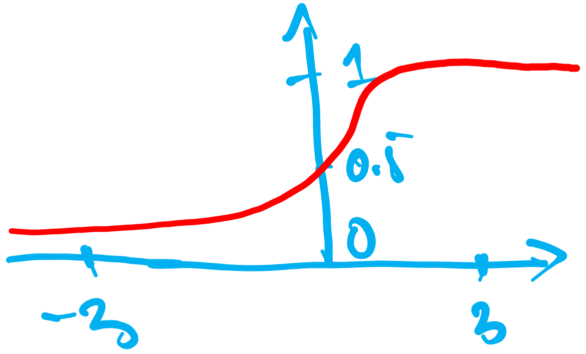
Noted that logistic regression model will produce two result, 1 and 0.
It also can avoid the missclassification.

2. LOGISTIC REGRESSION

- In this case we will use S-curve to evaluate features of model. It graphs as follow:

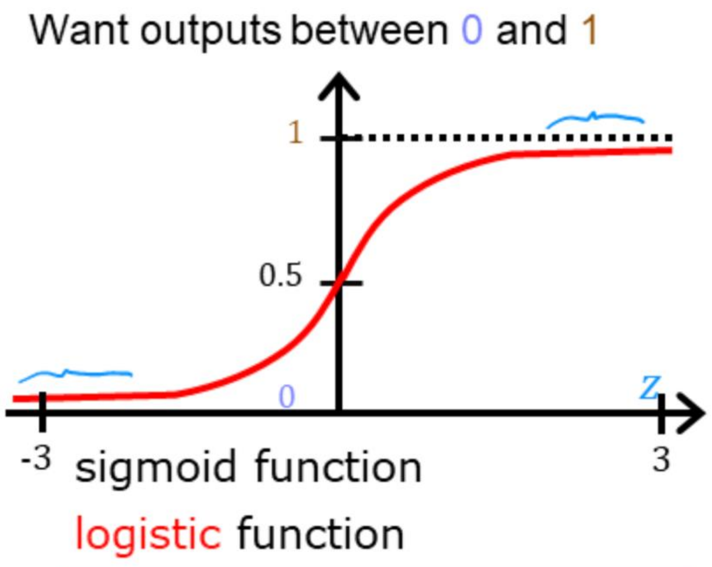


Sigmoid function will use in logstic regression, it graph as:



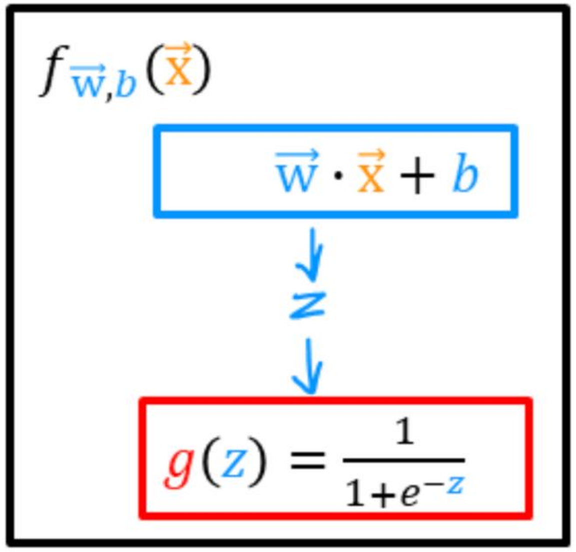
Sigmoid function (σ), this activation can result only 1 and 0.

$$g(z) = \frac{1}{1+ e^{-z}} \qquad 0 < g(z) < 1$$



outputs between 0 and 1

$$g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1$$



$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$

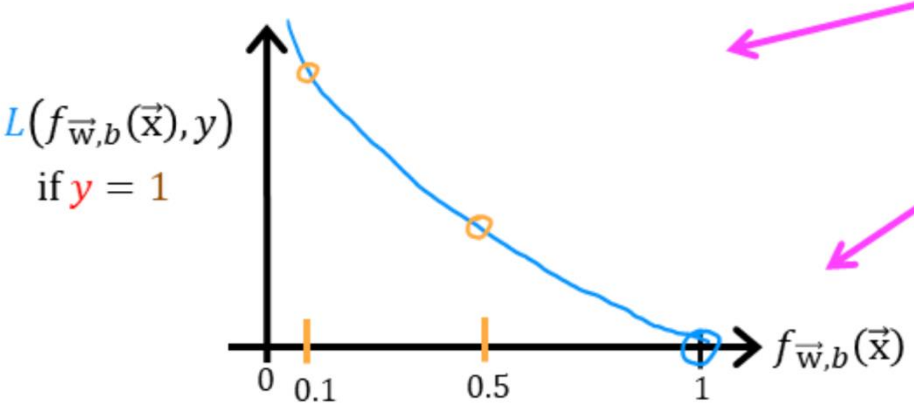
"logistic regression"

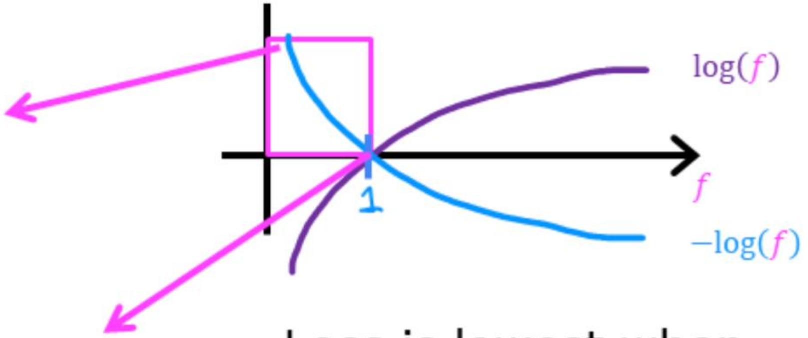
- Probability is $p(y = 0) + p(y = 1) = 1$
  - y = 70%, mean that it closes to 1
  - Y = 30% mean that it closes to 0
- Some paper: $f_{\vec{x},b}\,(\vec{x}) = P(y = 1\,|\vec{x}; \vec{w}, b)$
- *W, and b are parameters of probability that y*

3. SIMPLIFIED COST FUNCTION FOR LOGISTIC REGRESSION

Logistic Loss Function

$$L\left(f_{\vec{w},b}(\vec{x}), y\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x})\right) & \text{if } y = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x})\right) & \text{if } y = 0 \end{cases}$$





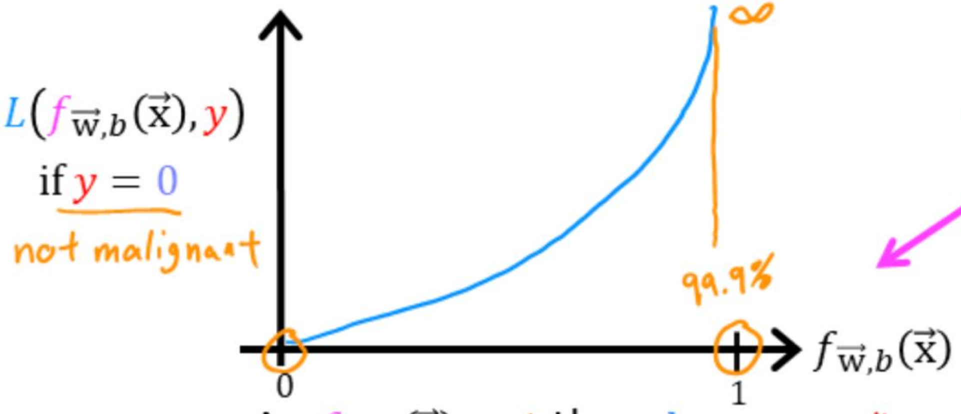
As $f_{\vec{w},b}(\vec{x}) \to 1$ then loss $\to 0$

Loss is lowest when $f_{\vec{w},b}(\vec{x})$ predicts close to true label y.

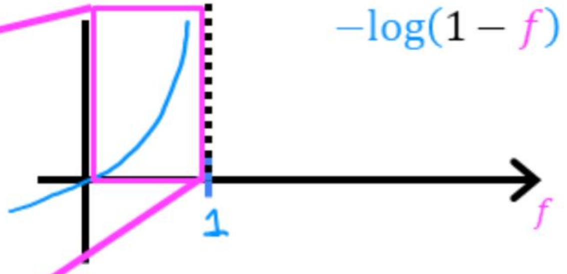
Logistic Loss Function

$$L\left(f_{\vec{w},b}(\vec{x}), y\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x})\right) & \text{if } y = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x})\right) & \text{if } y = 0 \end{cases}$$

As $f_{\vec{w},b}(\vec{x}) \to 0$ then loss $\to 0$



As $f_{\vec{w},b}(\vec{x}) \to 1$ then loss $\to \infty$



The further prediction $f_{\vec{w},b}(\vec{x})$ is from target y, the higher the loss.

Cost

cost

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2} \underbrace{L\left(f_{\vec{w},b}(\vec{x}), y\right)}_{\text{loss}}$$

$$= \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x})\right) & \text{if } y = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x})\right) & \text{if } y = 0 \end{cases}$$

convex
↳ can reach a global minimum

find w, b that minimize cost J

- Noted: The square error cost is not a good choice for logistic regression.

Squared Error Cost

cost

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2}_{\text{loss}}, \quad \hat{y}^{(i)} = f_{\vec{w},b}(\vec{x}^{(i)})$$

average of training set

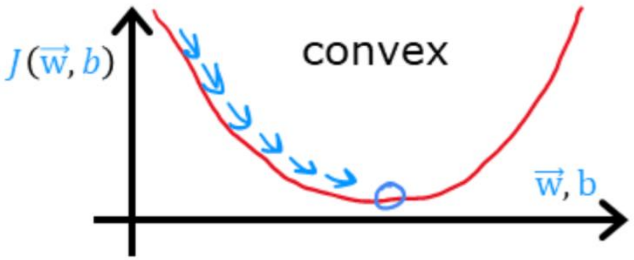
loss

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

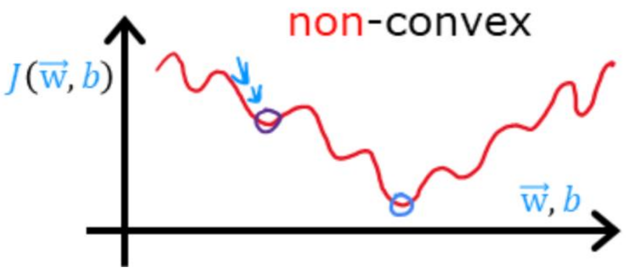
single training example

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

linear regression $f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$



logistic regression $f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$



a. Simplified loss function

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) \ if y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) if y^{(i)} = 0 \end{cases}$$

Hence,

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)} log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1 - y^{(i)}) \ log\left(1 - \ f_{\vec{w},b}(\vec{x}^{(i)})\right)$$

- $if y^{(i)} = 1 => L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = -\log(f_{\vec{w},b}\vec{x}))$
- $if y^{(i)} = 0 => L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = -\log(1 - f_{\vec{w},b}(\vec{x}))$

Simplified cost function is:

Loss

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)}\log(f_{\vec{w},b}(\vec{x}^{(i)})) - (1-y^{(i)})\log(1-f_{\vec{w},b}(\vec{x}^{(i)}))$$

Cost

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(f_{\vec{w},b}(\vec{x}^{(i)})) + (1-y^{(i)})\log(1-f_{\vec{w},b}(\vec{x}^{(i)}))\right]$$

4. GRADIENT DESCENT FOR LOGISTIC REGRESSION

<u>Gredient descent implementation</u>

- Previously, to select a fit parameters for logistic regression model, we need to find total value of parameters w, and b to descrease value of J(w,b)

a. Cost funciton

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

* Usual Gredient Descent

repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w},b)$$

$$b = b - \alpha\frac{\partial}{\partial b}J(\vec{w},b)$$

}

Noted:

Linear regression $f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$

Logistic regression $f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$

b. Gredient descent

repeat {

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right]$$

$$b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) \right]$$

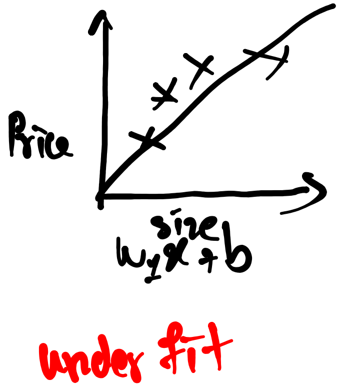
} Simultaneous updates

5. THE PROBLEM OF OVERFITTING

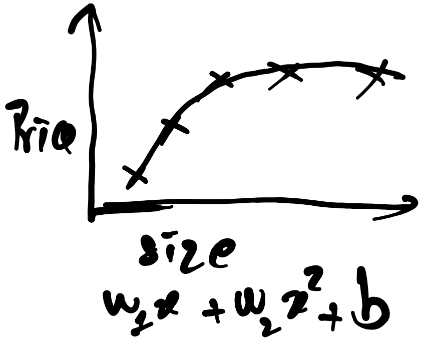
a. The problem of overfitting

Linear regression and Logistic Regression work very well for multi-tasks. But, in an application algorithm possibly faces the problem of overfitting such as: closely-related almost opposite problem.

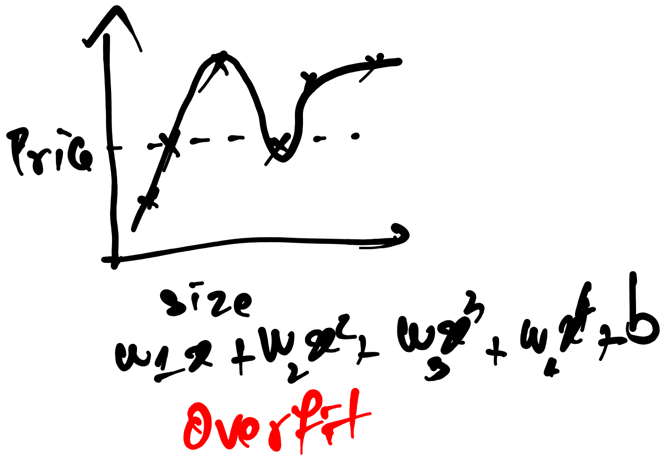
+ What's Overfitting?



. Does not fit the
Training set well.
"High Bias"



. Fits training set petty well
"Regularization"



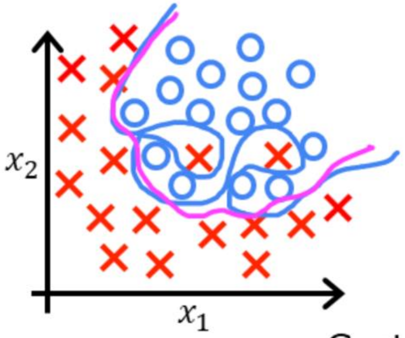
$f(x) = 28x - 385x^2 + 39x^3 - 174x^4 + 100$

. Fit the training set extremely well
"High Variance"

b. Classification

- Finding the tumor size to classify malignant or benign

Regularized logistic regression



$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + \cdots + b$$

$$f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}) = g(z) = \frac{1}{1+e^{-z}}$$

Overfitting

Cost function

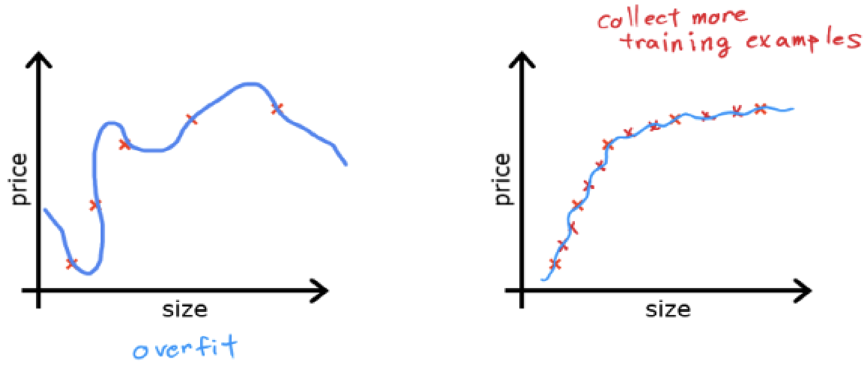
$$J(\vec{\mathrm{w}},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

To againts the problem of overfitting, we can:

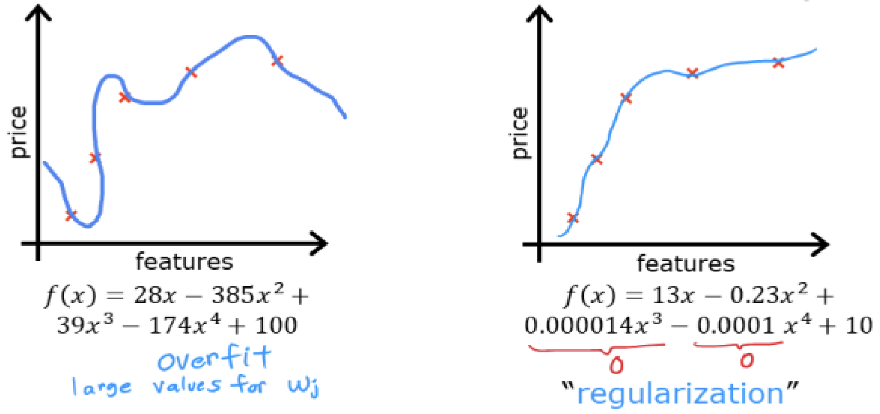
1. Collect more data
2. Collect features to include/exclude necessary or unnecessary
3. Allow algorithm to select itself features or eleminate features with big size(mean assign value 0) or assign small value to w parameter.

Overfitting

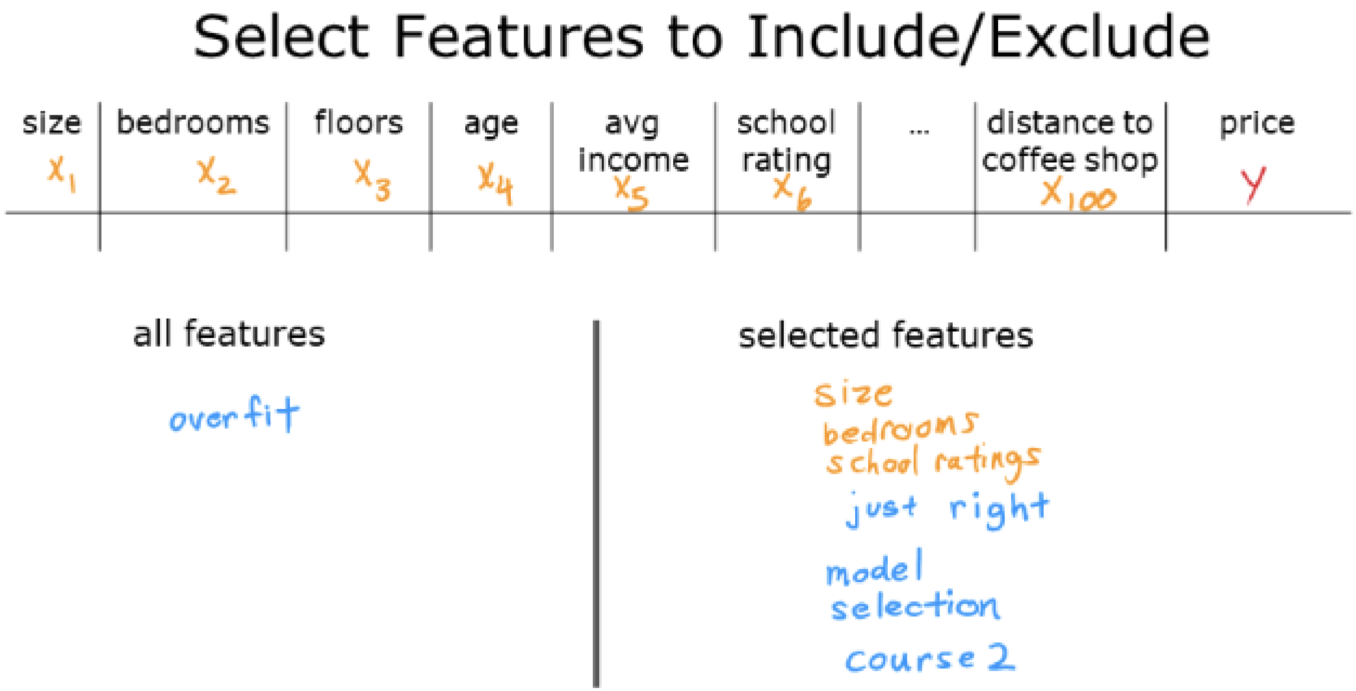
Collect more Training Data



Reduce the size of parameters w_j, b



If we have more parameters, How to assign value to parameters to penalize?



- Because n = 100 features, it difficults to drop and keep some features. To do so, We need penalize on features by adding new term Lamda (λ).
- **<u>Cost function:</u>**

$$J(\vec{w},b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2 + \underbrace{\lambda}_{\substack{\text{"lambda"}\\ \text{regularization parameter}}}\overbrace{\sum_{j=1}^{n} w_j^2}^{\text{regularization term}} + \lambda b^2\right]$$

can include or exclude b

$\lambda > 0$

- If λ = 0 => model is overfit
- If λ enomors $\lambda = 10^{10}$ => Model is underfit
- If λ not small or big => Model will well

- Regularized linear regression:

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

Gradient descent

Repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(w_j,b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

$j = 1, \ldots, n$

$$b = b - \alpha\frac{\partial}{\partial b}J(w_j,b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

}

don't have to regularize b

Cost function for regularized logistic regression

For regularized **logistic** regression, the cost function is of the form

$$J(\mathbf{w},b) = \frac{1}{m} \sum_{i=0}^{m-1} \left[-y^{(i)} \log\left(f_{\mathbf{w},b}\left(\mathbf{x}^{(i)} \right) \right) - \left(1 - y^{(i)}\right) \log \left(1 - f_{\mathbf{w},b}\left(\mathbf{x}^{(i)} \right) \right) \right] + \frac{\lambda}{2m} \sum_{j=0}^{n-1} w_j^2 \tag{3}$$

where:

$$f_{\mathbf{w},b}(\mathbf{x}^{(i)}) = sigmoid(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \tag{4}$$

Compare this to the cost function without regularization (which you implemented in a previous lab):

$$J(\mathbf{w},b) = \frac{1}{m}\sum_{i=0}^{m-1} \left[(-y^{(i)} \log\left(f_{\mathbf{w},b}\left(\mathbf{x}^{(i)} \right) \right) - \left(1 - y^{(i)}\right) \log \left(1 - f_{\mathbf{w},b}\left(\mathbf{x}^{(i)} \right) \right)\right]$$

As was the case in linear regression above, the difference is the regularization term, which is $\frac{\lambda}{2m} \sum_{j=0}^{n-1} w_j^2$

Including this term encourages gradient descent to minimize the size of the parameters. Note, in this example, the parameter b is not regularized. This is standard practice.

- Regularized Logistic Regression

$$\min_{\vec{w},b} J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

Gradient descent

Repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(w_j,b) \qquad j=1\ldots n$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

no Σ

one feature j

$$b = b - \alpha\frac{\partial}{\partial b}J(w_j,b)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

}

don't have to regularize b

Gradient descent for logistic regression

repeat {

looks like linear regression!

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_j^{(i)} \right]$$

$$b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) \right]$$

} simultaneous updates

Same concepts:

- Monitor gradient descent (learning curve)
- Vectorized implementation
- Feature scaling

Linear regression $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

Logistic regression $f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1 + e^{(-\vec{w}\cdot\vec{x}+b)}}$

HOMEWORK