WEEK 3

Logistic Regression

Regularization

WEEK 3

Classification

$y = 0$ or 1

Logistic regression: $0 \le h_\theta(x) \le 1$

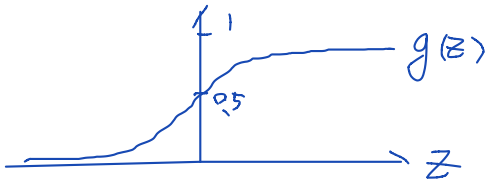
Logistic regression — hypothesis representation

model: want $0 \le h_\theta(x) \le 1$

$h_\theta(x) = g(\theta^T x)$

$= \frac{1}{1 + e^{-z}}$ ← Sigmoid or logistic function

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



for binary classification: $h_\theta(x) = P(y=1 \mid x;\theta)$

$P(y=0 \mid x;\theta) + P(y=1 \mid x;\theta) = 1$

Logistic regression — decision boundary

$h_\theta(x) = P(y=1 \mid \theta; x)$ { predict $y=1$ if $h_\theta(x) \geq 0.5$
- - - - $y=0$ if $h_\theta(x) < 0.5$

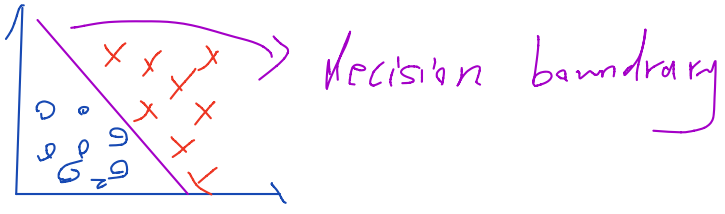
$g(z) = \frac{1}{1+e^{-z}}$

$g(z) \geq 0.5$ when $z \geq 0$

↳ $\theta^T x \geq 0$

$g(z) < 0.5$ when $z < 0$

↳ $\theta^T x < 0$



Logistic regression — cost function

Train set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots (x^{(m)}, y^{(m)})\}$

$x = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$ $x_0 = 1$, $y \in \{0, 1\}$

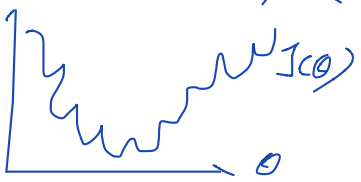
m train sets, n features

$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ how to choose parameters θ ?

Cost function

Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

& if we use the above same cost function for Logistic regression,
the cost function will be "non-convex" (has many local minimums)

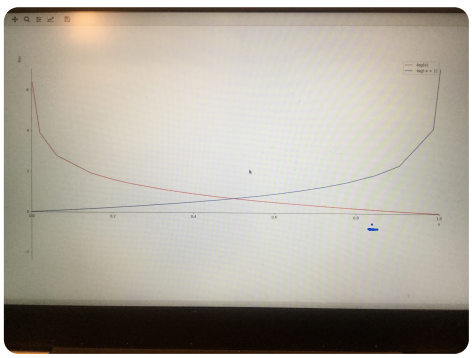


which is not garenteed to converge to global minimum
Thus we have to modify the cost function for logistic regression

defination:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x), y)$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$



Plot of cost

Simplified Cost function and gradient decerent

Logistic regression cost function: $J_\theta = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$

Simplification:

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

then:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right]$$

To fit parameters θ: $\min_\theta J(\theta)$

To make a prediction giving new x: $h(\theta) = \frac{1}{1+e^{-\theta^T x}}$

Gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

then:

$$\theta_j := \theta - \alpha \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad [\text{Sim update all } \theta_j]$$

Andrew Ng did put $\frac{1}{m}$ here, which I think is a mistake

Looks exactly the same as linear regression!

Except the definition of hypothesis $h_\theta(x)$ is diff.

Feature scaling also works for logistic regression!

Vectorized implementation:

$$\theta := \theta - \frac{\alpha}{m} X^T \cdot (g(X\theta) - \vec{y})$$

Logistic regression — adv. optimization

given θ, we have code that Compute:

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ for $j = 0, 1, \cdots n$

Optimization algorithms

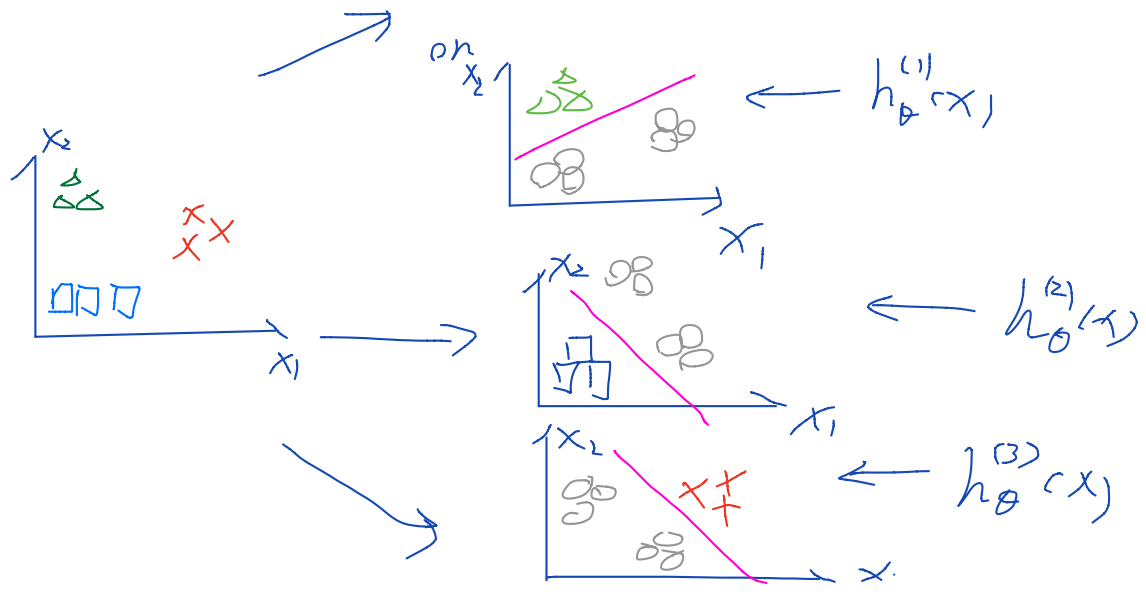
- Gradient decent
- Conjugate gradient
- BFGS
- L-BFGS

advantages:
- No need to manually pick α
- often faster than gradient decent

Disadvantages:
- more complex

Multi-class classification

one-vs-all (rest) classification



$h_\theta^{(i)}(x) = P(y = i \mid x; \theta)$, $i = 1, 2, 3$

one-vs-all

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$

On a new output x, to make a prediction, pick the class i that maximizes: $\max_i h_\theta^{(i)}(x)$

Problem of over fitting

Overfitting:

If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) \approx 0$), but fail to generalize to new example

Addressing overfitting

Options:

1. Reduce number of features
   a. Manually select which features to keep
   b. Model selection algorithm (later)
2. Regularization
   a. keep all features, but reduce magnitude / values of θ_j
      ↳ Works well when having a lot of features, each of which contributes a bit to prediction y

Regularization — cost function

Regularization:

Small values for parameters $\theta_0, \cdots \theta_n$

↳ "simpler" hypothesis's

↳ less prone ~~to~~ over fitting

regularized Cost function

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

λ: regularization parameter

omitted θ_0 convention add θ_0 doesn't make any diff

λ: too larger ⟶ underfit .

Regularized linear regression:

① gradient decent:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$$

$(j = ~~0~~, 1, 2, 3, \cdots n)$

$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

usually α, λ small, m large, so $1 - \alpha\frac{\lambda}{m} < 1$

② Normal Equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}_{m \times n+1} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$\min_\theta J(\theta)$

↳ $\theta = (X^TX + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix})^{-1}X^Ty$

$(n+1)\times(n+1)$

Non-invertibility

if $m \le n$, $[X^T X]$ non-invertible

if $\lambda > 1$, then $(X^T X + \lambda \begin{bmatrix} 0 & & \\ & 1 & \\ & & 1 \end{bmatrix})^{-1}$ is garanted to be invertiable

Regularized logistic regression

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right]$$

regularize

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Gradient descent:

Same as linear regression, except that hypothesis's func $h_\theta(x)$ is different.