WEEK - 3

Classification Problem o/1

Eg: Email: Spam/ Not spam?

Tumor (Malignent/ Benign)

Online Transactions (Fradd/NOL)

$y \in \{0, 1\}$

Binary class

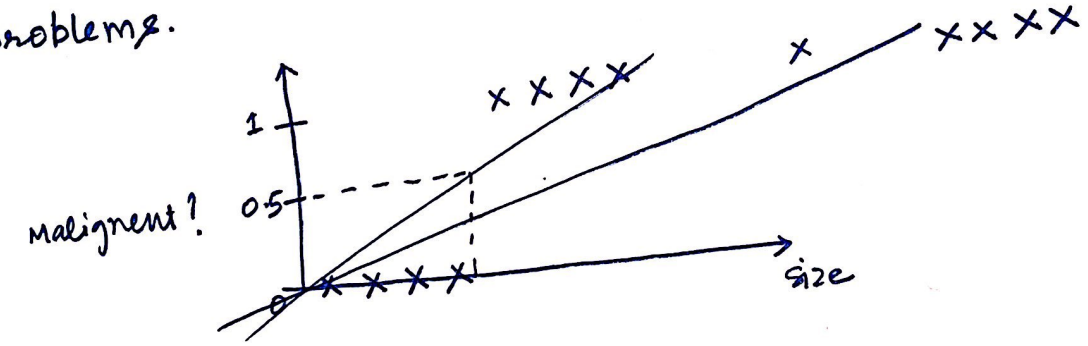
0: Negative (eg. ~~begin~~ benign)

1: Positive (eg. Malignant)

$y \in \{0, 1, 2, 3 \dots\}$

Multi-class classification problem

Linear regression cannot be applied to classification problems.



Logistic Regression: classification problem.

$$0 \leq h_\theta(x) \leq 1$$

• Hypothesis : Logistic Regression Representation :

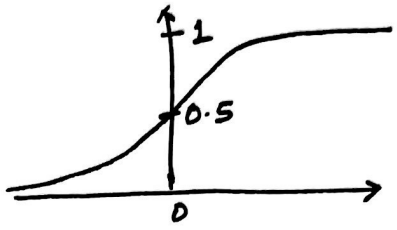
Want $0 \leq h_\theta(x) \leq 1$

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

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

sigmoid fn.

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



→ sigmoid fn.

→ Logistic fn.

→ Interpretation of Hypothesis Output :

$h_\theta(x)$ = estimated prob. that $y = 1$ on o/p.

Example: if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor size} \end{bmatrix}$

$$h_\theta(x) = 0.7$$

Tell patient that 70% chance of tumor being malignant

→ $h_\theta(x) = p(y=1 \mid x; \theta)$

→ $p(y=0 \mid x; \theta) = 1 - h_\theta(x)$

} "probability that $y=1$, given x, parameterized by θ"

Decision Boundary

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

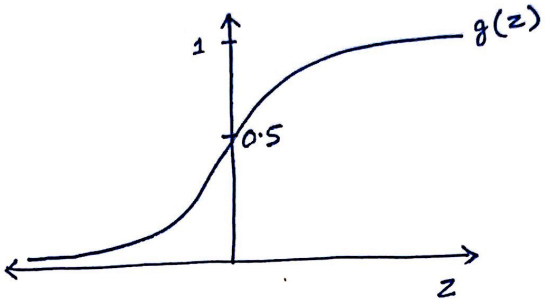
$g(z) = \frac{1}{1+e^{-z}}$

Predict

suppose "$y=1$" if $h_\theta(x) \geq 0.5$

"$y=0$" if $h_\theta(x) < 0.5$

$h_\theta(x)$ gives a probability that $y=1$



$g(z) \geq 0.5$

when $z \geq 0$

$h_\theta(x) = g(\theta^T x) \geq 0.5$

whenever

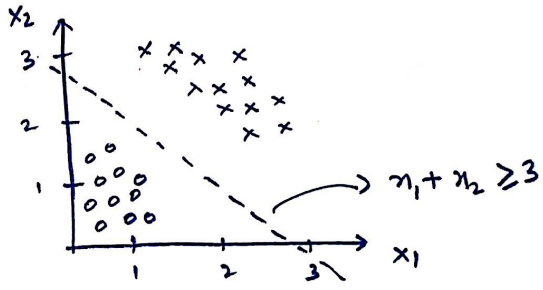
$\theta^T x \geq 0$

Example:

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + x_2 \theta_2)$

$-3 \quad +1 \quad +1$

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$



$\theta^T x$

→ predict '$y=1$' if $-3 + x_1 + x_2 \geq 0 \Rightarrow x_1 + x_2 \geq 3$

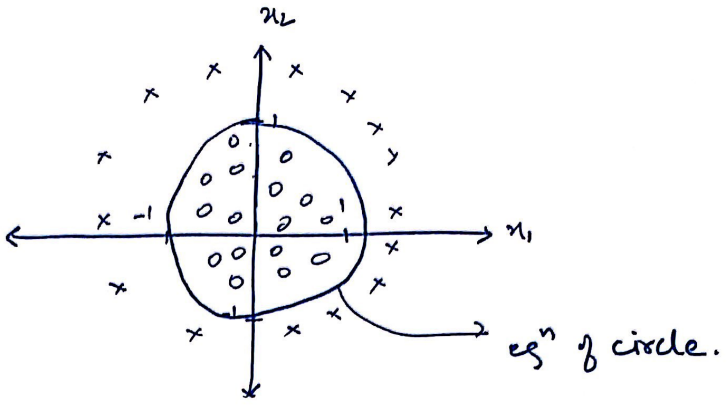
$\Downarrow$

Decision boundary is a property of $h_\theta(x)$ & not of the data set (or parameters)

Non-Linear Decision Boundaries :

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \rightarrow \text{instance}$$

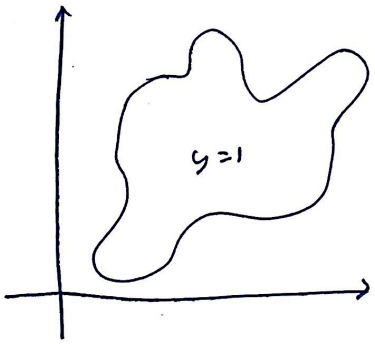


predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0 \longrightarrow \boxed{x_1^2 + x_2^2 \geq 1}$

**

Parameters (θ) defines the decision boundary

✓ Training set may be used to fit the parameters θ



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_2^2 + \theta_4 x_1^2 + \theta_5 x_1^2 x_2 + \theta_6 x_2^2 x_1 + \dots)$$

• COST FUNCTION :

Training set : $\{(x^1, y^1), (x^2, y^2) \dots (x^m, y^m)\}$

m examples. $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \mathbb{R}^{n+1}$ $\quad x_0 = 1$, $y \in \{0, 1\}$

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

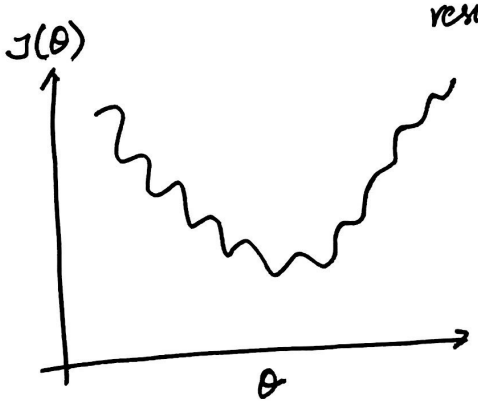
---

if we use the cost fn. of linear regression

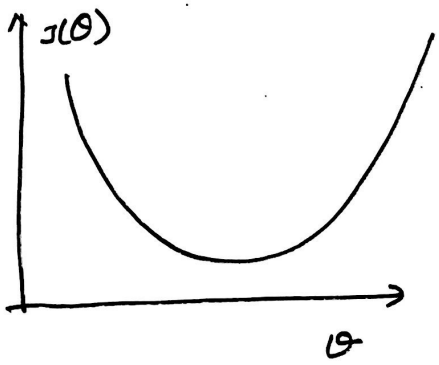
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y)^2$$

↘ Wait

↙ results in



Non-convex

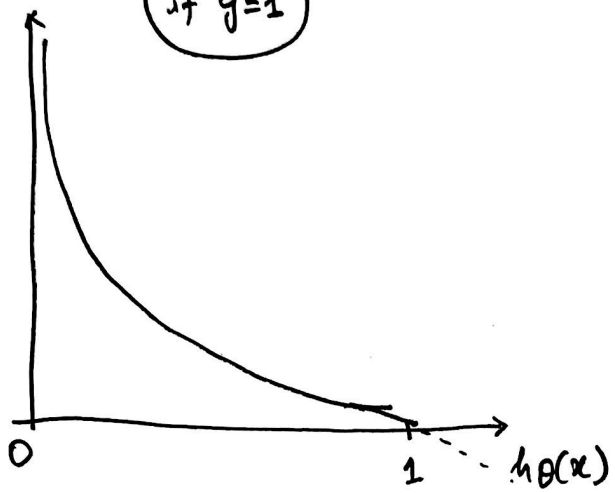


Convex

⇒ Logistic regression Cost function

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

if $y=1$

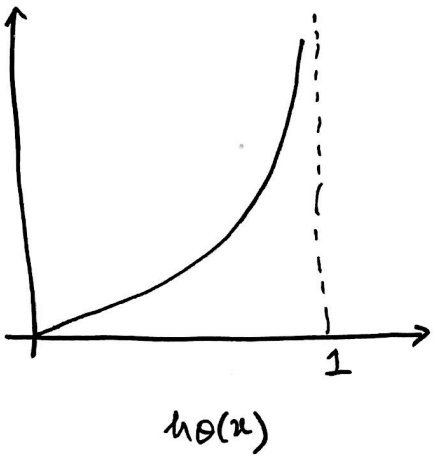


① cost $=0$ if $y=1$ & $h_\theta(x)=1$

② But as $h_\theta(x) \to 0$, cost $\to \infty$

captures intuition that if $h_\theta(x)=0$, (predict $P(y=1|x;\theta)=0$) but $y=1$, we will penalize learning algorithm by a very large cost.

if $y=0$



② cost $=0$ if $y=0$, $h_\theta(x)=0$

• Simplified Cost function and Gradient descent

$$\left[\text{cost}\left(h_\theta(x), y\right) = \underbrace{-y \cdot \log\left(h_\theta(x)\right)}_{} - \underbrace{(1-y)\log\left(1-h_\theta(x)\right)}_{}\right]$$

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

To fit parameter θ : $\longrightarrow$ Get θ ✓

$$\min_{\theta} J(\theta)$$

To make a prediction given new x ;

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

Gradient descent:

Repeat until convergence(num_iters):

{

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

}

#(simultaneously update all θ_j)

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

derivative of $\frac{\partial J(\theta)}{\partial \theta_j}$ for logistic regression as well.

Advanced optimization Algorithms:

$\partial J = (x' * (x * theta - y))/m$

different optimizations Algorithms:

- → gradient descent
- → conjugate descent
- → BFGS
- → L-BFGS

No need to manually pick α.
often faster than gradient descent
disadvantage: More compeex.

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial J}{\partial \theta_1} = 2(\theta_1 - 5)$$

$$\frac{\partial J}{\partial \theta_2} = 2(\theta_2 - 5)$$

Cost Function in Octave

```
function [jval, gradient] = costfunction(theta).

    jval = (theta(1) - 5)^2 + (theta(2) - 5)^2 ;

    gradient = zeros(2,1) ;

    gradient(1) = 2 * (theta(1) - 5)

    gradiest(2) = 2 * (theta(2) - 5)
```

Data structure.

options = optimset ('GradObj', 'on', 'MaxIter', '100');

initialTheta = zeros(2,1);

[optTheta, functionVal, exitFlag] ...

= fminunc(@costFunction, initialTheta, options);

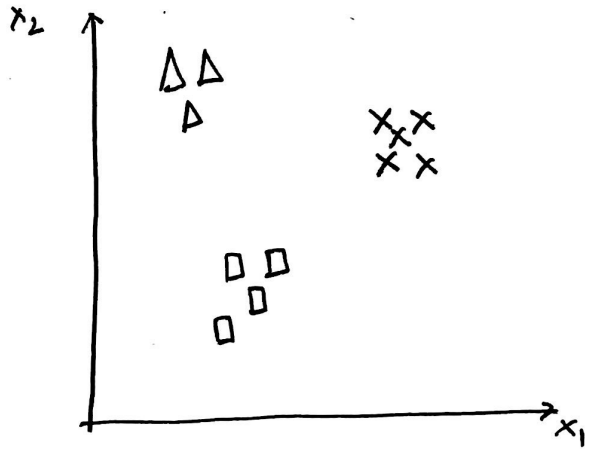
fminunc: fn. minimization unconstrained

$\theta \in \mathbb{R}^d$ $d \geq 2$

- **Multi-class classification Problem.:**

example: ① Email foldering / tagging: Work ($y=1$), freinds ($y=2$), family ($y=3$), Hobby ($y=4$)

② Medical diagrams: Not ill ($y=1$), cold ($y=2$), flu. ($y=3$)

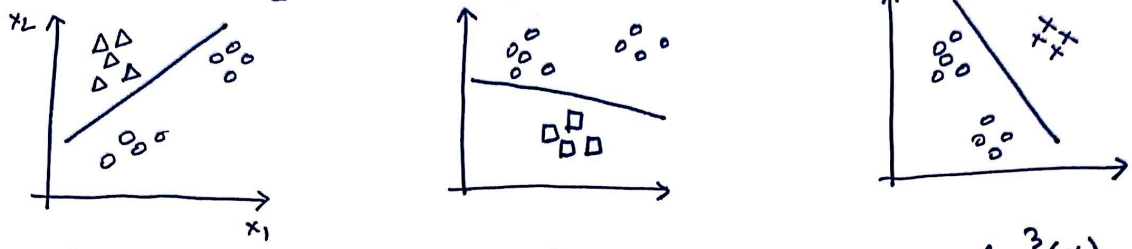


One-vs-all classification :

class 1 : △ $y=1$

class 2 : □ $y=2$

class 3 : × $y=3$



$h_\theta^1(x)$ $h_\theta^2(x)$ $h_\theta^3(x)$

Hypothesis 1

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

Probability that $y=i$ (belongs to class i)

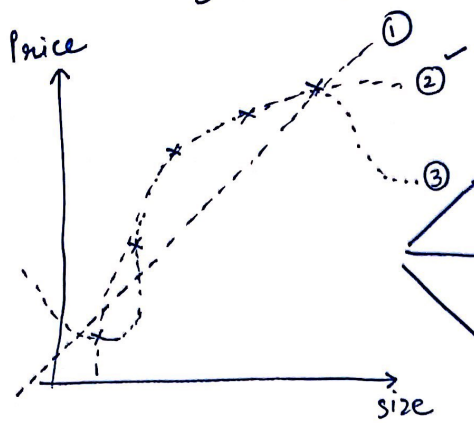
i ↑ class i

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

→ On a new input x, to make a prediction, pick a class i that maximizes $h_\theta^i(x)$: maximizing the prob.

$$\max_i h_\theta^i(x)$$

Solving the problem of Overfitting :



$h_\theta(x) = \theta_0 + \theta_1 (x)$

"Underfit", "High bias"

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit", "High variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples)

generalize → how well it applies to new data.

[(It may work very well on training data but doesn't work on actual data)]

Addressing Overfitting : options.

① Reduce number of features
- Manually select which features to keep.
- Model selection Algorithm (later)

② Regularization:
- Keep all the features, but reduce magnitude / values of parameters θ_j
- Works well when we have a lot of features, each of which contributes a bit to predicting y.

• Regularization : Cost function :

→ Small values for ~~values~~ parameter $(\theta_0 \dots \theta_n)$

- "Simpler" hypothesis
- less prone to overfitting

} Penalize all the parameters.

Example :

we penalize and make θ_3, θ_4 really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right)^2 + 1000\theta_3^2 + 1000\theta_4^2$$

$$\theta_3, \theta_4 \approx 0$$

Ex.:

- features : $x_0, x_1, x_2 \dots x_n$
- Parameters : $\theta_0, \theta_1, \theta_2 \dots \theta_{100}$

} we don't know which features/param. to penalize.

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

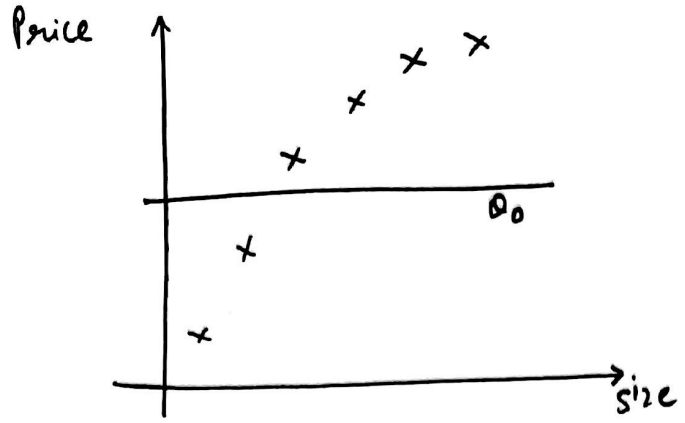
→ This will result in $\theta_j \approx 0$

$\sum_{i=1}^{n} \theta_j^2$ → regularization term.

λ → Regularization parameter [Not going to penalize θ_0. very little difference]

* if λ is set to extremely large value (for instance, $\lambda = 10^{10}$) then we might have "Underfit" case.

* while solving for θ, $J(\theta)$ will become min, $\theta_j \approx 0$.



our $\min_{\theta} J(\theta)$ will give values of $\theta_j \neq 0$

$\therefore \; h_\theta(x) = \theta_0$

we might end up "underfitting our data".

REGULARIZED LINEAR REGRESSION ($\min_{\theta} J(\theta)$)

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

Repeat until convergence

(1) {

$$\theta_0 := \theta_0 - \alpha\left[\frac{1}{m}\sum_{i=1}^{n}\left(h_\theta(x^i) - y^i\right).x_0^i\right] \rightarrow \frac{\partial J}{\partial \theta_0}$$

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

}

$$\theta_j := \theta_j\underbrace{\left(1 - \frac{\alpha.\lambda}{m}\right)}_{<1} - \alpha\left[\frac{1}{m}\sum\left(h_\theta(x^i) - y^i\right).x_j^i\right]$$

$\Downarrow$

shrinking the parameter a little bit

② NORMAL EQUATION

$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{2})^T \\ \vdots \\ (x^{m})^T \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix} \in \mathbb{R}^m$$

X ↙ design matrix

$$\theta = (X^T X)^{-1} X^T y$$ ~~Normal.~~

↳ $$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}\right)^{-1} . X^T . y$$ while using regularization

\# Non-invertibility

when $m \leq n$ (↙ training set, → features) } too less data available to curve fit

if $\lambda > 0$

$$\theta = \left(X^T X + \lambda [\quad]\right)^{-1} . X^T y$$

↳ non-invertible / singular / degenerate.

This matrix will be invertible.

REGULARIZED LOGISTIC REGRESSION :

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

$$+\left[\frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2\right] \qquad \theta \in (\theta_0 \theta_1 \dots \theta_n)$$

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

① Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha\left[\frac{1}{m}\sum_{i=1}^{m} (h_\theta(x) - y)^2\right] x_0^i$$

$$\theta_j := \theta_j - \alpha\underbrace{\left[\left(\frac{1}{m}\Sigma(h_\theta(x^i) - y^i)\, x_j^i\right) + \frac{\lambda}{m}\theta_j\right]}_{\frac{\partial J}{\partial \theta}} \quad j = 1 \dots n$$

}

Summary of Linear/Logistic Regression

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i$$

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

$$\frac{\partial J}{\partial \theta} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x) - y)\, x_j^i \quad [\lambda = 0]$$

$$h_\theta(x) = \underbrace{g(\theta^T x)}_{\text{sigmoid}} = \frac{1}{1 + e^{-\theta^T x}}$$

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

For both linear as well as logistic regression

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x) - y)\, x_0^i$$

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

For regularized linear/logistic regression

Note: for complex polynomial regression. (Map feature)

```
for i = 1 : degree.
    for j = 0 : i
    output(:, end+1) = (x1.^(i-j)).* (x2.^(j));
```

example: $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \dots + \theta_{28} x_1^6 . x_2^6$

$\theta = 28 \times 1$ $\quad X = m \times 28$

$x = 28 \times 1$ $\quad y = 28 \times 1$

Special: One-Vs-All regularized expression

$K \in \{1 \dots 10\}$: 10 classifiers.

↓ classes

$\theta \in \mathbb{R}^{K \times (n+1)}$: Each row of θ, will contains parameters for class i

Program

```
function [all_theta] = oneVsAll [X, y, num_labels, lambda]
{
   m= size(X,1);
   n= size(X,2);
   x= [ ones(m,1)  X ];
   all_theta = zeros(num_labes , n+1);
   for  c = 1: num_labels,
        {
          initial_theta = zeros(n+1 , 1)
          options = optimset ('GradObj' , 'on' , 'MaxIter' ,50);
          all_theta(c,:) = ...
                    fmincg (@t)( lrcostfunction ( t, x, (y==c),
                                 lambda )), ...
                                  (initial_theta , options);
        }
}
```

% (y==c) creates a logical array for a class.

$y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$ ← original

$y == 2 \Rightarrow$ transform. $y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

```
function p = predictOneVsAll(all_theta, X)
{
    m = size(X, 1)
    p = zeros(m, 1)
    X = [ones(m, 1) X];

    predict = sigmoid(X * all_theta');
    [v, p] = max(predict, [], 2);
}
```