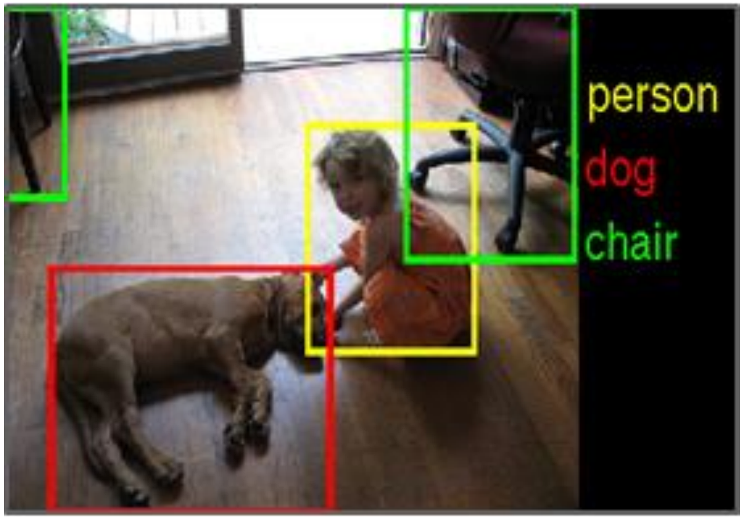
Linear Classifiers

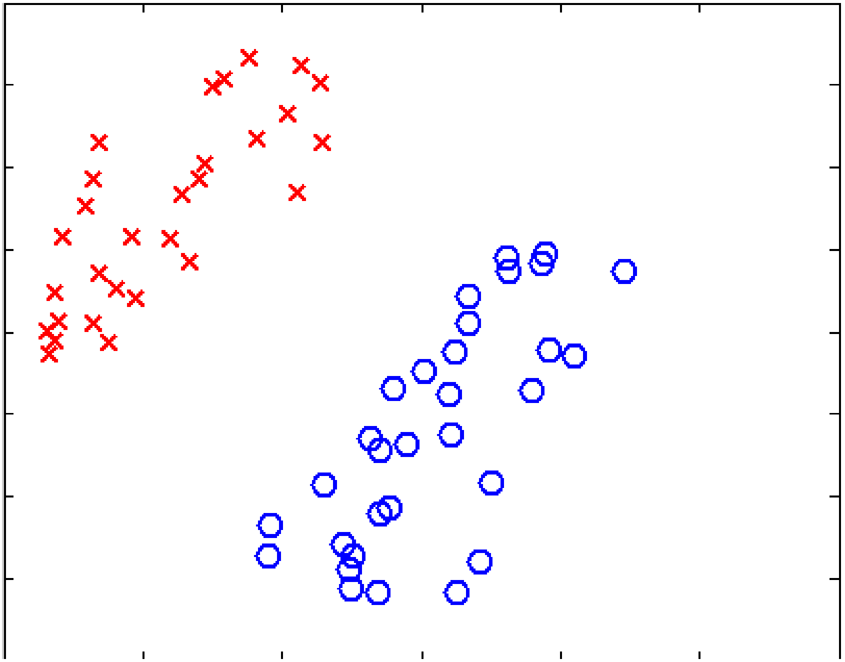
Xiaoming Qin

Apr 27,2016

*Many slides based on Bishop's book, Hauskrecht's slides, Urtasun and Zemel's slides, Andrew Ng Machine Learning class notes

Classification





Classification as Regression

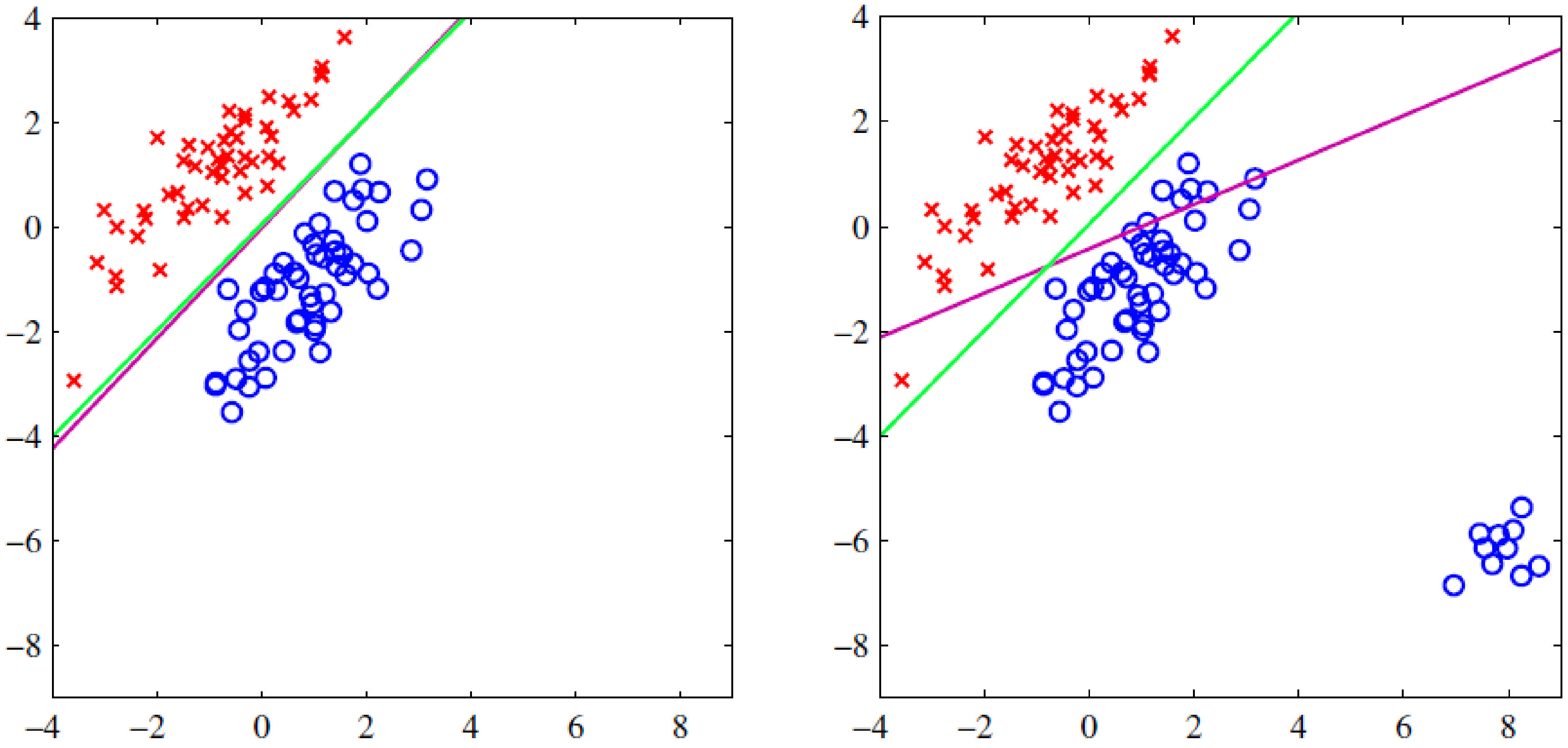
- Can we do this task using what we have learned in the previous lecture?
- Simple hack: Ignore that the input is categorical!
- Suppose we have a binary problem, $t \in \{-1, 1\}$
- Assuming the standard model used for regression

$$y = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

- How can we obtain $\mathbf{w}$?
- Use least squares, $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$. How is $\mathbf{X}$ computed? and $\mathbf{t}$?
- Which loss are we minimizing? Does it make sense?

$$\ell_{square}(\mathbf{w}, t) = \frac{1}{N} \sum_{i=1}^{N} (t_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Problem with least squares



Discriminate Functions

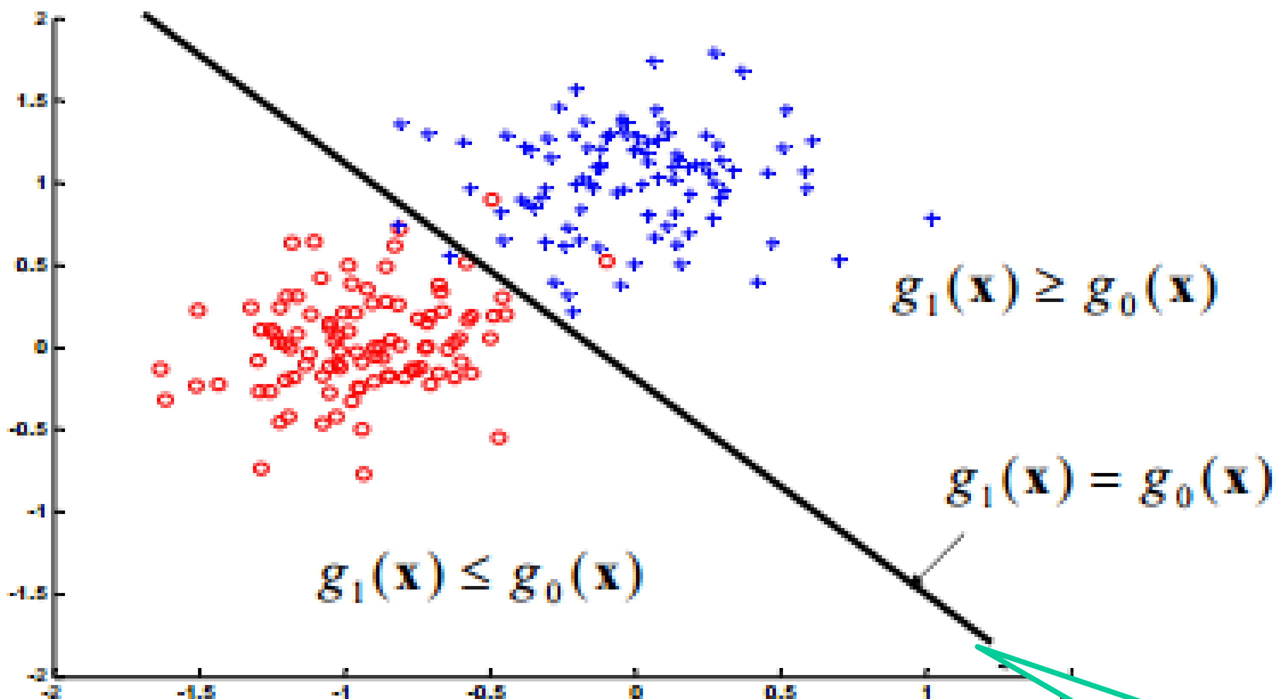
- A common way to represent a **classifier is by using**
  - **Discriminant functions**
- **Works for both the binary and multi-way classification**
- **Idea:**
  - For every class $i = 0, 1, \ldots k$ define a function $g_i(\mathbf{x})$ mapping $X \to \Re$
  - When the decision on input $\mathbf{x}$ should be made choose the class with the highest value of $g_i(\mathbf{x})$

$$y^* = \arg\max_i \; g_i(\mathbf{x})$$

- So what happens with the input space? Assume a binary case.

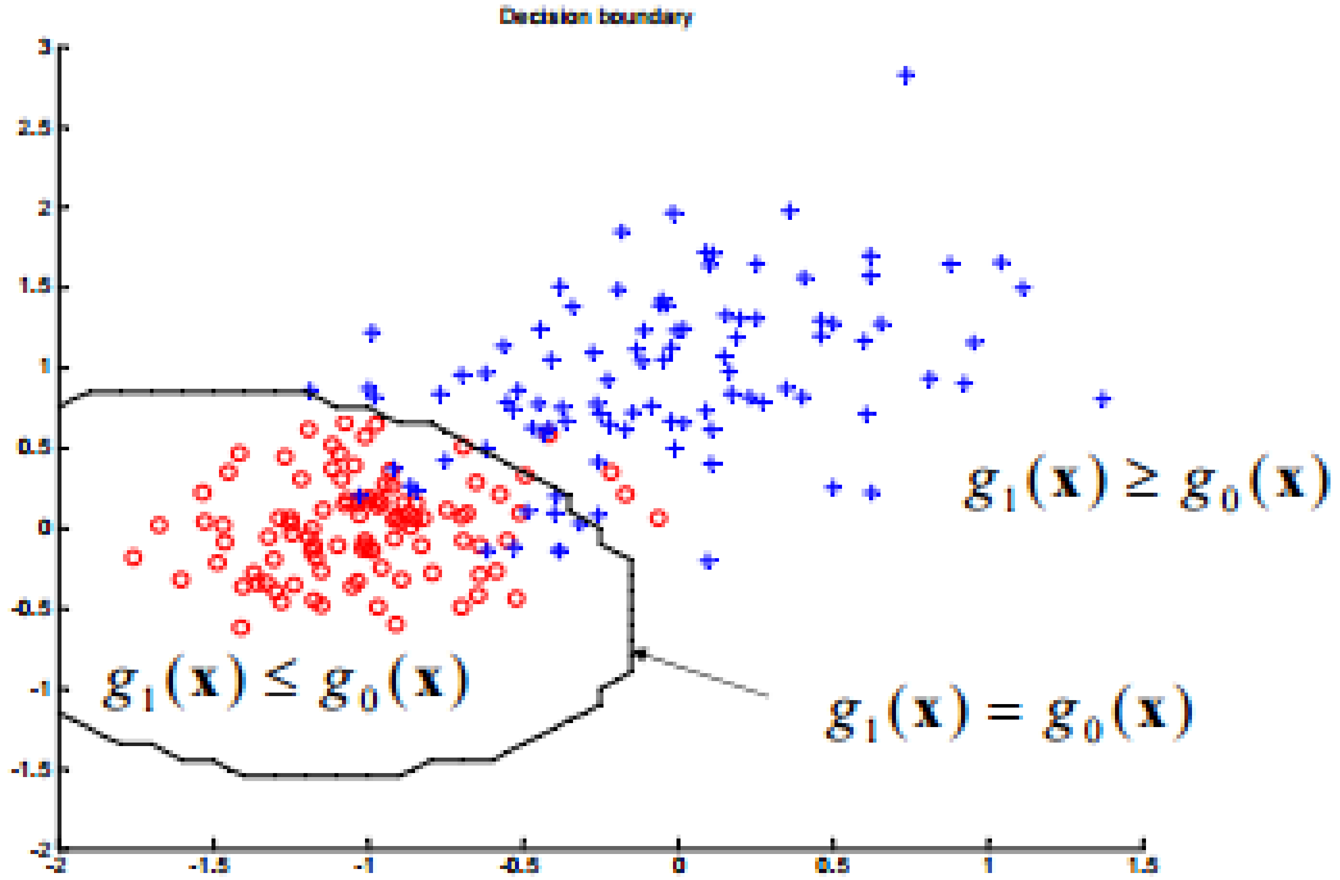
Linear Decision Boundary

- **Decision boundary:** discriminant functions are equal



直线的性质

Quadratic Decision Boundary



Intuitive Change for Classifier

- Our classifier has the form

$$f(\mathbf{x}, \mathbf{w}) = w_o + \mathbf{w}^T \mathbf{x}$$

- A reasonable **decision rule** is

$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}, \mathbf{w}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

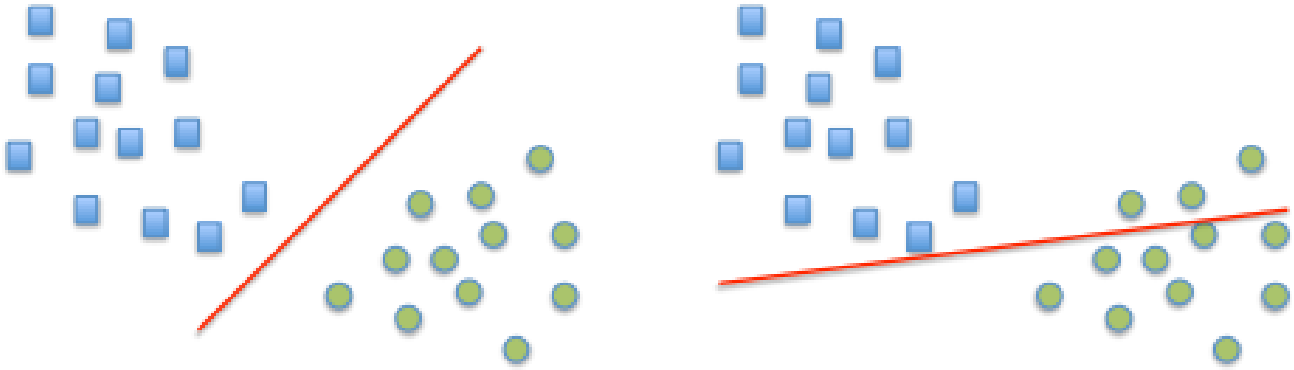
- How can I mathematically write this rule?

$$y = \text{sign}(w_0 + \mathbf{w}^T \mathbf{x})$$

- How does this function look like?

Learning Linear Classifiers

- Learning consists in estimating a "good" decision boundary
- We need to find $\mathbf{w}$ (direction) and w_0 (location) of the boundary
- What does "good" mean?
- Is this boundary good?



- We need a criteria that tell us how to select the parameters

Problem

- The classifier we have looked at is

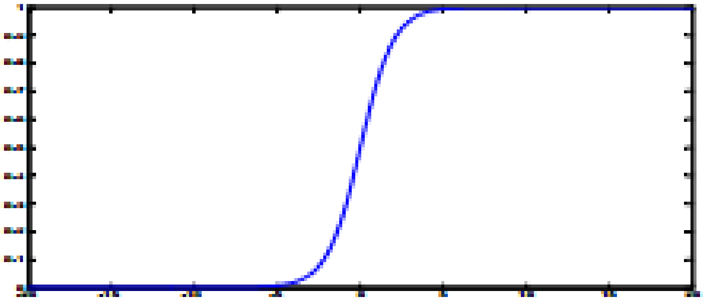
$$y(\mathbf{x}) = \text{sign}(w_0 + \mathbf{w}^T\mathbf{x})$$

- It was difficult to optimize any loss on $\ell(y, t)$ due to the form of $y(\mathbf{x})$
- Can we have a smoother function such that things become easier to optimize?

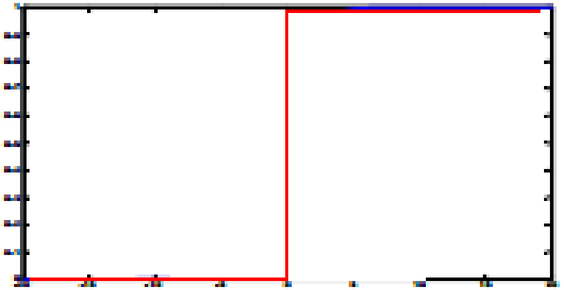
Logistic Function

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

- Is also referred to as a **sigmoid function**
- takes a real number and outputs the number in the interval [0,1]
- Models a smooth switching function; replaces hard threshold function



Logistic (smooth) switching



Threshold (hard) switching

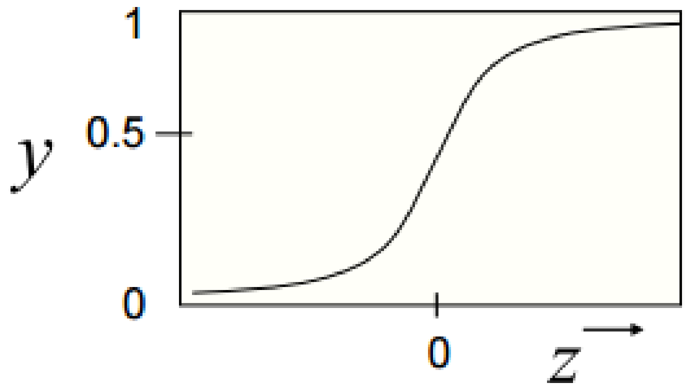
Logistic Regression

- An alternative: replace the *sign*$(\cdot)$ with the **sigmoid** or **logistic function**
- We assumed a particular functional form: sigmoid applied to a linear function of the data

$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

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



- The output is a smooth function of the inputs and the weights

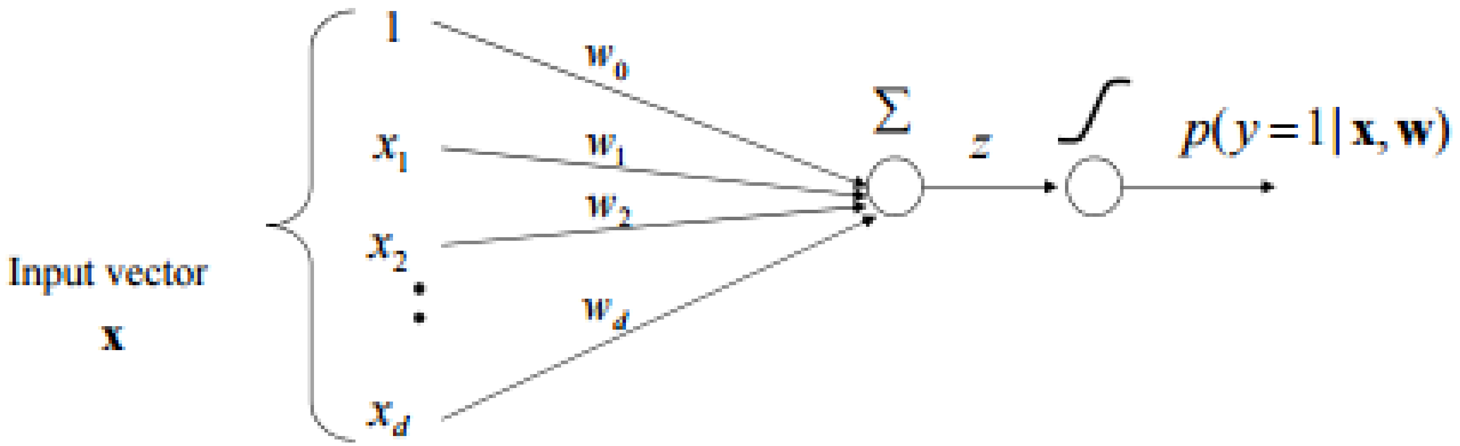
Logistic Regression Model

- **Discriminant functions:**

$$g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x}) \qquad\qquad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x})$$

- **Values of discriminant functions vary in interval [0,1]**
  - **Probabilistic interpretation**

$$f(\mathbf{x}, \mathbf{w}) = p(y = 1 \mid \mathbf{w}, \mathbf{x}) = g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



Logistic Regression Model

- We learn **a probabilistic function**

$$f : X \rightarrow [0,1]$$

  – where f describes the probability of class 1 given $\mathbf{x}$

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T \mathbf{x}) = p(y = 1 \mid \mathbf{x}, \mathbf{w})$$

Note that: $p(y = 0 \mid \mathbf{x}, \mathbf{w}) = 1 - p(y = 1 \mid \mathbf{x}, \mathbf{w})$

- Making decisions with the logistic regression model:

If $p(y = 1 \mid \mathbf{x}) \geq 1/2$ then choose **1**
Else choose **0**

Linear Decision Boundary

- Logistic regression model defines a **linear decision boundary**
- **Why?**
- **Answer:** Compare two **discriminant functions**.
- **Decision boundary:** $g_1(\mathbf{x}) = g_0(\mathbf{x})$
- For the boundary it must hold:

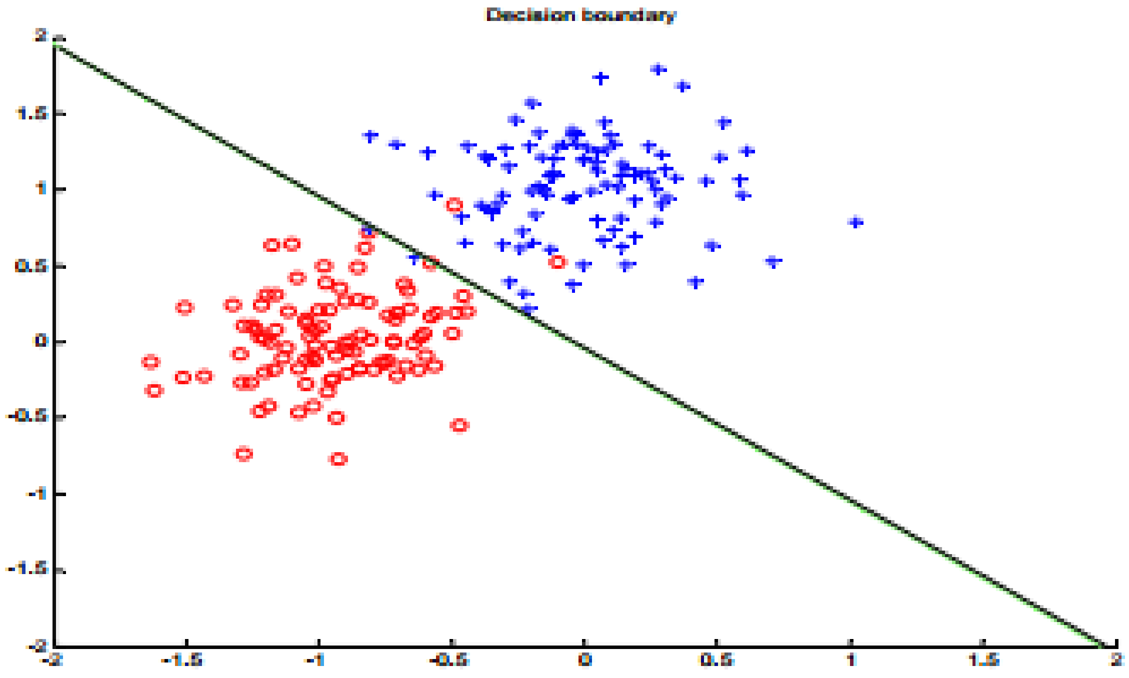
$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^T\mathbf{x})}{g(\mathbf{w}^T\mathbf{x})} = 0$$

$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{\dfrac{\exp-(\mathbf{w}^T\mathbf{x})}{1+\exp-(\mathbf{w}^T\mathbf{x})}}{\dfrac{1}{1+\exp-(\mathbf{w}^T\mathbf{x})}} = \log \exp-(\mathbf{w}^T\mathbf{x}) = \mathbf{w}^T\mathbf{x} = 0$$

Linear Decision Boundary

- **LR defines a linear decision boundary**

Example: 2 classes (blue and red points)



Logistic regression: parameter learning

Likelihood of outputs

- **Let**

$$D_i = < \mathbf{x}_i, y_i > \qquad \mu_i = p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$$

- **Then**

$$L(D, \mathbf{w}) = \prod_{i=1}^{n} P(y = y_i \mid \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1 - y_i}$$

- **Find weights w that maximize the likelihood of outputs**
  - Apply the log-likelihood trick. The optimal weights are the same for both the likelihood and the log-likelihood

$$l(D, \mathbf{w}) = \log \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1 - y_i} = \sum_{i=1}^{n} \log \mu_i^{y_i} (1 - \mu_i)^{1 - y_i} =$$

$$= \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

Good news: $l(W)$ is concave function of W

Bad news: no closed-form solution to maximize $l(W)$

Derivation of the gradient

- **Log likelihood** $l(D,\mathbf{w}) = \sum_{i=1}^{n} y_i \log \mu_i + (1-y_i)\log(1-\mu_i)$

- **Derivatives of the loglikelihood**

$$\frac{\partial}{\partial w_j} l(D,\mathbf{w}) = \sum_{i=1}^{n} \frac{\partial}{\partial z_i}[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)]\frac{\partial z_i}{\partial w_j}$$

$$\frac{\partial z_i}{\partial w_j} = x_{i,j}$$

Derivative of a logistic function

$$\frac{\partial g(z_i)}{\partial z_i} = g(z_i)(1-g(z_i))$$

$$\frac{\partial}{\partial z_i}[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)] = y_i \frac{1}{g(z_i)}\frac{\partial g(z_i)}{\partial z_i} + (1-y_i)\frac{-1}{1-g(z_i)}\frac{\partial g(z_i)}{\partial z_i}$$

$$= y_i(1-g(z_i)) + (1-y_i)(-g(z_i)) = \boxed{y_i - g(z_i)}$$

$$\boxed{\nabla_{\mathbf{w}} l(D,\mathbf{w}) = \sum_{i=1}^{n} \mathbf{x}_i(y_i - g(\mathbf{w}^T\mathbf{x}_i)) = \sum_{i=1}^{n} \mathbf{x}_i(y_i - f(\mathbf{w},\mathbf{x}_i))}$$

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

线性回归的梯度

Derivation of the gradient

- **Notation:** $\mu_i = p(y_i = 1 | \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$
- **Log likelihood**

$$l(D, \mathbf{w}) = \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)$$

- **Derivatives of the loglikelihood**

$$-\frac{\partial}{\partial w_j} l(D, \mathbf{w}) = \sum_{i=1}^{n} -x_{i,j}(y_i - g(z_i))$$

Nonlinear in weights !!

$$\nabla_{\mathbf{w}} -l(D, \mathbf{w}) = \sum_{i=1}^{n} -\mathbf{x}_i(y_i - g(\mathbf{w}^T \mathbf{x}_i)) = \sum_{i=1}^{n} -\mathbf{x}_i(y_i - f(\mathbf{w}, \mathbf{x}_i))$$

- **Gradient descent:**

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha(k) \nabla_{\mathbf{w}} [-l(D, \mathbf{w})] |_{\mathbf{w}^{(k-1)}}$$

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} + \alpha(k) \sum_{i=1}^{n} [y_i - f(\mathbf{w}^{(k-1)}, \mathbf{x}_i)] \mathbf{x}_i$$

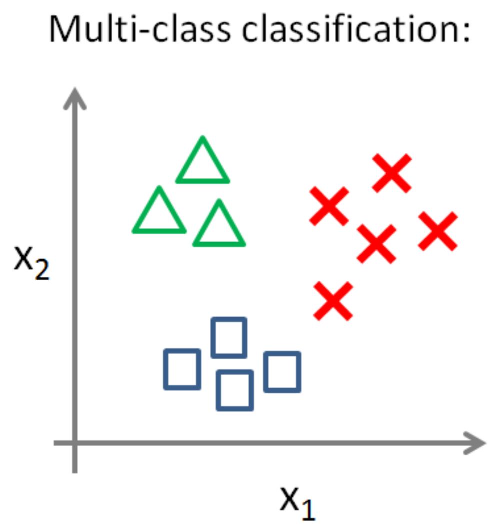
$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

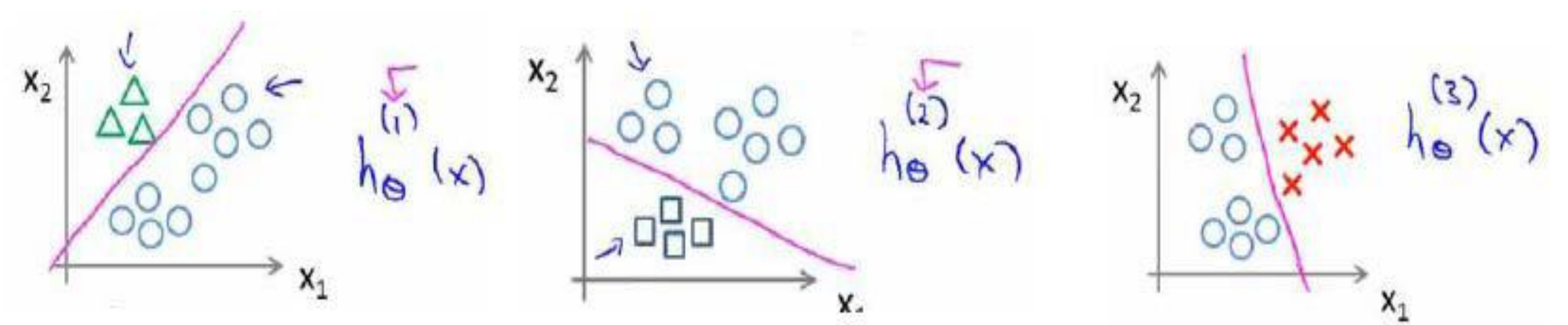
线性回归梯度下降算法

Multiclass Classification

- **Solution 1**: One vs. all

  Need 3 logistic regression classifier





Multiclass Classification

- **Solution 2**: Softmax Regression

Recall for logistic regression:

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\} \qquad y^{(i)} \in \{0, 1\}.$$

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

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

Softmax Regression

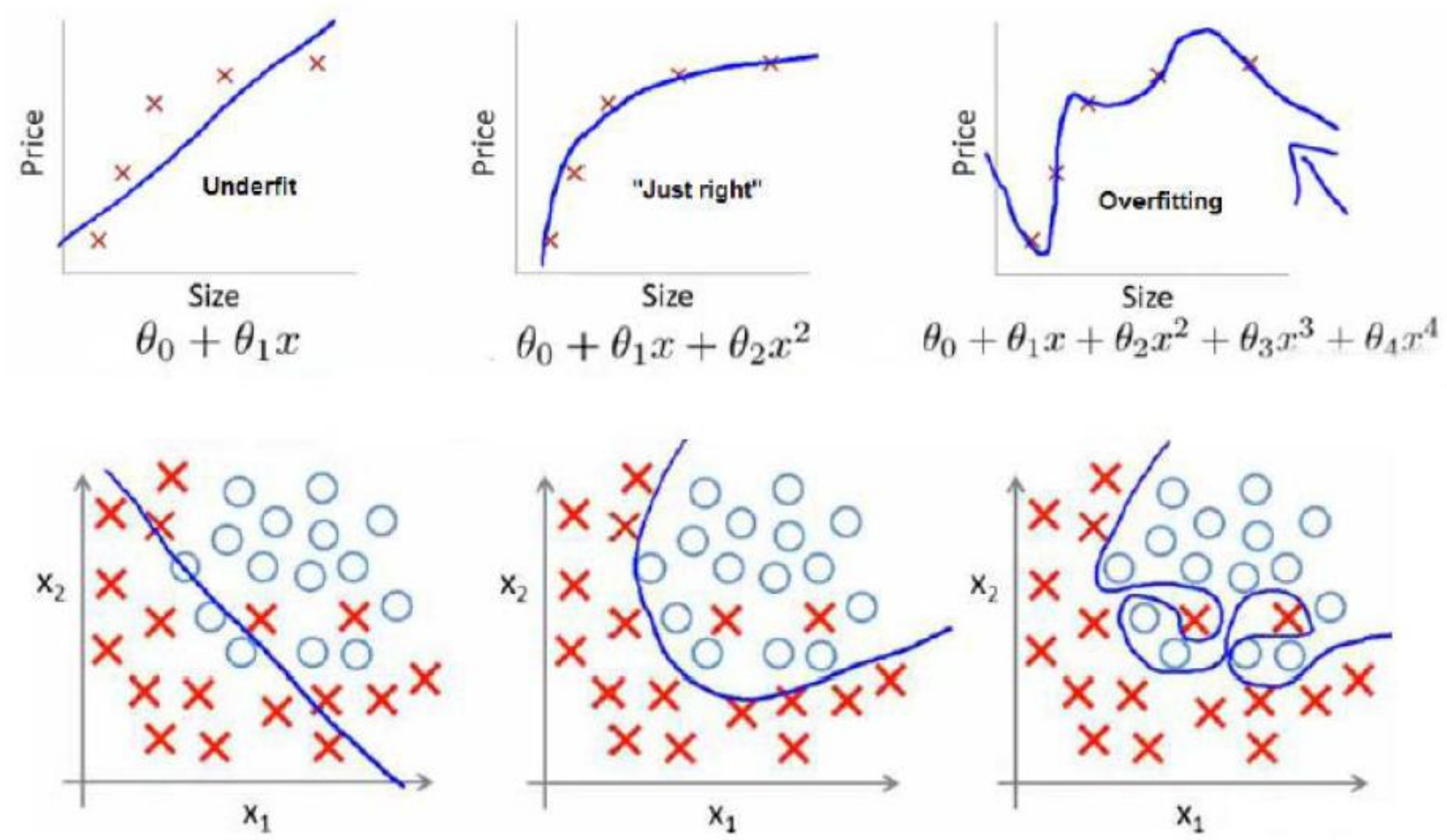
$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\} \qquad y^{(i)} \in \{1, 2, \ldots, k\}$$

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}; \theta) \\ p(y^{(i)} = 2 | x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k | x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^{m} \sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \right]$$

指示函数，内部为真取1，为假取0

Overfitting Problem*



Regularized logistic regression*

λ : Regularization Parameter

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

(1) Lambda = 0
(2) Lambda = ∞

$$Repeat\ \ until\ \ convergence\{$$

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

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

$$for\ \ j=1,2,\ldots n$$

$$\}$$

Task 3

1. Implement Logistic regression on machine learning-ex2
2. Implement one-vs-all logistic regression(10 class) on machine learning-ex3 Section 1
3. Implement Softmax regression (10 class)
4. Run MNIST hand-written digits dataset on Softmax classifier
5. PS: remember to split dataset into training and testing data
6. PS.PS: remember to record your results(such as error rate and picture etc.) and generate PDF

请将源程序和结果的 pdf 版到 github 上，每个人上传一个文件夹到我的 github 文件夹 task_3 下，命名格式：ex2_你的编号 eg：ex2_01

Note: All the assignments and datasets can be found in my github

https://github.com/SimonChin1/ML_Group.git

References

1. http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression
2. Coursera Machine Leaning course by Andrew Ng