Logistic Regression

Lecture 5b

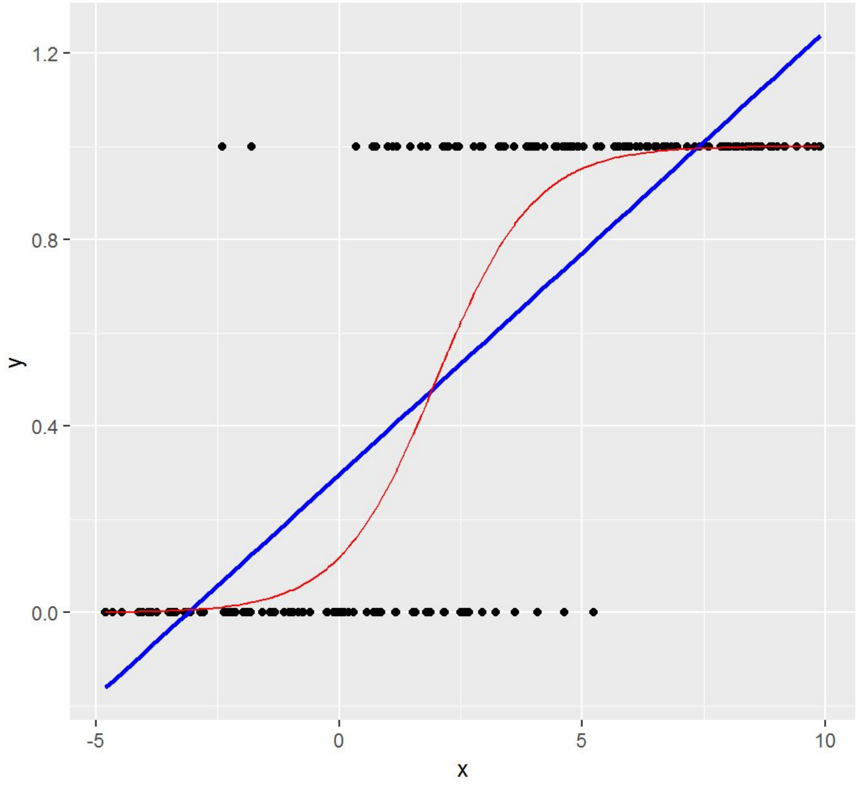
Last time: Classification

- ❑ Select good **performance measures** for classification tasks
- ❑ Know how to pick appropriate **precision/recall tradeoff**
- ❑ Extend to **Multiclass Classification** with One-versus-All or One-versus-One

Today: Learning Objectives

- ❑ Understand **Logistic Regression** in a classification context
- ❑ Formulate the optimization using **gradient descent** to find model parameters
- ❑ Demo on the **IRIS dataset**
- ❑ Explore **Softmax Regression** to handle multiclass

First Look on Logistic Regression



The linear regression (blue line) doesn’t fit the data well, and it produces predicted probabilities below 0 and above 1.

On the other hand, the logistic regression fit (red curve) with its typical “S” shape follows the data closely and always produces predicted probabilities between 0 and 1.

Logistic Regression

- To estimate the **probability** that an sample belongs to a particular class
- Example: The probability of an email being a spam

$$\begin{aligned}\hat{p} &= P(y=1|x;\theta)\\ &= 1 - P(y=0|x;\theta)\end{aligned}$$

- p^ will give us the **probability** that our output is 1 (a spam). For example, p^=0.8 gives us a probability of 80% that our output is 1. Our probability that our prediction is 0 is just the complement of our probability that it is 1 (in this case, the probability that it is 0 is 20%)

Estimating Probabilities

Linear Regression: $\hat{y} = h_\theta(\mathbf{x}) = \theta^\mathrm{T}\mathbf{x}$

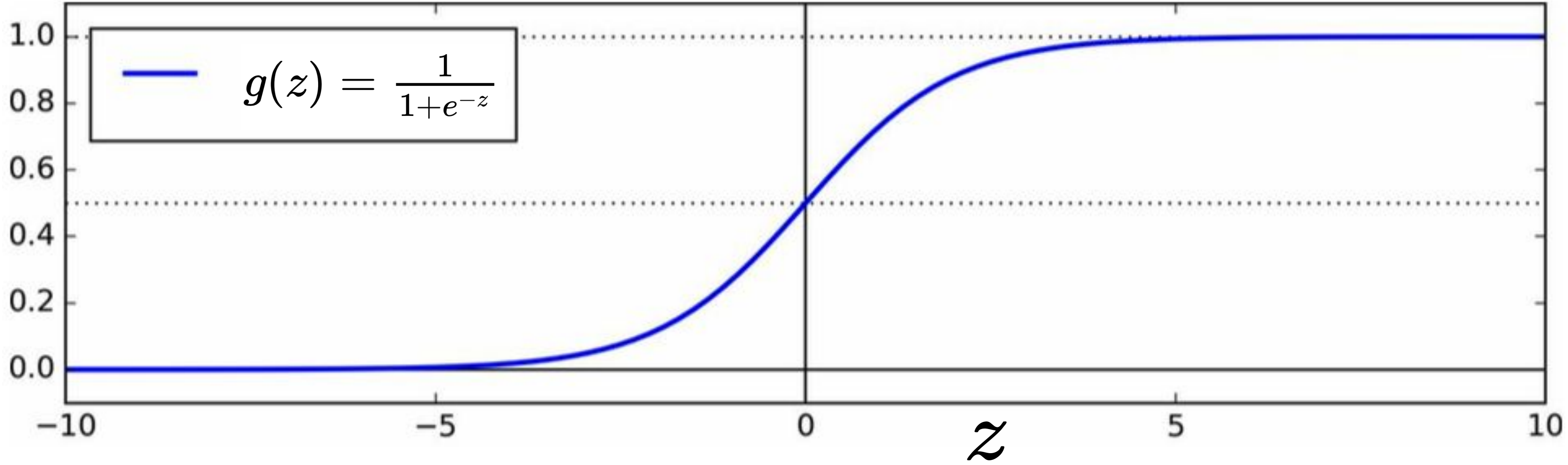
Logistic Regression estimates: $\hat{p} = h_\theta(\mathbf{x}) = g(\theta^\mathrm{T}\mathbf{x})$

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

Probability that output is 1

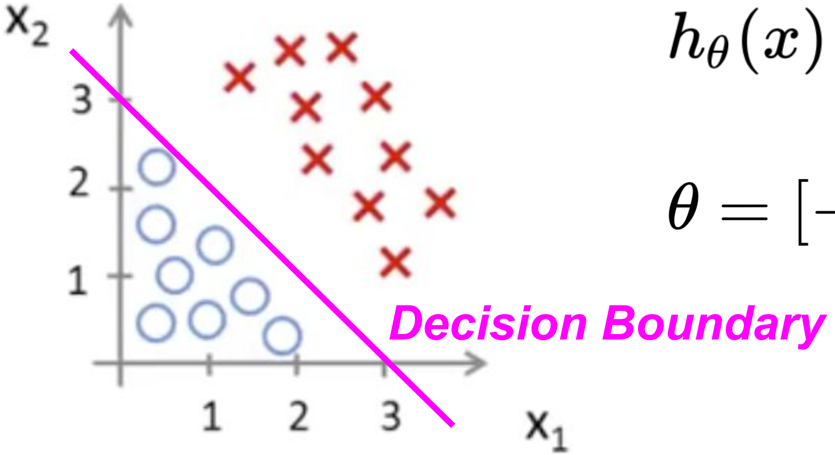
Sigmoid/Logistic Function

Logistic (Sigmoid) Function



$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5, \quad g(z) < 0.5 \rightarrow z < 0 \rightarrow \theta^T \mathbf{x} < 0 \\ 1 & \text{if } \hat{p} \geq 0.5. \quad g(z) \geq 0.5 \rightarrow z \geq 0 \rightarrow \theta^T \mathbf{x} \geq 0 \end{cases}$$

Decision Boundary: Linear Example



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

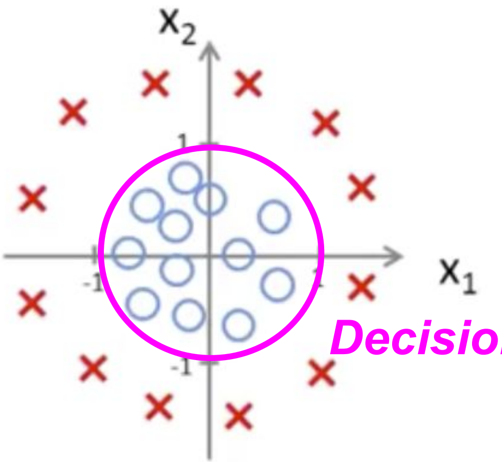
$$\theta = [-3, 1, 1]$$

$$\hat{y} = 1$$
$$\rightarrow \theta^T \mathbf{x} \geq 0$$
$$\rightarrow -3 + x_1 + x2 \geq 0$$
$$\rightarrow x_1 + x_2 \geq 3$$

$$x_1 + x_2 = 3$$

$$\hat{y} = 0$$
$$\rightarrow \theta^T \mathbf{x} < 0$$
$$\rightarrow -3 + x_1 + x_2 < 0$$
$$\rightarrow x_1 + x_2 < 3$$

Decision Boundary: Non-linear Example



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

$$\theta = [-1, 0, 0, 1, 1]$$

$$\hat{y} = 1$$
$$\rightarrow \theta^T \mathbf{x} \geq 0$$
$$\rightarrow -1 + x_1^2 + x_2^2 \geq 0$$
$$\rightarrow x_1^2 + x_2^2 \geq 1$$

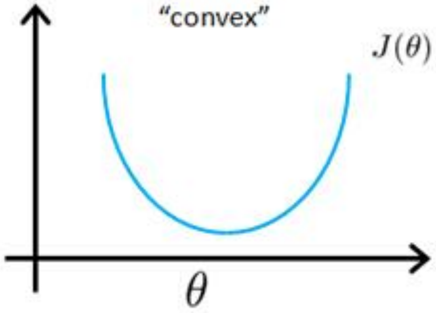
$$x_1^2 + x_2^2 = 1$$

$$\hat{y} = 0$$
$$\rightarrow \theta^T \mathbf{x} < 0$$
$$\rightarrow -1 + x_1^2 + x_2^2 < 0$$
$$\rightarrow x_1^2 + x_2^2 < 1$$

Cost function

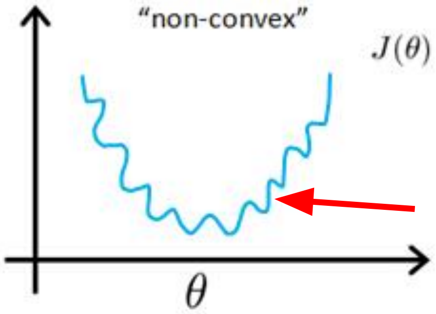
Linear Regression MSE (Square-Loss):

$$\text{Cost}\ (h_\theta(\mathbf{x}), y) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T\mathbf{x}^{(i)} - y^{(i)})^2$$



Will the same cost function work for Logistic Regression

$$\text{Cost}\ (h_\theta(\mathbf{x}), y) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{1+e^{\theta^T\mathbf{x}^{(i)}}} - y^{(i)}\right)^2$$



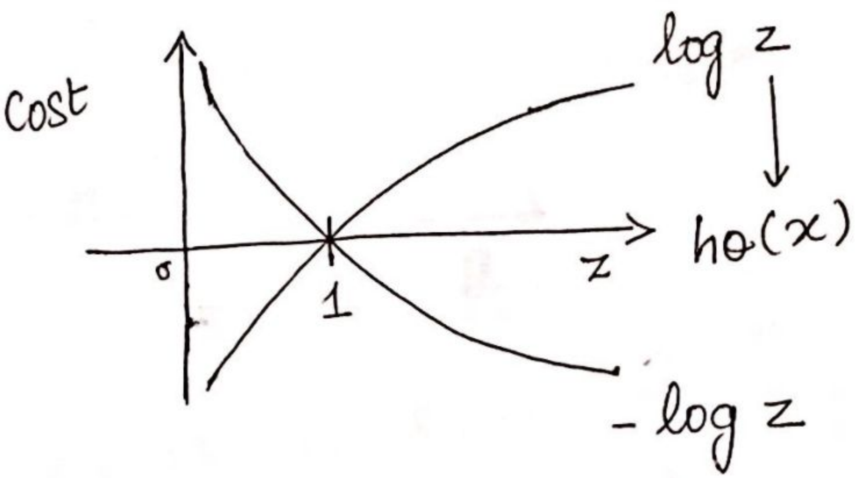
Too many local minima

Need a new cost function → Log-Loss

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

Case 1: When `y = 1`

$$\text{Cost }\ (h_\theta(\mathbf{x}), y) = \ -\log(h_\theta(\mathbf{x}))$$

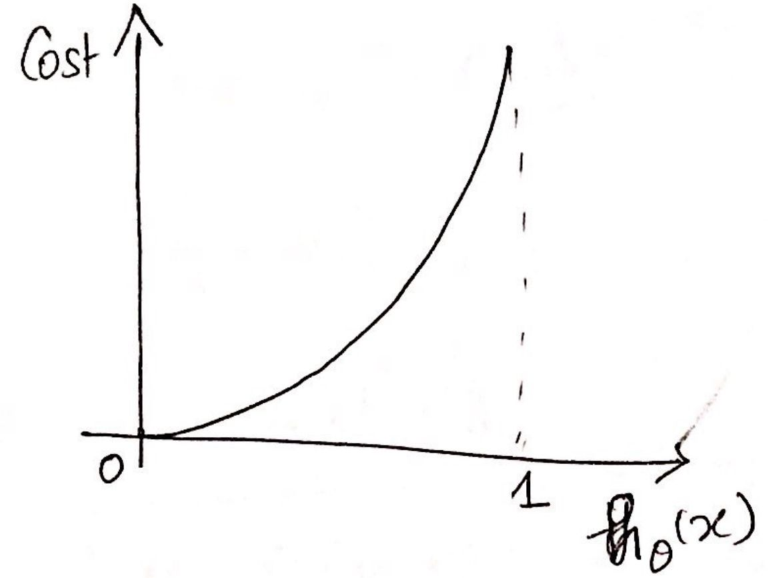


$$\text{Cost }\ (h_\theta(\mathbf{x}), y) = 0 \text{ if } \ h_\theta(\mathbf{x}) = 1$$

$$\text{Cost }\ (h_\theta(\mathbf{x}), y) \to \infty \text{ if } \ h_\theta(\mathbf{x}) \to 0$$

Case 2: When `y = 0`

$$\text{Cost } (h_\theta(\mathbf{x}), y) = -\log(1 - h_\theta(\mathbf{x}))$$



$$\text{Cost } (h_\theta(\mathbf{x}), y) = 0 \text{ if } h_\theta(\mathbf{x}) = 0$$

$$\text{Cost } (h_\theta(\mathbf{x}), y) \to \infty \text{ if } h_\theta(\mathbf{x}) \to 1$$

New Cost Function: Log-Loss

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

Or equivalently (more compact form):

$$\text{Cost}\ \left(h_\theta(\mathbf{x}), y\right) = -y\log(h_\theta(\mathbf{x})) - (1-y)\log(1 - h_\theta(\mathbf{x}))$$

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

*Advanced content on the probability perspective is available on the bonus slides

Good and Bad News of Logistic Regression

Bad news

No known closed-form to compute the value of parameter that minimizes this cost function (no equivalent of the Normal Equation)

Good news

The cost function is **convex**, so Gradient Descent (or any other optimization algorithm) is guaranteed to find the global minimum.

Gradient Descent Formulation

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), (j = 1 \ldots n)$$

Take partial derivative: $$\frac{\partial J}{\partial \theta_j} = \boxed{\frac{\partial J}{\partial h_\theta}} \cdot \boxed{\frac{\partial h_\theta}{\partial z}} \cdot \boxed{\frac{\partial z}{\partial \theta_j}}$$

Let’s start with cost for **<u>one</u>** example (x,y)

$$\begin{aligned}
\frac{\partial J}{\partial h_\theta} &= \frac{\partial}{\partial h_\theta}\left(- y \log(h_\theta) - (1 - y) \log(1 - h_\theta)]\right) \\
&= -y \frac{1}{h_\theta} - (1 - y) \frac{1}{1 - h_\theta} (-1) \\
&= \left(\frac{-y}{h_\theta}\right) + \left(\frac{1-y}{1-h_\theta}\right) \\
&= \frac{(-y + yh_\theta + h_\theta - yh_\theta)}{h_\theta(1-h_\theta)} = \frac{(h_\theta - y)}{h_\theta(1-h_\theta)}
\end{aligned}$$

Gradient Descent Formulation

$$\frac{\partial J}{\partial \theta_j} = \boxed{\frac{\partial J}{\partial h_\theta}} \cdot \boxed{\frac{\partial h_\theta}{\partial z}} \cdot \boxed{\frac{\partial z}{\partial \theta_j}}$$

$$\frac{\partial J}{\partial h_\theta} = \frac{(h_\theta - y)}{h_\theta(1 - h_\theta)}$$

$$\begin{aligned}\frac{\partial h_\theta}{\partial z} &= \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right)\\ &= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right)\\ &= \frac{1}{(1+e^{-z})}\left(1 - \frac{1}{(1+e^{-z})}\right)\\ &= h_\theta(1 - h_\theta)\end{aligned}$$

$$\begin{aligned}\frac{\partial z}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\left(\theta^T \mathbf{x}\right)\\ &= \frac{\partial}{\partial \theta_j}\left(\sum_{j=1}^{n} \theta_j x_j\right)\\ &= x_j\end{aligned}$$

Gradient Descent Formulation

$$\frac{\partial J}{\partial \theta_j} = \boxed{\frac{\partial J}{\partial h_\theta}} \cdot \boxed{\frac{\partial h_\theta}{\partial z}} \cdot \boxed{\frac{\partial z}{\partial \theta_j}}$$

$$= \boxed{\frac{(h_\theta - y)}{h_\theta(1-h_\theta)}} \boxed{h_\theta(1 - h_\theta)} \boxed{x_j}$$

$$= (h_\theta - y)x_j$$

For the average cost of all training examples:

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

This looks identical to linear regression!

Gradient Descent Step

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta) \\ \frac{\partial}{\partial \theta_1} J(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\theta) \end{bmatrix} = \begin{bmatrix} \frac{1}{m} \sum_i \left(h_\theta(\mathbf{x}^{(i)}) - y^{(i)} \right) x_0^{(i)} \\ \frac{1}{m} \sum_i \left(h_\theta(\mathbf{x}^{(i)}) - y^{(i)} \right) x_1^{(i)} \\ \vdots \\ \frac{1}{m} \sum_i \left(h_\theta(\mathbf{x}^{(i)}) - y^{(i)} \right) x_n^{(i)} \end{bmatrix} = \frac{1}{m} \mathbf{X}^T \left(h_\theta(\mathbf{X}) - \mathbf{y} \right)$$

$$\theta = \theta - \frac{\alpha}{m} \mathbf{X}^T \left(\frac{1}{1+e^{-\mathbf{X}\theta}} - \mathbf{y} \right)$$

$\leftarrow$ **use Batch, Mini-batch, or Stochastic Gradient Descent**

The Iris Dataset

- Contains the sepal and petal length and width of 150 iris flowers of 3 different species: Setosa, Versicolor, and Virginica.
- Let's try to build a classifier to detect **Virginica** based only on the **petal width** feature.

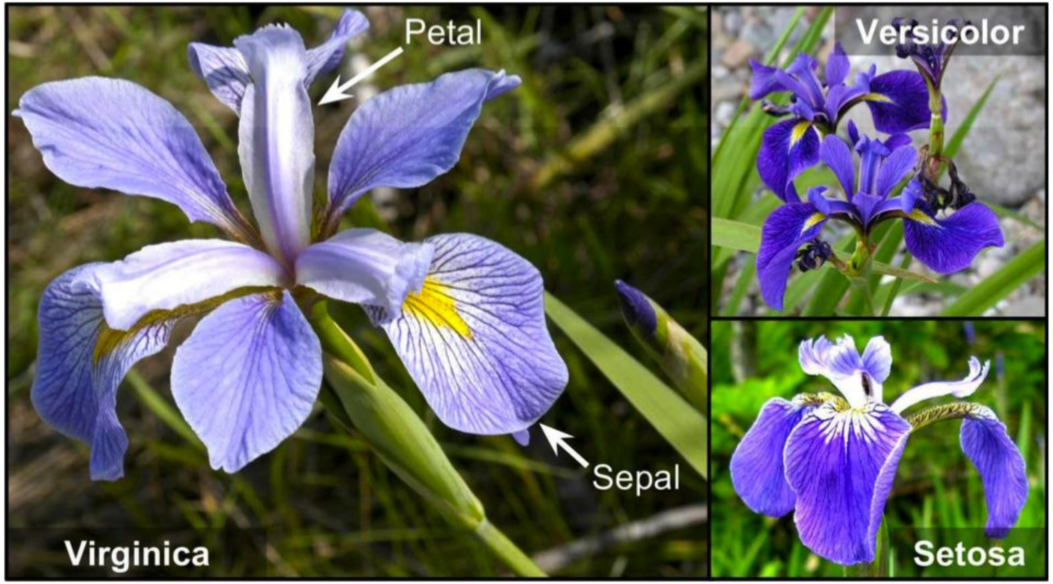
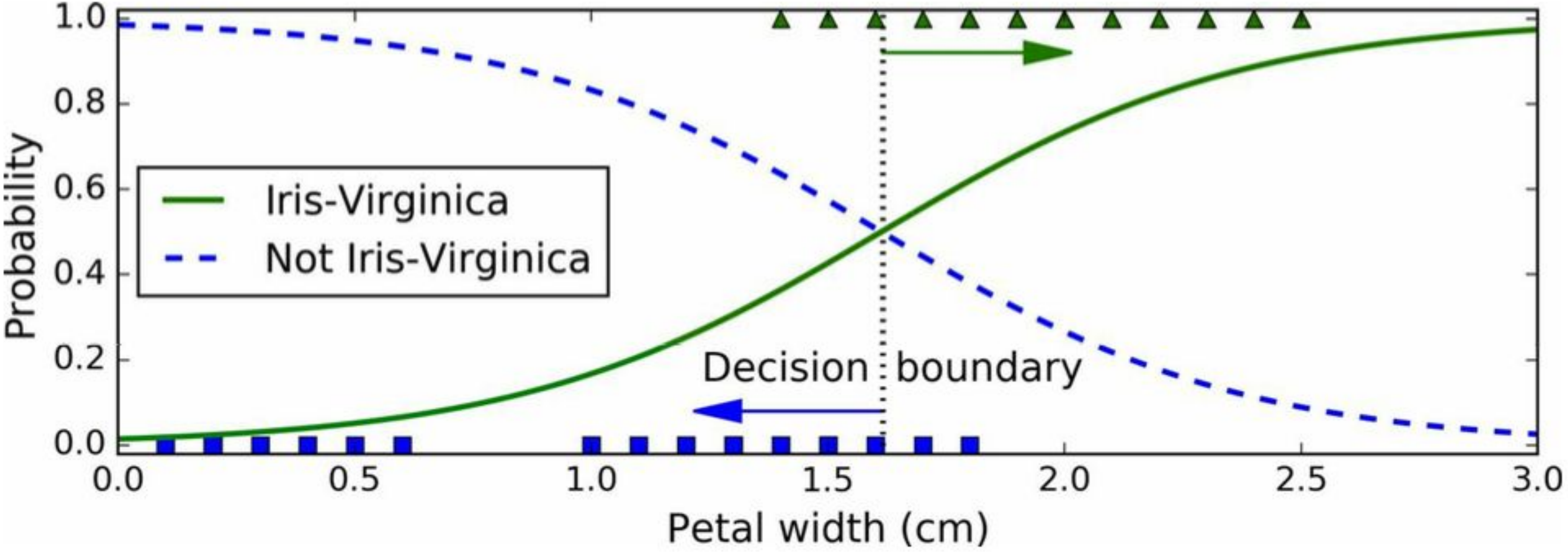
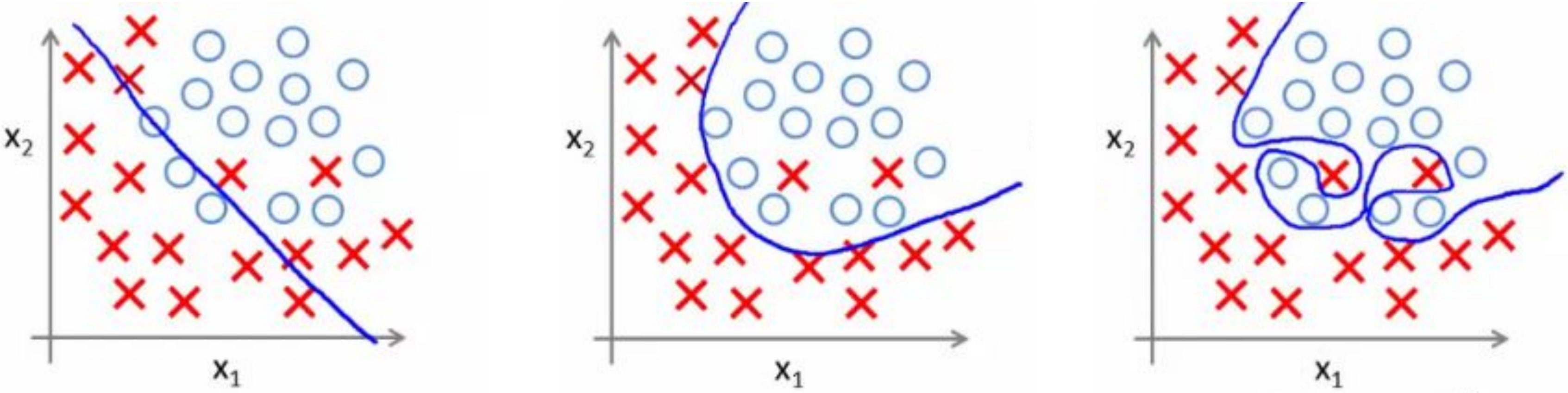


Figure 4-22. Flowers of three iris plant species[16]

Estimated Probabilities and Decision Boundary



Overfitting Problem in Classification



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

(g = sigmoid function)

UNDERFITTING
(high bias)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$
$+\theta_3 x_1^2 + \theta_4 x_2^2$
$+\theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$
$+\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2$
$+\theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots$

OVERFITTING
(high variance)

Ridge Logistic Regression

Hypothesis: $h_\theta(\mathbf{x}) = g(\theta^{\mathrm{T}}\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \dots)$

Cost Function (add L-2 Norm):

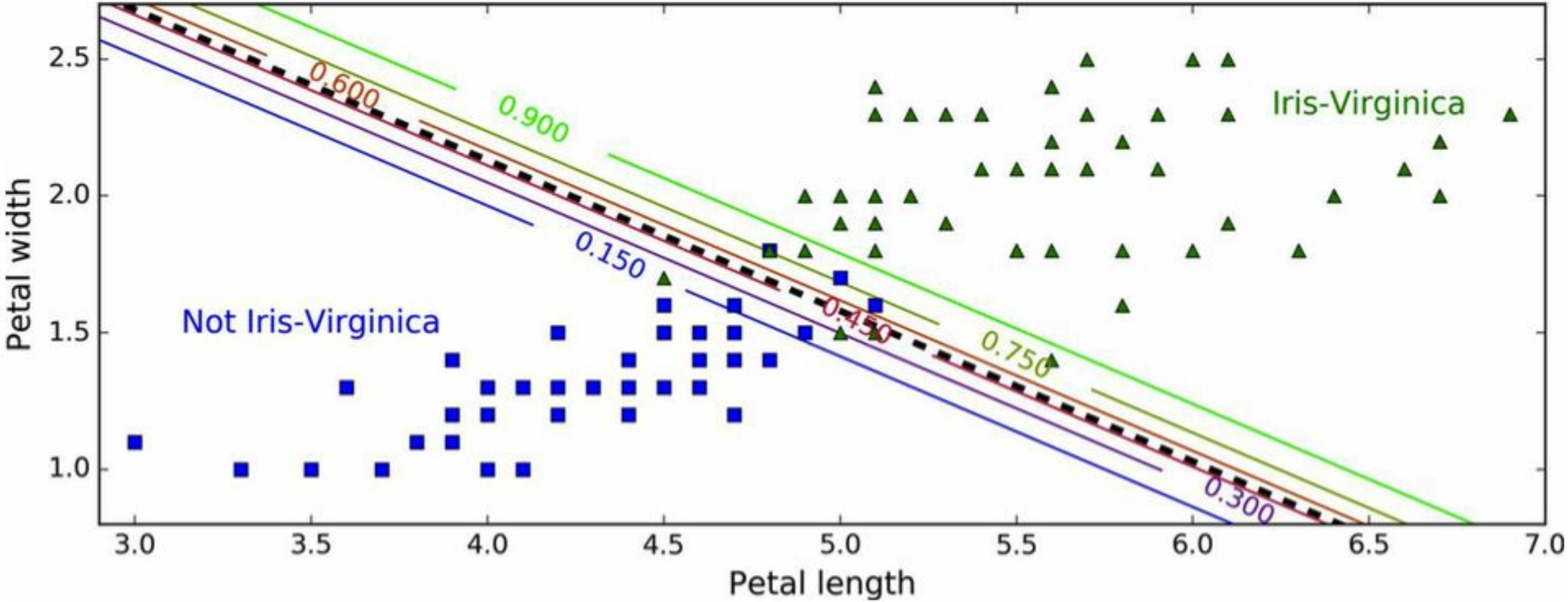
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(\mathbf{x}^{(i)})) + (1-y^{(i)})\log(1-h_\theta(\mathbf{x}^{(i)}))] + \frac{\lambda}{2}\sum_{j=1}^{n}\theta_j^2$$

Gradient Descent (similar to ones of Linear Regression):

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

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

2-D example with probability estimates



Supporting multiple classes

- We need to handle multiple classes (ie. classifying digits)
- Given an input `x`, we want our hypothesis to estimate the probability for each value of **`k = 1,...,K`**

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=k|x;\theta) \end{bmatrix} = \frac{1}{\underbrace{\sum_{j=1}^{K} \exp({\theta^{(j)}}^T x)}_{\text{Normalizing term (sum to 1)}}} \begin{bmatrix} \exp({\theta^{(1)}}^T x) \\ \exp({\theta^{(2)}}^T x) \\ \vdots \\ \exp({\theta^{(k)}}^T x) \end{bmatrix}$$

Softmax Regression

Generalized Logistic Regression for **multiple** classes (Multinomial Logistic Reg.)

1. For sample $\mathbf{x}$, compute a score for each class `k:` $\theta^{(k)^T}\mathbf{x}$

2. Estimate the prob. for each class by applying the softmax function:

$$\hat{p}_k = P(y = k|\mathbf{x};\theta) = \frac{\exp(\theta^{(k)^T}\mathbf{x})}{\sum_{j=1}^{K}\exp(\theta^{(j)^T}\mathbf{x})}$$

3. Pick the highest score:

$$\hat{y} = \arg\max_k P(y = k|\mathbf{x};\theta) = \arg\max_k(\theta^{(k)^T}\mathbf{x})$$

Softmax Cost Function as cross entropy

Logistic Regression Cost Function:

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

$$= -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=0}^{1} y_k^{(i)} \log\left(P(y^{(i)} = k|\mathbf{x}^{(i)};\theta)\right)\right]$$

Generalize for **K** classes:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log P(y^{(i)} = k|\mathbf{x}^{(i)};\theta)\right]$$

$$= \boxed{-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)\right] \quad \leftarrow \textbf{cross entropy}}$$

Demo on the Iris Dataset

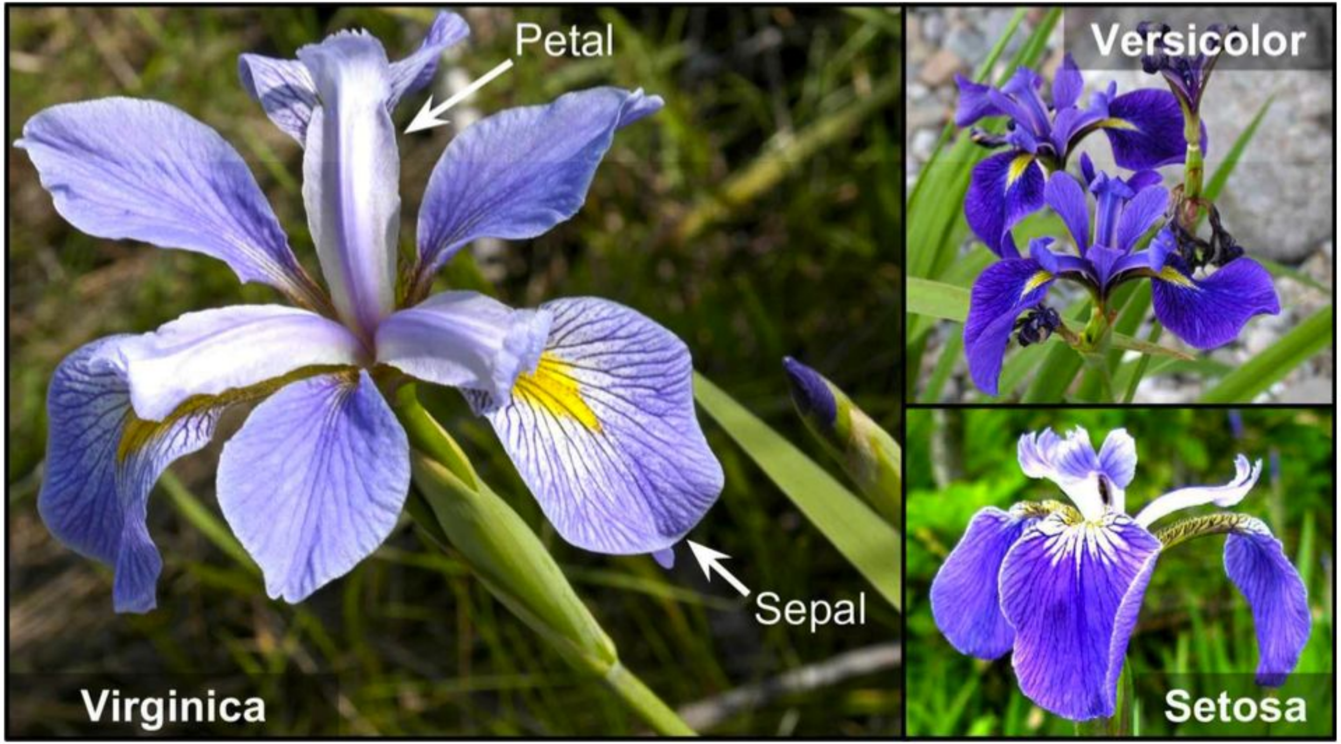
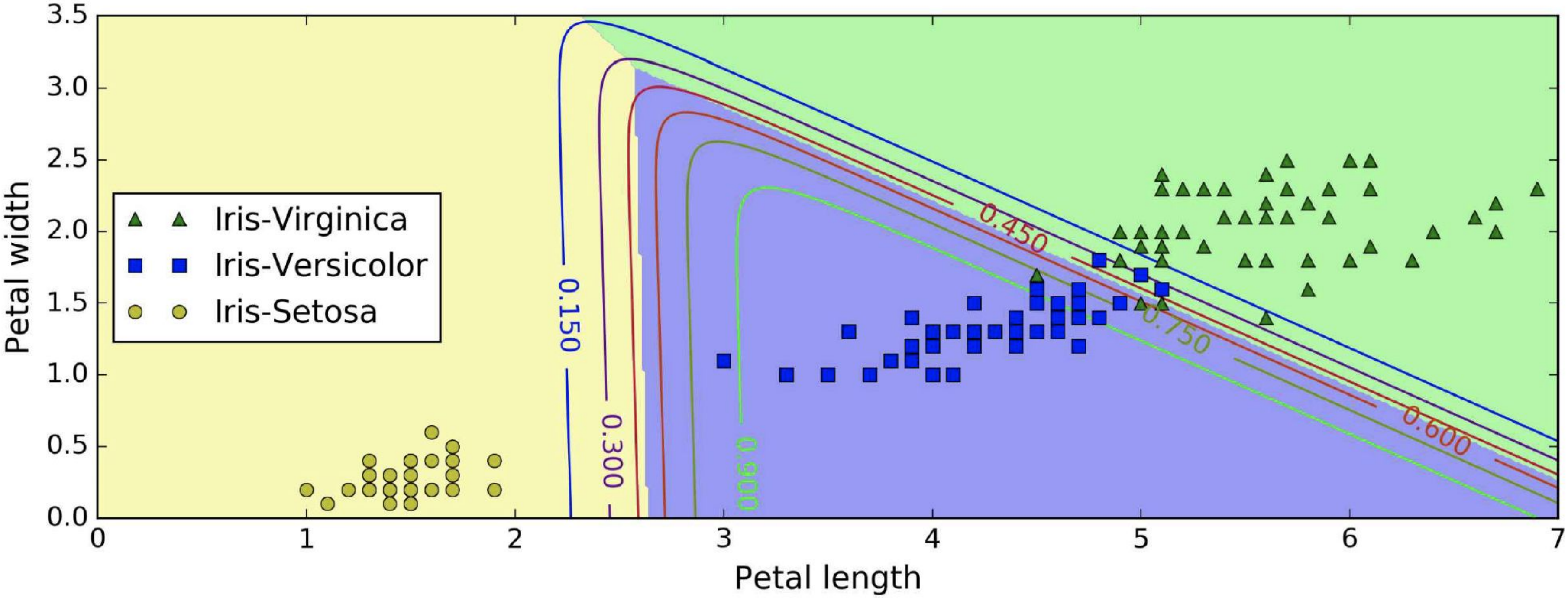


Figure 4-22. Flowers of three iris plant species[16]

Decision Boundaries



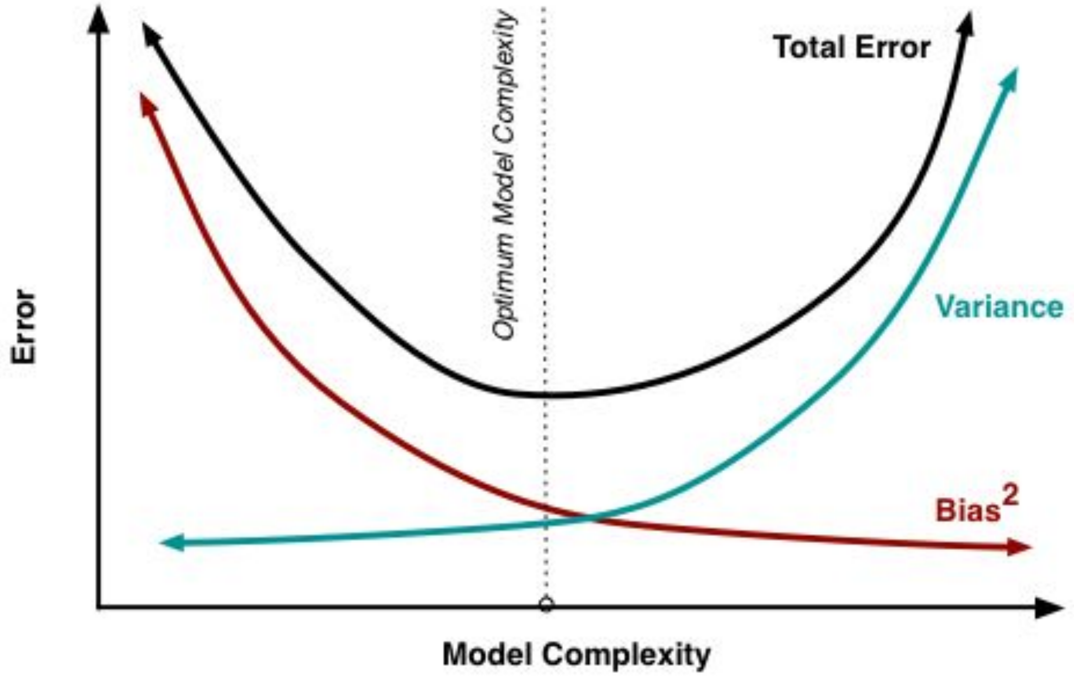
Summary: Learning Objectives

- ❏ Understand **Logistic Regression** in the classification problem
- ❏ Formulate the optimization of **gradient descent** to compute model parameter
- ❏ Demo on the **IRIS dataset**
- ❏ Explore **Softmax Regression** to handle multiclass

Coming up: SUPPORT VECTOR MACHINE!

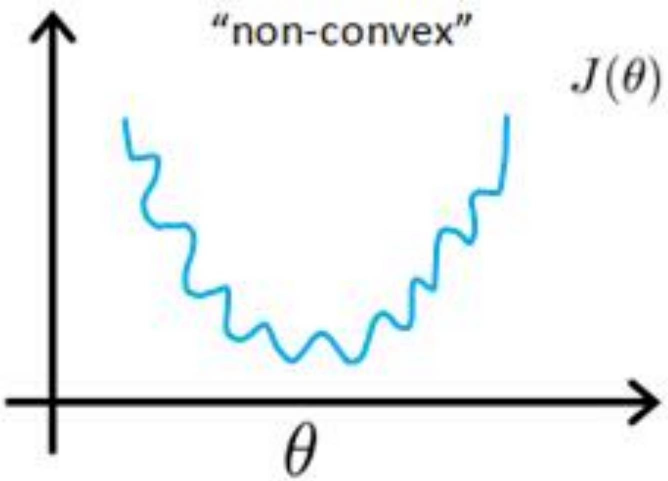
Bonus Materials

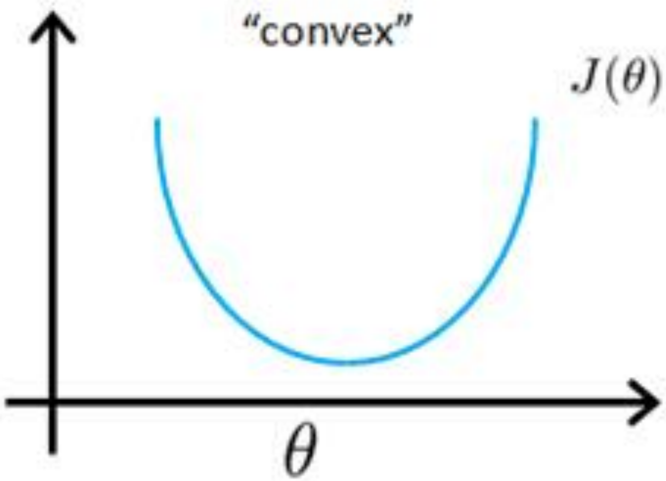
Model Complexity



The Convexity of cost function

- We cannot use the same cost function as linear regression because the Logistic Sigmoid Function will cause the output to be a non-convex function with many local minima.





New Cost function

Linear Regression:

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

$$= \text{Cost}\ \ (h_\theta(\mathbf{x}), y)$$

Logistic Regression:

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

Why two cases? and why using `-log` ?

Probability Perspective of Logistic Regression

By definition:

$$P(y=1|x;\theta) = h_\theta(x)$$
$$P(y=0|x;\theta) = 1 - h_\theta(x)$$

Compactly rewritten:

$$p(y|x;\theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Assume all `m` training examples were generated independently, we can write the likelihood of the params:

$$\begin{aligned} L(\theta) &= p(\mathbf{y}|\mathbf{X};\theta) \\ &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

Probability Perspective (cont)

It will be easier to deal with the log likelihood:

$$\begin{aligned} l(\theta) &= \log L(\theta) \\ &= \log \left(\prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \right) \\ &= \sum_{i=1}^{m} y^{(i)} (\log h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \end{aligned}$$

Cost function as the negative log likelihood:

$$\begin{aligned} J(\theta) &= -\log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} (-\log h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \end{aligned}$$