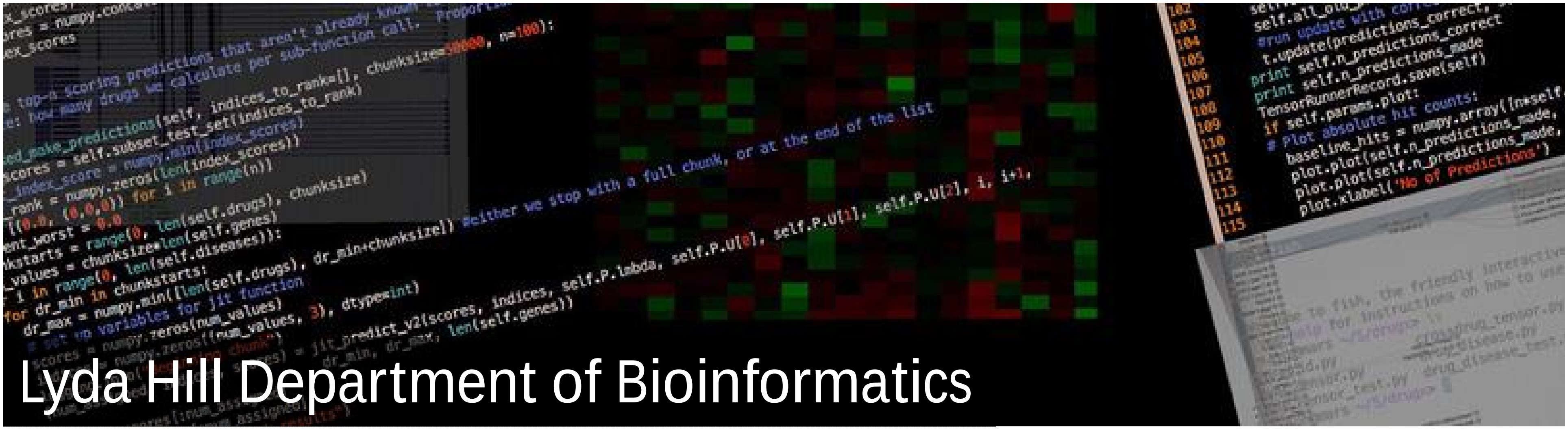


Lyda Hill Department of Bioinformatics

Introduction to machine learning

Albert Montillo, Ph.D.

Albert.Montillo@UTSouthwestern.edu

Deep Learning for Precision Health Lab
Web: utsouthwestern.edu/labs/montillo

Special thanks

§ This talk consists of original slides and slides I’ve adapted from several sources. Special thanks to slides with content curtesy of those listed below.

– Dr Andrew Ng, CS Dept, Stanford University

– Dr Hugo Larochelle, Université de Sherbrooke

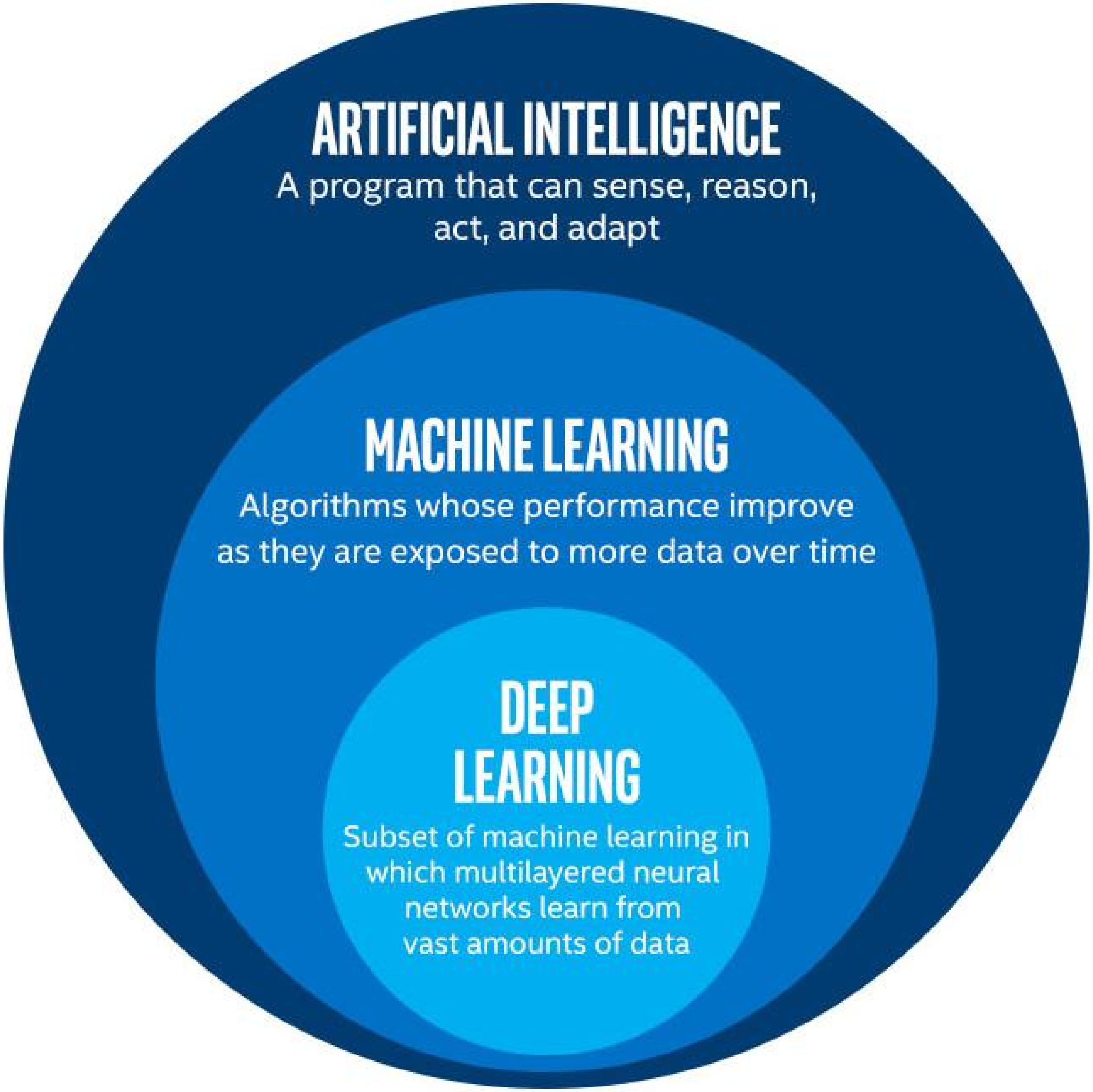
– Dr Aarti Singh, CMU

– Adam Harley, Ryerson University

– Lily Peng, MD/PhD, Google

– Brett Kuprel, EE dept, Stanford

What is machine and deep learning? Where do they fit in?



ARTIFICIAL INTELLIGENCE

A program that can sense, reason, act, and adapt

MACHINE LEARNING

Algorithms whose performance improve as they are exposed to more data over time

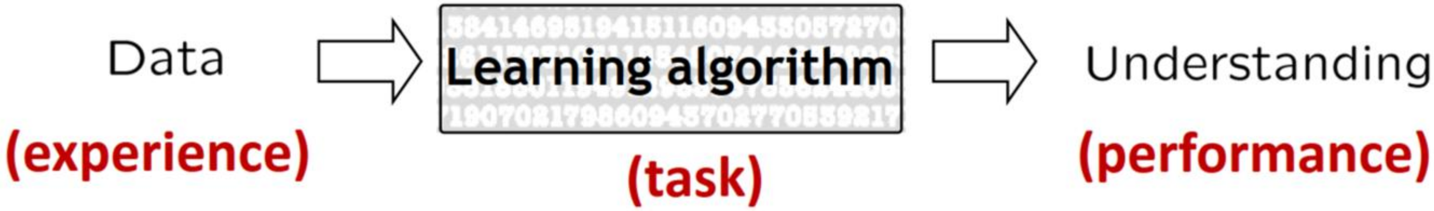
DEEP LEARNING

Subset of machine learning in which multilayered neural networks learn from vast amounts of data

The focus of this course is machine learning

We will focus on the study of algorithms whose

- performance improves
- at some task
- with increasing experience



Data

(experience)

Learning algorithm

(task)

Understanding

(performance)

Broad categories of machine learning tasks

Supervised learning

- Classification
- Regression

Unsupervised learning

- Density estimation
- Clustering
- Dimensionality reduction

Semi-supervised learning

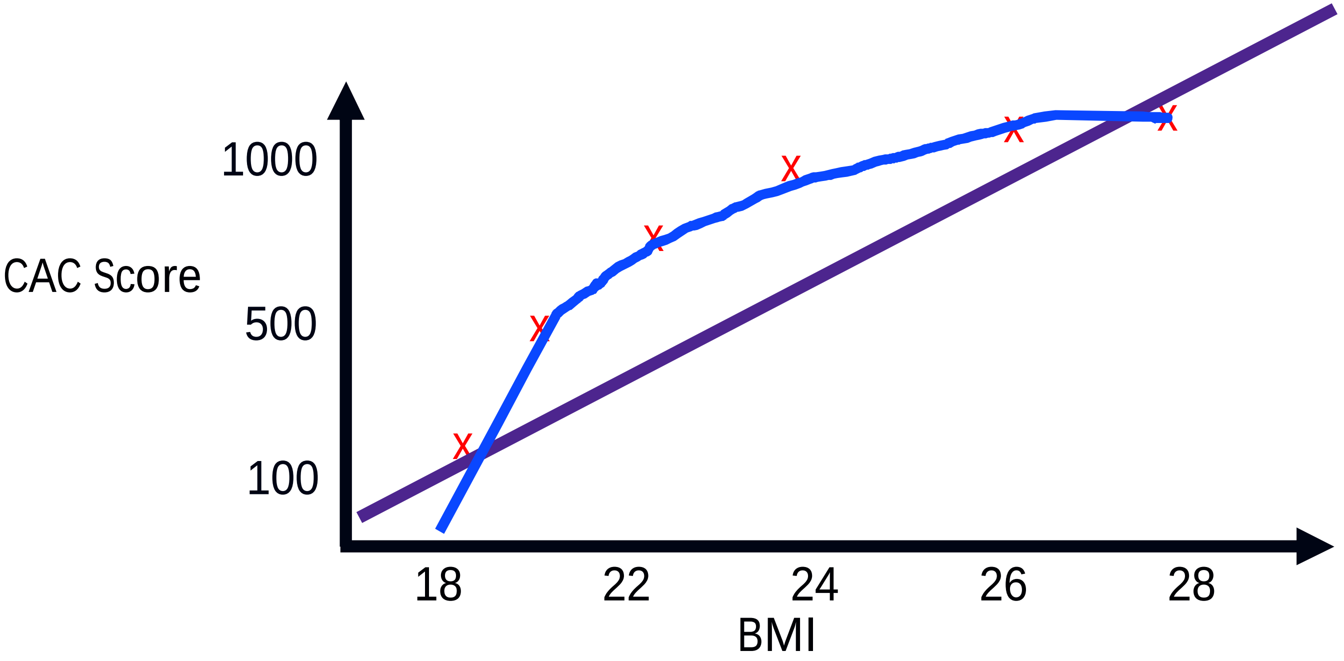
Active Learning

Reinforcement learning

And many others ….

Examples: supervised learning: regression

Predict a patients coronary arteries calcium score (CAC) given their BMI



Supervised learning

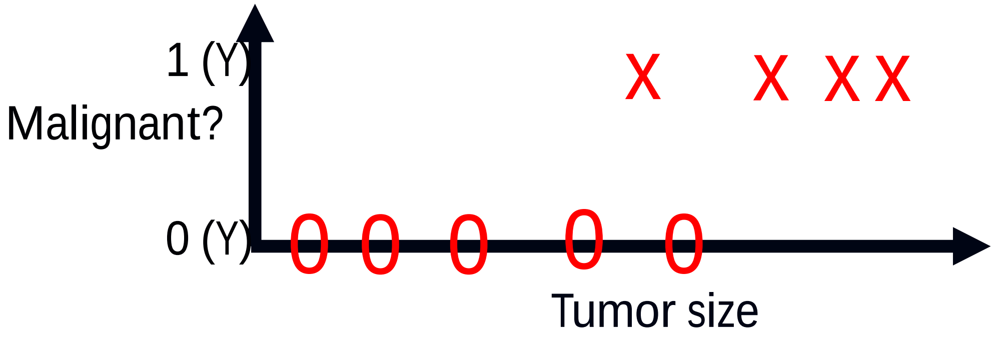
- **Correct answers are provided**

Regression: predict a continuous valued output

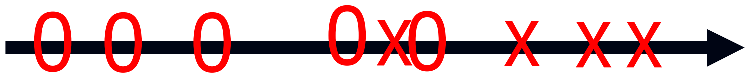
Regression model maps sample's observed feature vector, *x*, to a continuous values output (*y*): $\mathrm{y} = \mathrm{h}(x; \theta)$ **where θ are model parameters.**

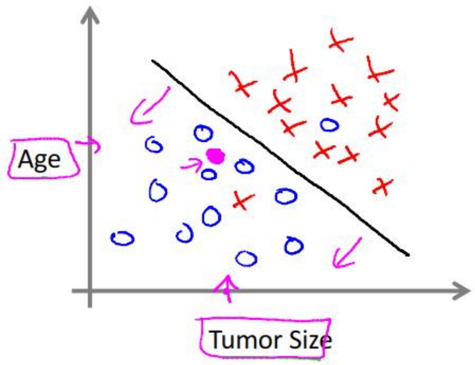
Examples: supervised learning: classification

- Predict whether a brain tumor is benign or malignant based on its volume in cm^3



Many features can be used: Size, age, uniformity of cell size, shape





Supervised learning

- Correct answers are provided

Classification: predict a discrete valued output (binary, nominal – ordered discrete, categorical – non-orderd)

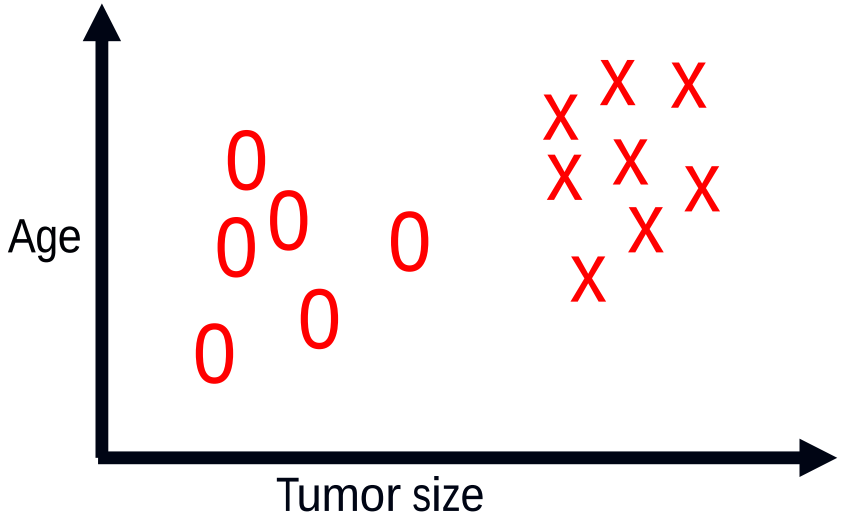
Classification model maps sample's observed feature vector, *x*, to a discrete valued output (*y*): $y = h(x;\theta)$ where y={0,1} two-cat. Classification (e.g. benign vs malignant), or {1,2,3,4} to predict one of 4 disease stages.

Which of the following would you treat as a classification problem?

1. Identify all the places your face appears in a set of photos
2. Reduce email clutter, by determining which email is spam?
3. You would like to predict the event free survival time of a cancer patient in months
4. Diagnose a breast tumor as benign or malignant
5. Predict whether a genetic mutation is present in a patient from one of their in vivo MRIs.
6. Predict the level of atrophy a AD patient will have in 18mo on a given candidate therapy

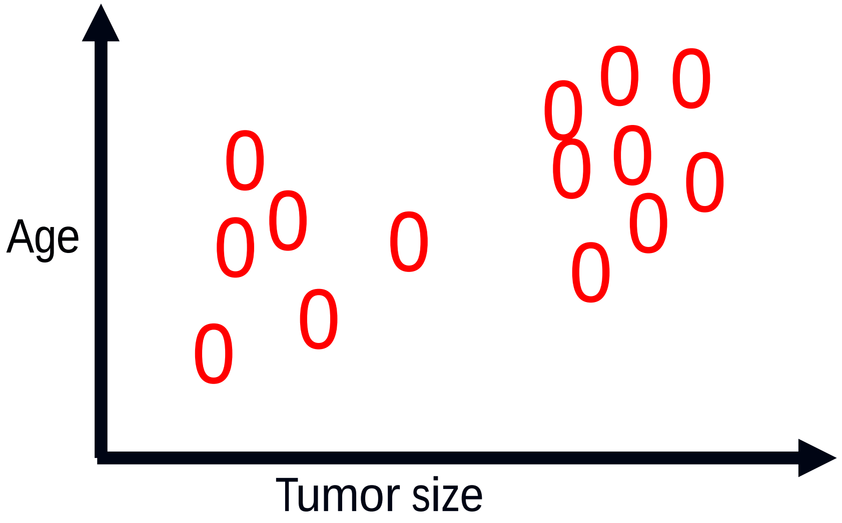
Examples: unsupervised learning: clustering

- **Determine if there are different groups of tumors present and their characteristics.**



Unsupervised learning

- **Correct answers are NOT provided**



Clustering: groups similar samples together. Can represent each group by a set of measures (group centroid). E.g. Can be used to identify cohorts that may respond differently to different therapies

Clustering: defines a mapping from a observed feature vector, x, to a cluster number , y. $\mathbf{y} = \mathbf{h}(\boldsymbol{x};\boldsymbol{\theta})$ **where** $\boldsymbol{y} \in \{\mathbf{1},\mathbf{2}\}$ **if there are two clusters.**

Which of the following would you address with unsupervised learning?

1. Given a dataset of autistic patients, group the patients into distinct groups that have similar clinical and behavioral measures

2. Given a dataset with 1,000 RNA-seq values per subject, reduce the number of features/subject down to a more manageable number of just 2 features for visualization of the distribution of high and low performers on a cognitive exam.

3. Given a dataset of images with each voxel labeled as tumor, edema and healthy tissue, learn to label the voxels on images not part of the dataset.

4. Determine the probability of a subject with a specific ethnicity being diagnosed with Alzheimer's at each age from 1-100

Why we use the programming language, python?

- Developer time is one of the most valuable resources.
    - Salaries in a co, lab are one of the greatest expenses in research

- Python helps you be more productive
    - Python is a higher level, interpreted language
        - Does not require compilation to binary code which delays testing of code
            - Contrast to C, C++, Fortran
    - Write fewer lines of code than C++ or Java.

- Have access to the widest array of state of the art machine learning libraries (more so than Matlab).

- Business friendly license (free)
    - Contrast to Matlab

Supervised learning: Linear regression

- **Given**
  - **A set *D* of *m* labeled training data samples:**

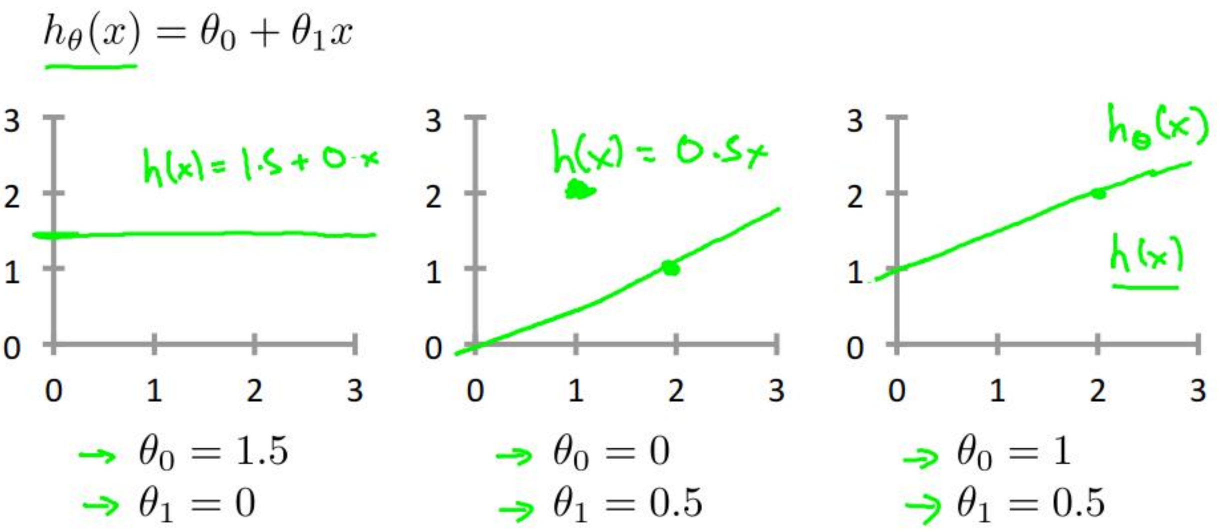
$$D = \{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

where $x^{(i)}$ is the ith training example with label $y^{(i)}$

  - **A linear regression model mapping *x*'s to *y*'s:**

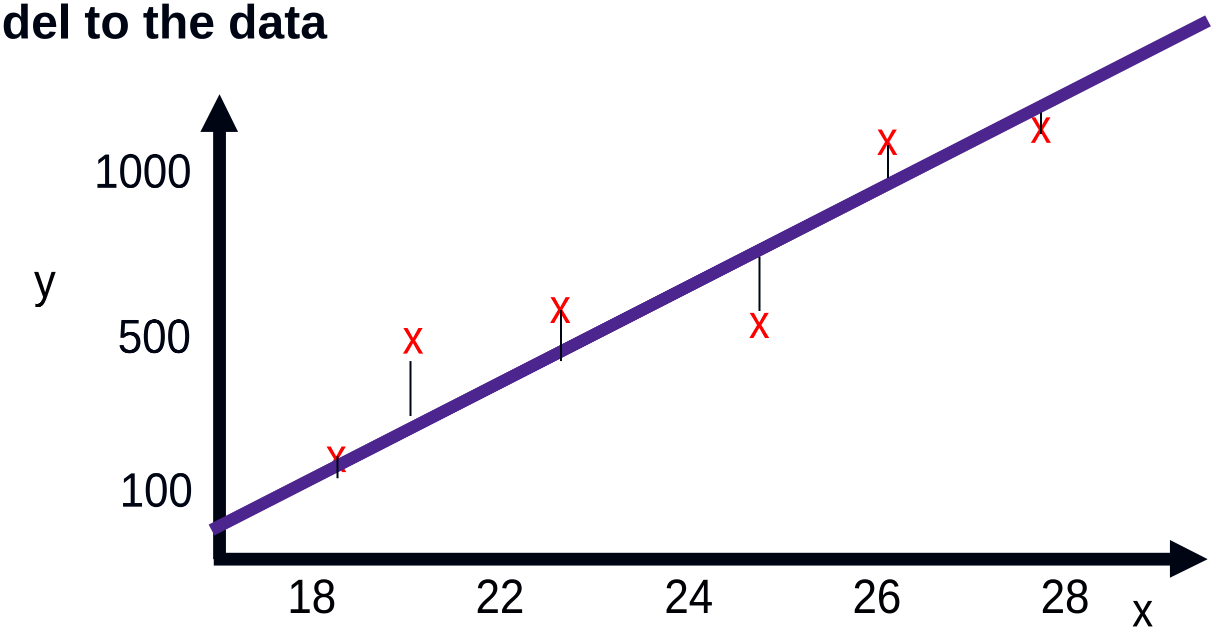
$$\mathbf{y} = \mathbf{h}(x; \theta) = \theta_0 + \theta_1 x$$

- **Find the θ_0, θ_1 that have minimal error on the training examples**



Supervised learning: Linear regression

- Any given choice of θ_0, θ_1 will result in some candidate fit of the model to the data



- Error in model fit can be computed quantitatively with the squared error loss function

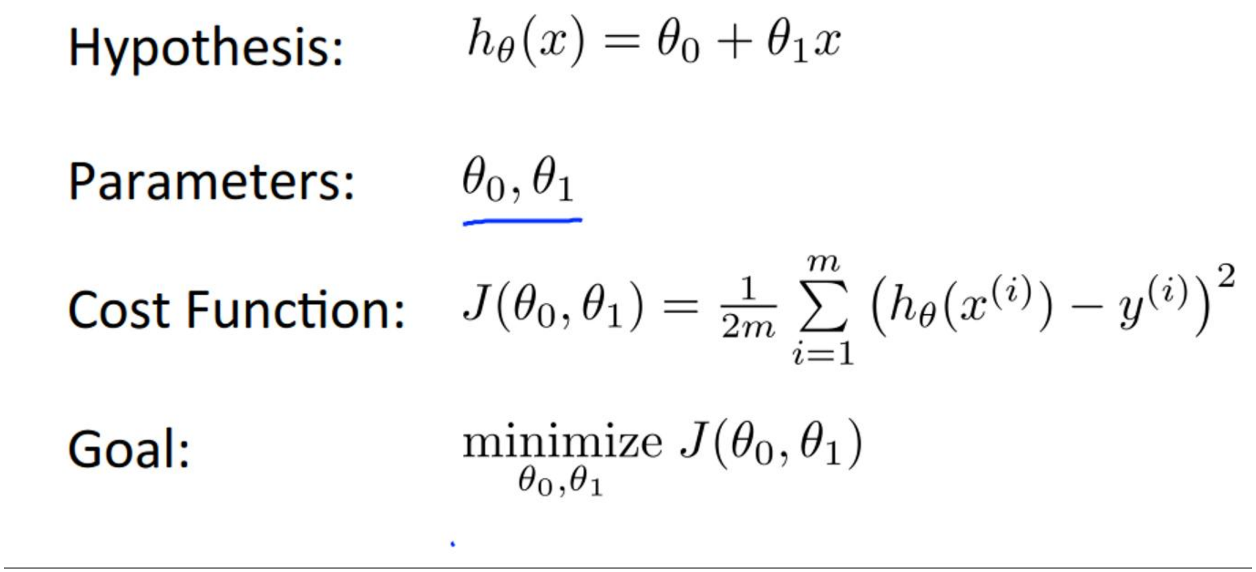
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Goal:

$$\underset{\theta_0, \theta_1}{\text{minimize}}\, J(\theta_0, \theta_1)$$

Supervised learning: Linear regression

- **Review notation**



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

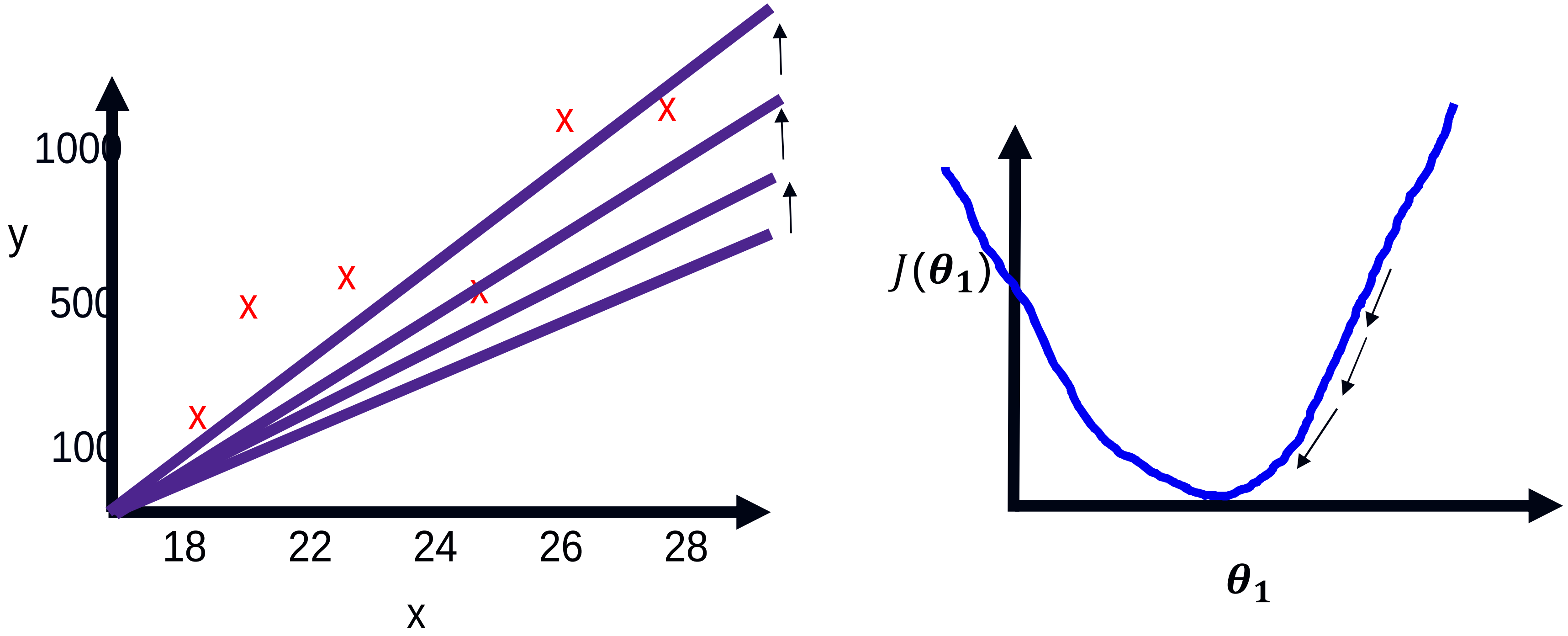
Parameters: $\underline{\theta_0, \theta_1}$

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

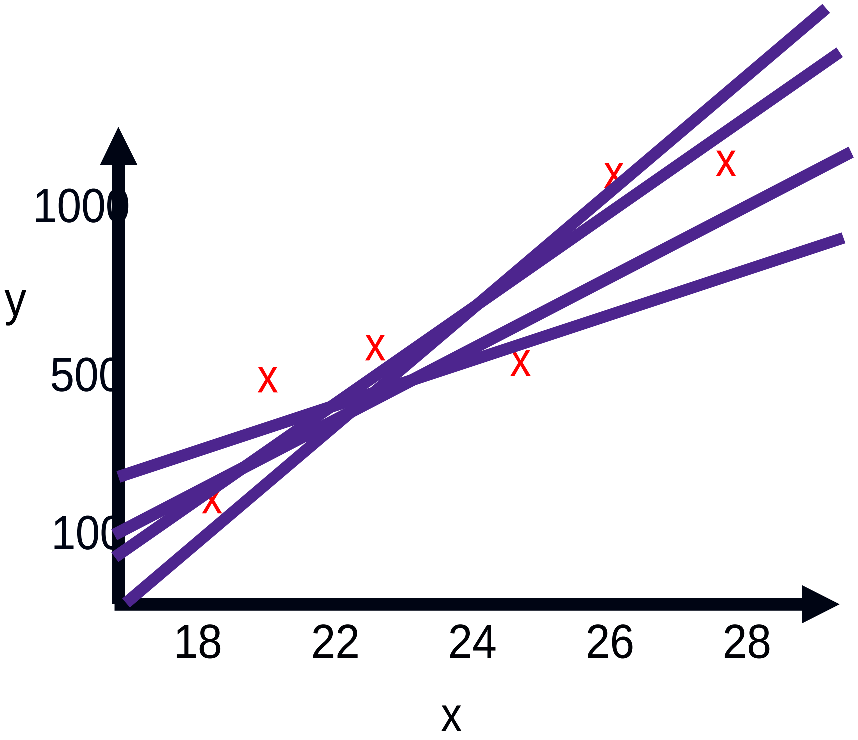
Fitting the Linear regression model

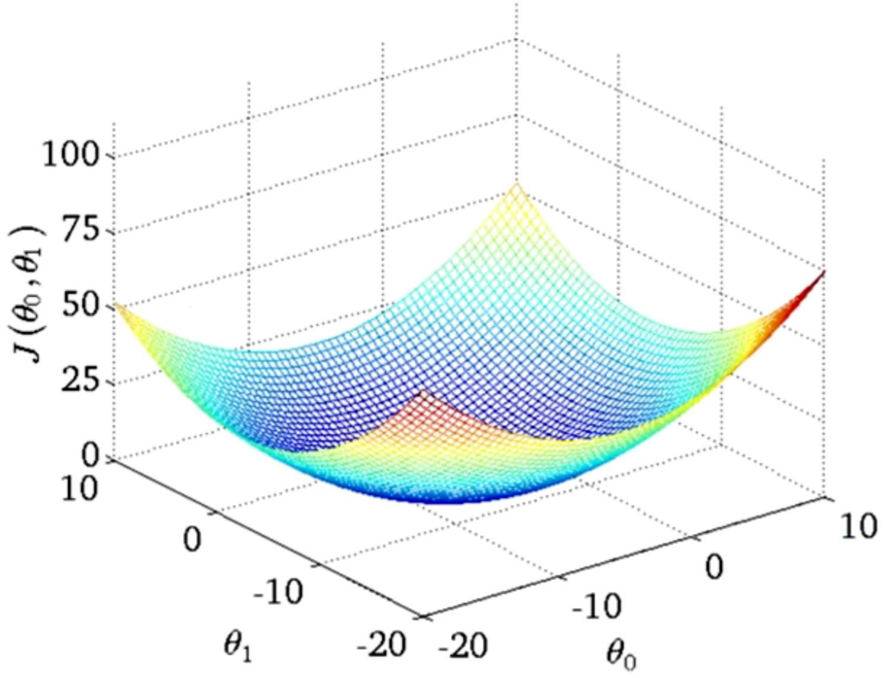
- If we assume $\theta_0 = 0$ then the cost function is parabolic.



Fitting the Linear regression model

- In general $\theta_0 \neq 0$ then the cost function is bowl shaped.





Note: *h(x)* and $J(\theta_0, \theta_1)$ are distinct. h(x) is a function of x for fixed theta. *J()* is a function of theta
We want an efficient algorithm to find the theta that minimizes the cost function.
For two model parameters we can visualize as above or as contour plots show *J()* of constant height

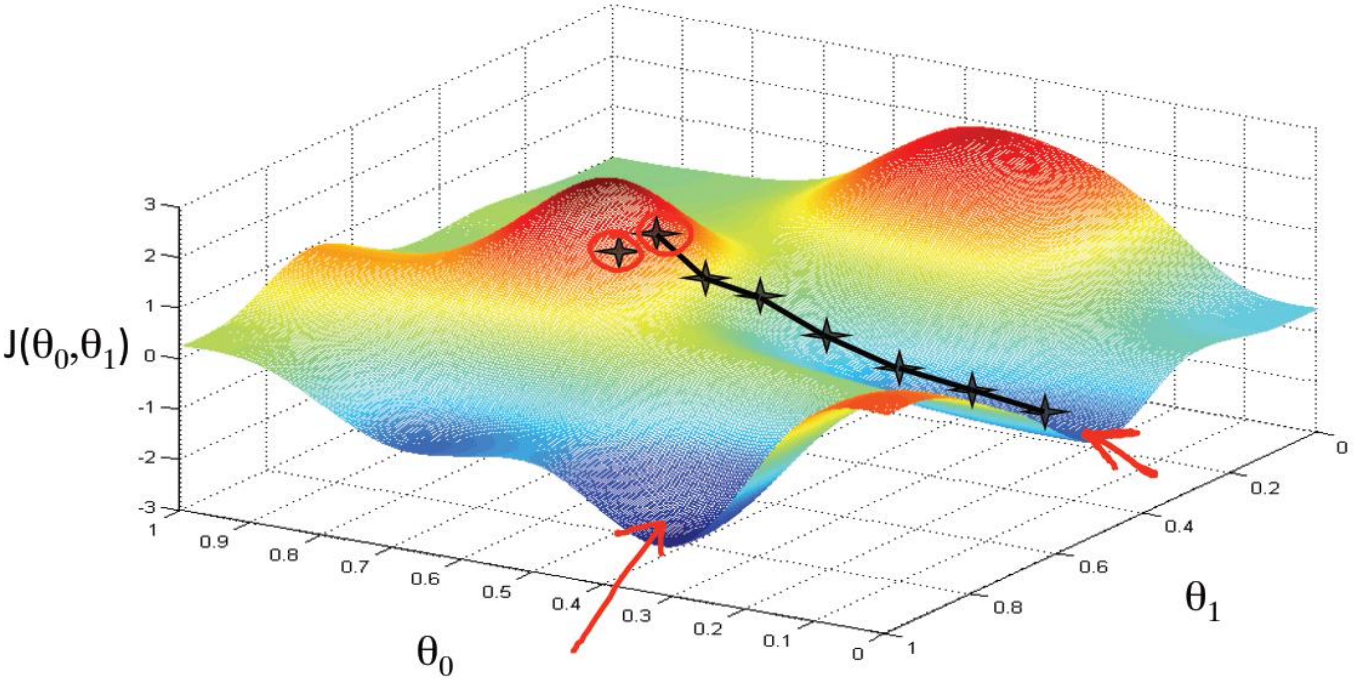
In many real world problems (#model parameters>>2) the cost function is higher dimensional so we cannot visualize it readily.

Supervised learning: Linear regression: gradient descent

- **We have a cost function $\boldsymbol{J(\theta_0, \theta_1)}$ that we want to minimize by finding optimal values of $\boldsymbol{\theta_0, \theta_1}$**

$$\min_{\boldsymbol{\theta_0,\theta_1}} \boldsymbol{J(\theta_0, \theta_1)}$$

- **Gradient descent procedure:**
  - **Start with an initial guess for $\boldsymbol{\theta_0, \theta_1}$**
  - **Iteratively change $\boldsymbol{\theta_0, \theta_1}$ to reduce $\boldsymbol{J(\theta_0, \theta_1)}$ until we (hopefully) end up at a minimum**
    - **Each iteration takes a step in the direction of the negative gradient of $\boldsymbol{J(\theta_0, \theta_1)}$**

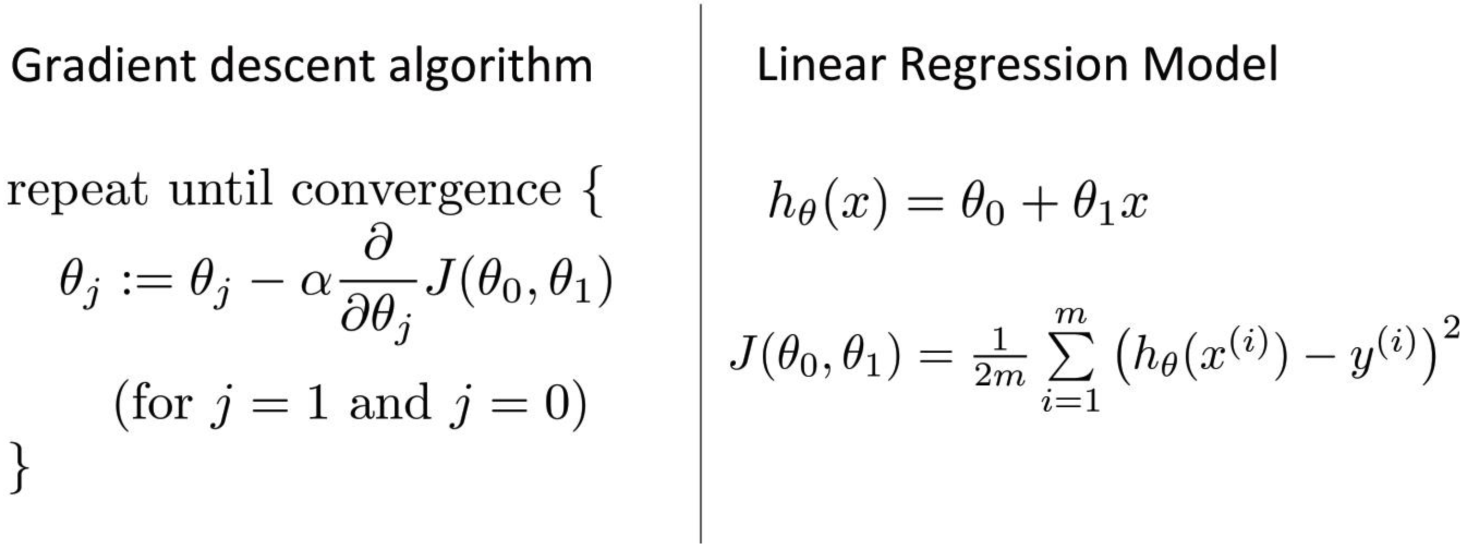


Gradient descent derivation

- We have a cost function $\boldsymbol{J(\theta_0, \theta_1)}$ that we want to minimize by finding optimal values of $\boldsymbol{\theta_0, \theta_1}$

$$\min_{\boldsymbol{\theta_0, \theta_1}} \boldsymbol{J(\theta_0, \theta_1)}$$

- Gradient descent algorithm and model:



Gradient descent algorithm

```
repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
        (for $j = 1$ and $j = 0$)
}
```

Linear Regression Model

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

$$J(\theta_0, \theta_1) = \tfrac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Gradient descent derivation

- We have a cost function $J(\theta_0, \theta_1)$ that we want to minimize by finding optimal values of θ_0, θ_1

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

- Gradient descent derivation:

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

For j=0 :

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)$$

For j=1 :

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) x^{(i)}$$

Gradient descent derivation

- **Gradient descent algorithm**

Repeat until convergence {

$$\boldsymbol{\theta_0} = \boldsymbol{\theta_0} - \boldsymbol{\alpha}\frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta_0} + \boldsymbol{\theta_1}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\right)$$

$$\boldsymbol{\theta_1} = \boldsymbol{\theta_1} - \boldsymbol{\alpha}\frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta_0} + \boldsymbol{\theta_1}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\right)\boldsymbol{x}^{(i)}$$

}

- This is called batch gradient descent because each step uses all of the *m* training examples

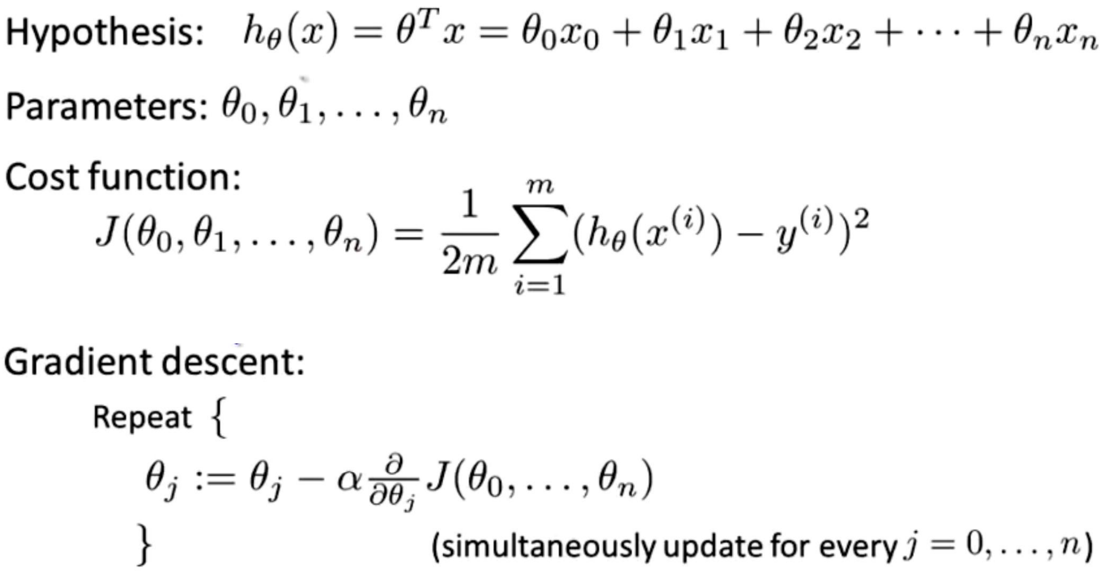
Multiple Linear regression

- Suppose we have more measures of each sample
  e.g. Predict CAC score from BMI, blood pressure, and age
- then our model has 3 measures rather than one:

$$h_{\theta}(x) = \theta^{T}x = \theta_{0}x_{0} + \theta_{1}x_{1} + \theta_{2}x_{2} + \theta_{3}x_{3}$$

where x_0=1 , x_1=BMI, x_2=bp, x_3=age

- In general for *n* measures we have:



Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

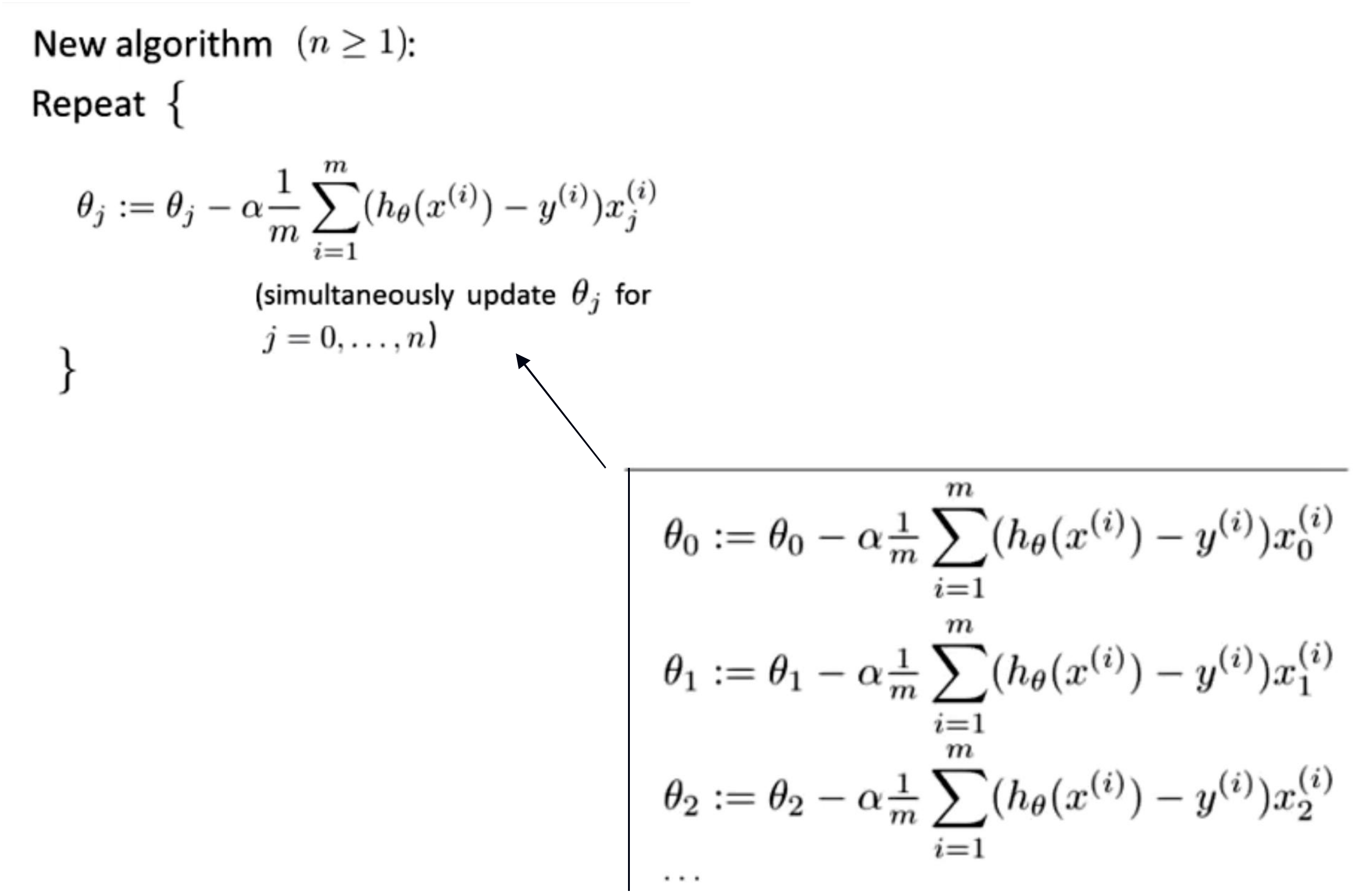
Repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$

} (simultaneously update for every $j = 0, \ldots, n$)

Multiple Linear regression

- **Our algorithm for gradient descent becomes:**



```
New algorithm $(n \geq 1)$:
Repeat {
    $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
        (simultaneously update $\theta_j$ for $j = 0, \ldots, n$)
}
```

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

$$\theta_1 := \theta_1 - \alpha \tfrac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \tfrac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$\ldots$

Multiple Linear regression: Practical considerations

- **Feature scaling for gradient descent**

  For efficient convergence to the optimal value of θ all features should be on a similar scale (have a similar range of values)

  1. Min-max scaling

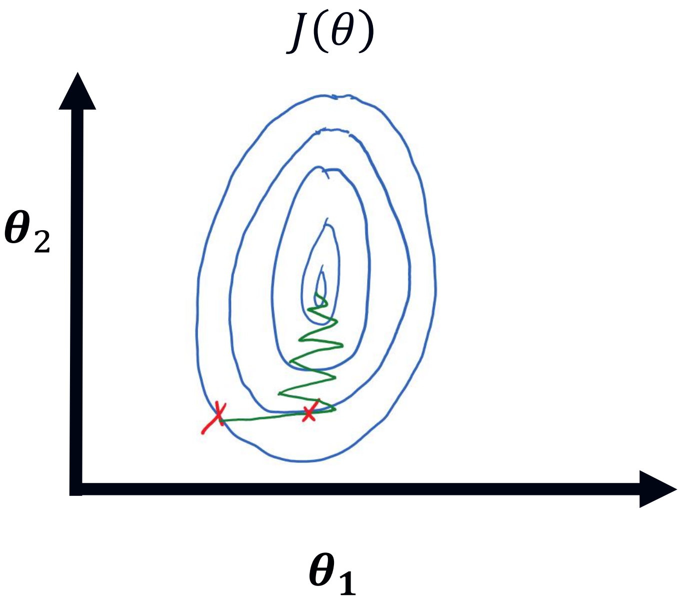
  - **E.g. If x_1 = 0-5000 while x_2 = 1-5**

    Then use rescaled features to similar range such as: $0 \leq x_i \leq 1$

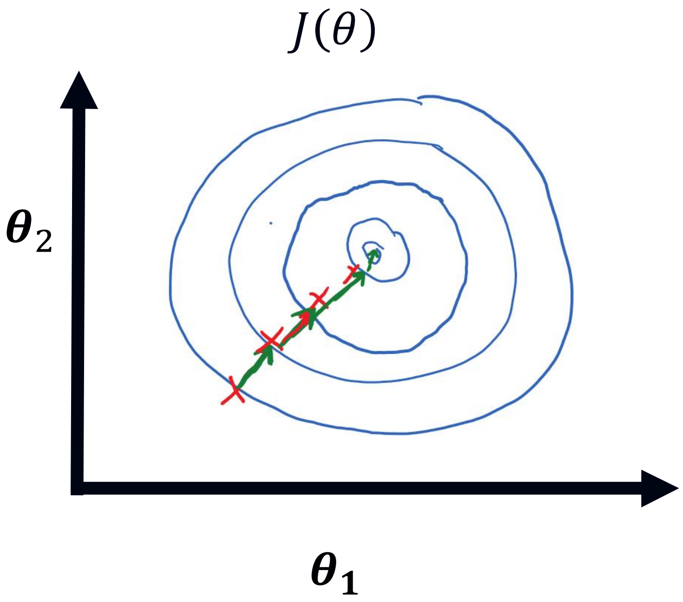
    $x_1' = x_1/ 5000$ and $x_2' = x_2/5$

  2. Standardization: Zero mean and unit variance

  Replace x_i with $\frac{x_i - \mu_i}{\sigma_i}$



Less efficient search



More efficient search

Multiple Linear regression: Practical considerations

- **How to choose learning rate α ?**

  $J(\theta)$ **should decrease on every iteration of GD. If increasing or cyclical, try smaller α**

  - **If α is too small … slow convergence**
  - **If α is too large … may not decrease in every iteration; or may diverge**

  - **Heuristic:**
    - **Try: α=0.001**
      if this doesn't diverge
    - **then try α=0.01**
    - **then 0.1 then 1 etc.**

Multiple Linear regression: Practical considerations

Comparison of gradient descent and normal equation strategies

- **Given: multiple linear regression with m samples, n features:**
  - **Let X be design matrix of m rows (samples) by n features per sample**
  - **Let y be the column vector of m labels,**
- **Then the normal eqn gives soln for our multiple linear regression model fitting in one step:**

$$\boldsymbol{\theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- **Gradient descent:**
  - **Need to choose α**
  - **Need many iterations**
  - **Works well even when n is large $n \geq 10^6$**
  - **Works even when $m \leq n$ particularly with regularization**

- **Normal equation:**
  - **No need to choose α**
  - **Since step, no iterations**
  - **Need to invert large matrix: $(X^TX)^{-1}$ requiring *O(n³)* operations**
    - **If $n \geq 10^4$ prefer gradient descent**
  - **X^TX may be non-invertible**
    - **Features are redundant: x_1=height in feet, x_2 = height in cm**
    - **More features than samples: $m \leq n$ à you must drop features or use regularization**

Supervised learning: logistic regression (classification)

- **Despite its name, logistic regression is a classifier**

- **Reminder: A classifier maps sample's observed feature vector, *x*, to a discrete valued output (*y*): $\mathbf{y} = \mathbf{h}(\boldsymbol{x}; \boldsymbol{\theta})$ rather than a cont val'd one.**

  - **For binary targets, $\mathbf{y} \in \{\mathbf{0}, \mathbf{1}\}$ can take on one of two values.**
    Tumor classification: benign, malignant

  - **For nominal targets, the target values are not intrinsically orderable**
    Glioblastoma subtype: classical, neural, mesenchymal, and proneural
    Often coded numerically: $\mathbf{y} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$

  - **For ordinal targets, the target values are orderable**
    Glioblastoma subtype: classical, neural, mesenchymal, and proneural
    Tumor grade: WHO Grade I, II, III, IV
    Often coded numerically: $\mathbf{y} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$

Supervised learning: logistic regression (classification)

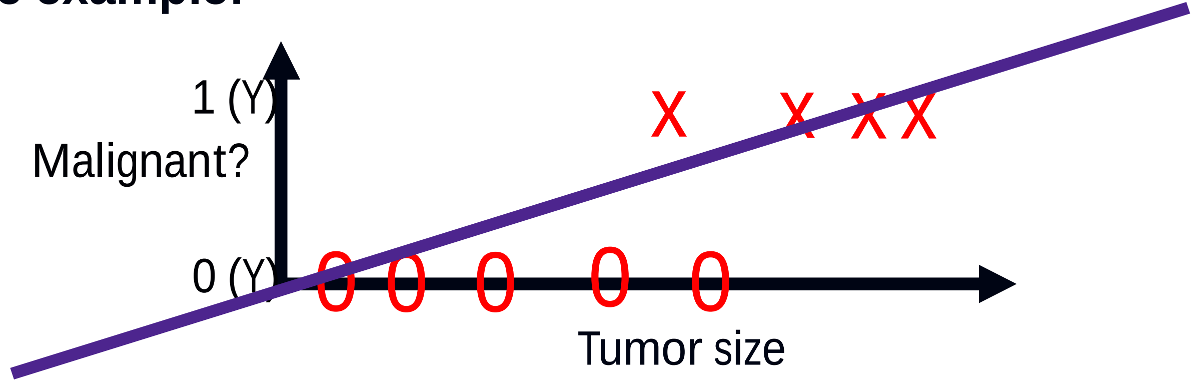
- Simplest form is binary classifier.

$$\mathbf{y} = \mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) \text{ where } \mathbf{y} \in \{\mathbf{0},\mathbf{1}\}$$

- Hypothesis predicts

$$\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{p}(\boldsymbol{y} = \mathbf{1} \mid \boldsymbol{x};\boldsymbol{\theta})$$

- Recall the example:



- Decision boundary : threshold $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta})$ at 0.5

  If $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) \geq \mathbf{0.5}$ predict y=1
  If $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) < \mathbf{0.5}$ predict y=0
  Decision boundary : threshold $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta})$ at 0.5

- Reasonable place to start linear model $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}$

  However $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta})$ can be >1 or <0

Supervised learning: logistic regression (classification)

- **Reasonable place to start linear model** $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}$

  However $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta})$ can be >1 or <0

- **We want a hypothesis mapping**

  $$\mathbf{0} \leq \mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) \leq \mathbf{1}$$

  such that $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta})$ tells us the probability of the predicted class being y=1:

  $$\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{p}(\boldsymbol{y} = \mathbf{1} \mid \boldsymbol{x};\boldsymbol{\theta})$$

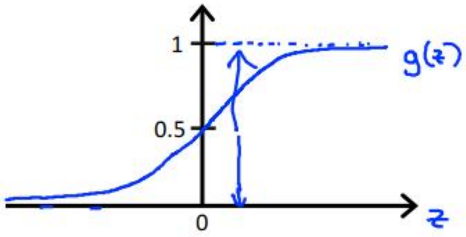
  Ex: If x=[x$_1$,x$_2$]^T = [1 tumorSize]^T and $\mathrm{h}(x;\theta)$ =0.75 then 75% chance tumor is malignant.

- **Logistic regression model:**

  $$\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x})$$

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

- $\boldsymbol{g}(\boldsymbol{z})$ **is the sigmoid function or logistic function (hence the models name)**

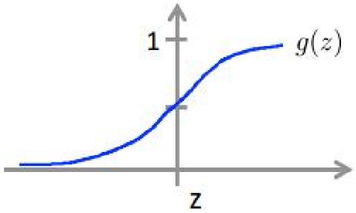


logistic regression: Decision boundary

- **Univariate model**:

$$\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{\theta}_{\mathbf{0}} + \boldsymbol{\theta}_{\mathbf{1}}\boldsymbol{x}_{\mathbf{1}})$$

Ex: predict y=1 if $\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) \geq \mathbf{0.5}$ i.e. when z ≥0 and predict y=0 o.w.

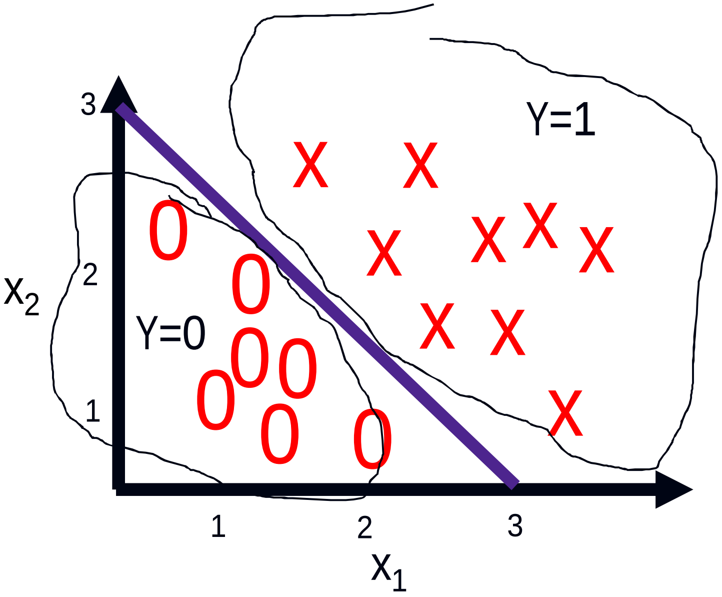


- **Multivariate model:**

$$\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{\theta}_{\mathbf{0}} + \boldsymbol{\theta}_{\mathbf{1}}\boldsymbol{x}_{\mathbf{1}} + \boldsymbol{\theta}_{\mathbf{2}}\boldsymbol{x}_{\mathbf{2}})$$

Ex: predict y=1 if $-3+x_1+x_2 \geq 0$

g() forms a cliff in feature space extending out of the x_1,x_2 plane for y=1 region

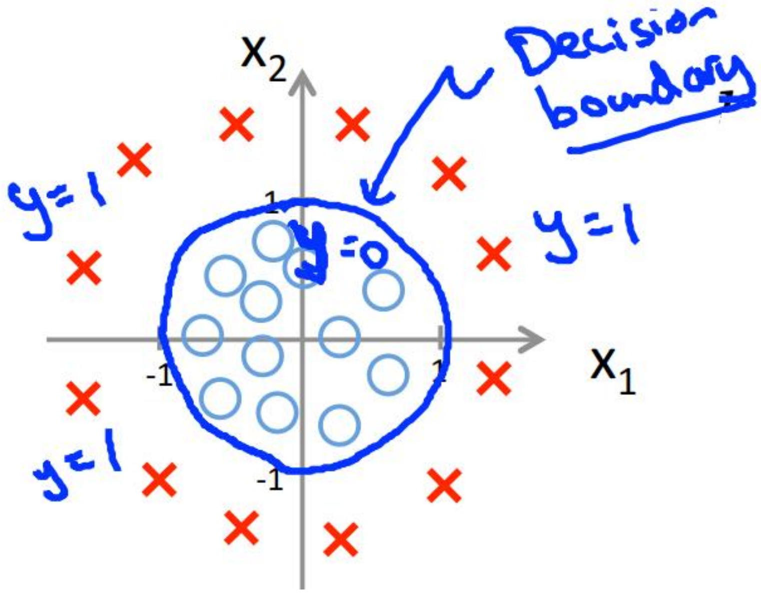


logistic regression: Decision boundary

- **Augmenting feature set with polynomial features, $\boldsymbol{x_1^2}$ $\boldsymbol{and}$ $\boldsymbol{x_2^2}$ allows for non-linear decision boundaries :**

$$\mathbf{h}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}^T\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{\theta_0} + \boldsymbol{\theta_1 x_1} + \boldsymbol{\theta_2 x_2} + \boldsymbol{\theta_3 x_1^2} + \boldsymbol{\theta_4 x_2^2})$$

Ex: predict y=1 if $\boldsymbol{-1 + x_1^2 + x_2^2 \geq 0}$



logistic regression: cost function

- Given
  - A set *D* of *m* labeled training data samples:

    $$D = \{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots , (x^{(m)}, y^{(m)}) \}$$

    where $x^{(i)} \in [x_0, x_1, x_2, . . , xn]^T$ is the ith training example with label $y^{(i)} \in \{0, 1\}$

  - $h(x; \theta) = \frac{1}{1+e^{-\theta^T x}}$
- How should we choose model parameters θ ?
- Cost function:

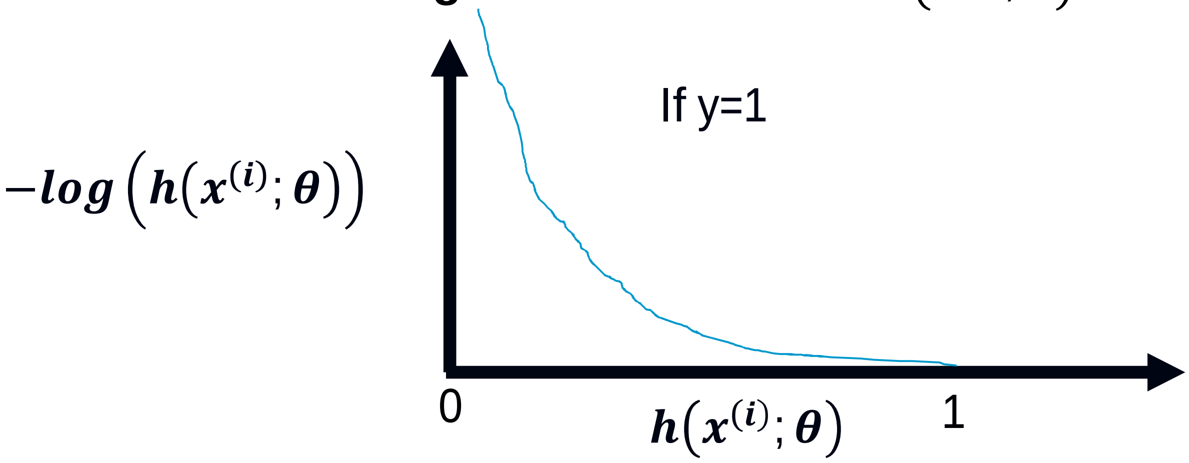
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h(x^{(i)}; \theta), y^{(i)})$$

$$Cost(h(x^{(i)}; \theta), y^{(i)}) = \begin{cases} -log\left(h(x^{(i)}; \theta)\right) & if\ y^{(i)} = 1 \\ -log\left(1 - h(x^{(i)}; \theta)\right) & if\ y^{(i)} = 0 \end{cases}$$

logistic regression: cost function   Case y=1

$$Cost\left(h\left(x^{(i)};\theta\right),y^{(i)}\right) = \begin{cases} -log\left(h\left(x^{(i)};\theta\right)\right) & if\ y^{(i)} = 1 \\ -log\left(1-h\left(x^{(i)};\theta\right)\right) & if\ y^{(i)} = 0 \end{cases}$$

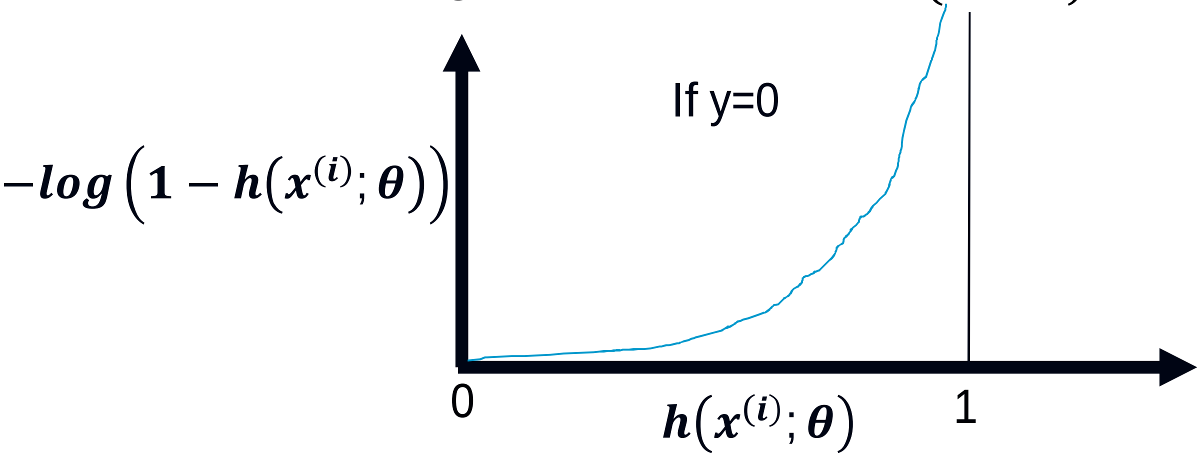
- When y=1 we want to penalize $p(y = 1 \mid x;\theta)$=0   because this is incorrect prediction
- This function assigns infinite cost as $h\left(x^{(i)};\theta\right) \to 0$



logistic regression: cost function   Case y=0

$$Cost\left(h\left(x^{(i)};\theta\right),y^{(i)}\right)=\begin{cases}-log\left(h\left(x^{(i)};\theta\right)\right) & if\ y^{(i)}=1\\ -log\left(1-h\left(x^{(i)};\theta\right)\right) & if\ y^{(i)}=0\end{cases}$$

- When y=0 we want to penalize $p(y=1\mid x;\theta)$=1 because this is incorrect prediction
- This function assigns infinite cost as $h\left(x^{(i)};\theta\right)\rightarrow 1$



logistic regression: Gradient Descent

- **For implementation our cost:**

$$\boldsymbol{Cost}\left(\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right),\boldsymbol{y}^{(i)}\right)=\begin{cases}-\boldsymbol{log}\left(\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right) & \boldsymbol{if}\ \boldsymbol{y}^{(i)}=\mathbf{1}\\ -\boldsymbol{log}\left(\mathbf{1}-\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right) & \boldsymbol{if}\ \boldsymbol{y}^{(i)}=\mathbf{0}\end{cases}$$

can be re-written as:

$$\boldsymbol{Cost}\left(\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right),\boldsymbol{y}^{(i)}\right)=-\boldsymbol{y}^{(i)}\boldsymbol{log}\left(\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right)-\left(\mathbf{1}-\boldsymbol{y}^{(i)}\right)\boldsymbol{log}\left(\mathbf{1}-\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right)$$

logistic regression: Gradient Descent cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)};\theta\right), y^{(i)}\right)$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)} log\left(h\left(x^{(i)};\theta\right)\right) - (1 - y^{(i)})\ log\left(1 - h\left(x^{(i)};\theta\right)\right)$$

- To fit θ we solve for

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

- To make a prediction for an untrained upon x:

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

$$where \quad h(x;\theta) = p(y = 1 \mid x;\theta)$$

logistic regression: Gradient Descent derivation

$$\boldsymbol{J(\theta)} = \frac{\boldsymbol{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} -\boldsymbol{y}^{(i)}\boldsymbol{log}\left(\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right) - (\boldsymbol{1}-\boldsymbol{y}^{(i)})\ \boldsymbol{log}\left(\boldsymbol{1}-\boldsymbol{h}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right)$$

- **To fit $\boldsymbol{\theta}$ we solve for** $\min_{\boldsymbol{\theta}} \boldsymbol{J(\theta)}$

Repeat until convergence {

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

}

- **After taking the derivative:**

Repeat until convergence {

$$\boldsymbol{\theta_j} = \boldsymbol{\theta_j} - \boldsymbol{\alpha}\frac{\boldsymbol{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\left(\boldsymbol{\theta_0} + \boldsymbol{\theta_1}\boldsymbol{x}^{(\boldsymbol{i})} - \boldsymbol{y}^{(\boldsymbol{i})}\right)\boldsymbol{x}^{(\boldsymbol{i})}$$

}

logistic regression: Advanced optimization

- Optimization algorithms include:
  - Basic:
    - Gradient descent
  - Advanced
    - Conjugate gradient
    - BFGS
    - L-BFGS

- Advanced algorithms:

  + do not require selecting α
  + typically faster than gradient descent
  - Are more complex … but implementations are available

Supervised learning: logistic regression

- What to do when the number of classes > 2?

  Multi-class classification: one vs all

- Train multiple 2 category classifiers:

  Train a logistic regression classifier $\boldsymbol{h}^{\boldsymbol{k}}(\boldsymbol{x};\boldsymbol{\theta})$ for each class, k, to predict the probability that y=k versus all the rest of the classes c≠k

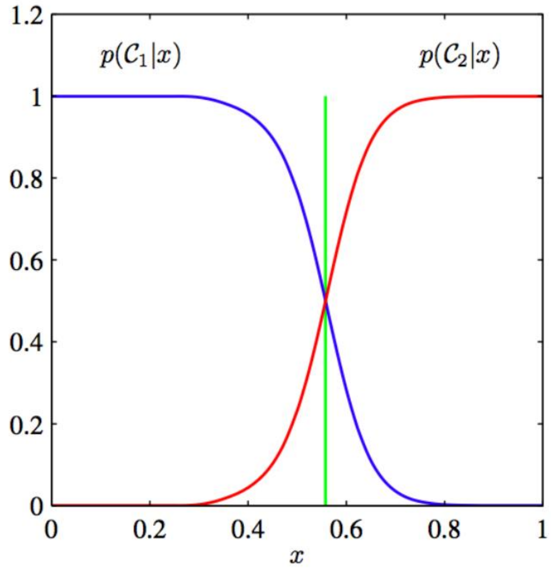
- For a new, test case, not in the training set, to make a prediction, choose the class k that maximizes:

$$\max_{\boldsymbol{k}} \boldsymbol{h}^{\boldsymbol{k}}(\boldsymbol{x};\boldsymbol{\theta})$$

Half way … would anyone prefer a break before continuing??

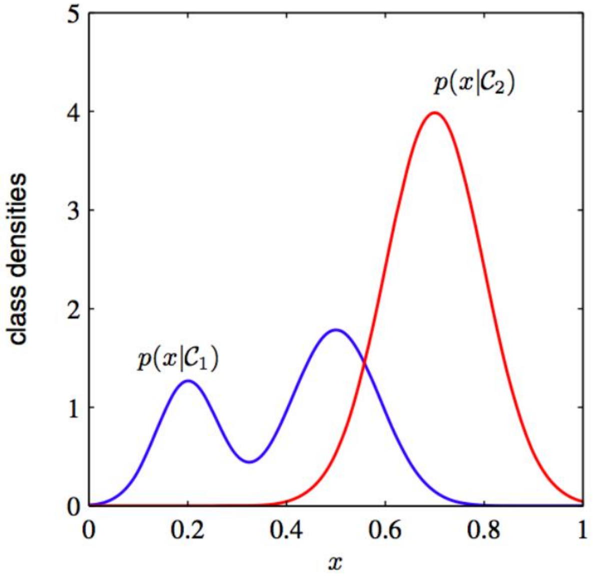
Generative vs discriminative models

- **So far we have discussed discriminative models**
  - **Choose class of decision (discriminant) functions in feature space**
  - **Estimate function parameters from training set**
  - **For classifiers, (logistic regression) new pattern is classified based on discriminant function**
  - **Fundamentally they directly estimate the class probability**



- **Generative probabilistic models**
  - **Model the density, $p(x|y)$ of input x for each class, $y \in \{C_1, C_2\}$**
  - **Estimate class prior probability $p(y)$**
  - **Use Bayes' rule to infer posterior distribution**

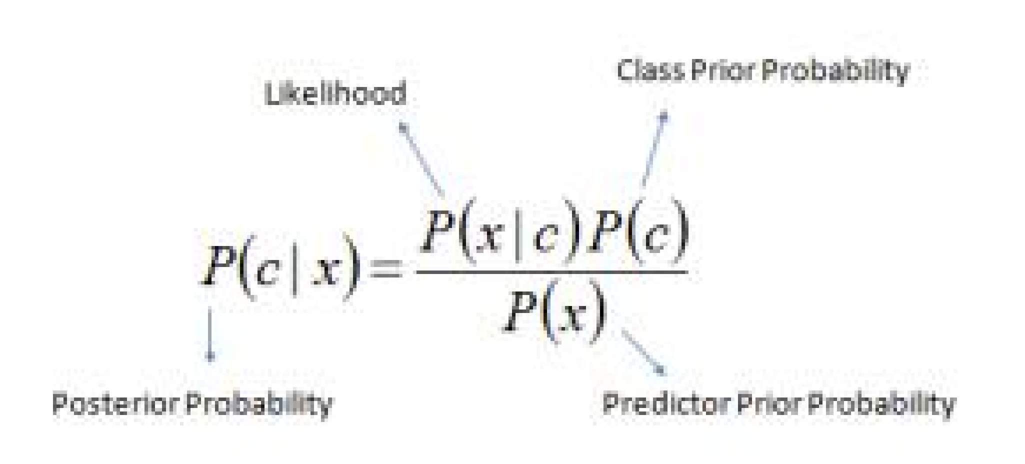
$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \quad \text{where } p(x) = \sum_y p(y)p(x|y)$$



Naïve Bayes Classifier

- A simple generative classifier for categorical inputs.
- Examples of categorical inputs (inputs w/limited possible values):
  - shape: round, not round (binary)
  - Color: red, orange, yellow, green (nominal)
  - Size: S (acorn), M (fist),L (head)   (ordinal)

- Based on Bayes rule which relates the input distribution for each class (likelihood of class c) to the posterior probability of the class P(y=c|x), that is present for a given input, x:



$$P(c\mid x)=\frac{P(x\mid c)P(c)}{P(x)}$$

- Factors on RHS are learned from the training data
- For inferencing, with novel data, x, choose the class with maximum posterior probability

Naïve Bayes Classifier

- It is called Naïve because it makes an assumption of the independence among the predictors
  - Presence of a particular feature in a class is unrelated to the presence of any other feature
    - A fruit may be classified as an apple if it is red, round and fist sized.
    - Watermelon if green, round, head sized
  - Assumption means all of these properties independently contribute to the probability that the fruit is an apple, even if these attributes depend on each other.

  - That is the a joint distribution can be decomposed into the product if independent distributions

  - If $\mathbf{X} = [x_0, x_1, x_2, . . , xn]^T$ then $p(X|c) = p(x_0|c)*p(x_1|c)*p(x_2|c)*...*p(x_n|c)$

- Putting these together
- $$p(y = c|X) = \frac{p(X|y = c)p(y=c)}{p(X)}$$

$$p(y = c|X) = \frac{p(x_0|c)*p(x_1|c)*p(x_2|c)*...*p(x_n|c)\ p(y = c)}{p(X)}$$

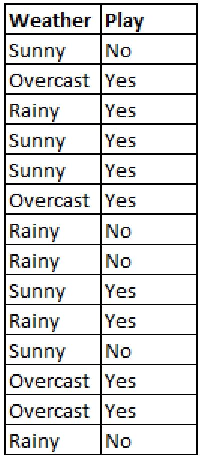
$$p(y = c|X) \propto \quad p(x_0|c)*p(x_1|c)*p(x_2|c)*...*p(x_n|c)\ p(y = c)$$

since p(X) dosen’t depend on class c, we need not compute it.

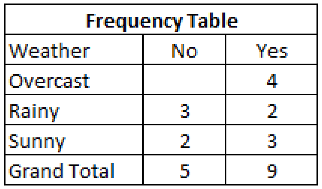
Naïve Bayes Classifier example

Given training data set of weather and corresponding outcome target variable Yes the soccer game was played or No it was not played

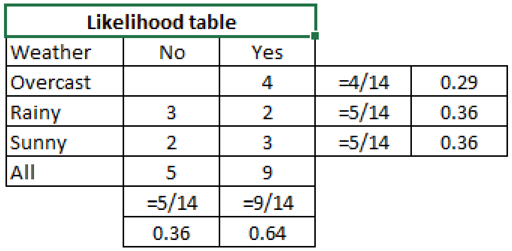
- Step 1) Convert the data into a frequency and likelihood tables:



Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No



Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9



Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

- Step 2) use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

  - P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)
  - P (Sunny |Yes) = 3/9 = 0.33;   P(Sunny) = 5/14 = 0.36;   P(Yes)= 9/14 = 0.64
  - P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability than P(No | Sunny) = 0.4

Gaussian Naïve Bayes Classifier

- **Extension of Naïve Bayes classifier to handle real-valued inputs. Still a classifier, not regression.**
  - **Requires only to compute training sample mean and variance of each x_i for each class.**
  - **Example: Label a voxel in a 3-channel image with one of 5 tissue types (segmentation), Compute training sample mean and variance of each image channel x_i for each of the 5 classes. Store these means and variances.**
  - **When a new image arrives, assign the MAP label to each voxel.**

- **Inferencing with Gaussian Naïve Bayes model**
  - **Given a newly observed input, X, compute the MAP(w) label**
  - **Compute $\left[\prod_{i=1}^{3} N(xi; \mu, \sigma^2, c_j)\right] * p(c_j)$ for each c_j. Then pick the c_j with the largest product.**
  - **That is we apply the Gaussian PDF and the computed mean and variance, compute the probability of each observed X**

where $N\left(x_i; \mu, \sigma^2, c_j\right) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$

- **Kernel functions:**
  - **rather than assuming Gaussian distribution for numerical inputs, more complex distributions can be used such as a variety of kernel density functions (beyond scope of ML1).**

Comparison of Naïve Bayes and logistic regression classifiers

- Both provide a final decision function to infer a class on new data.
- Naïve Bayes (NB) makes more restrictive assumptions and has higher asymptotic error, but converges faster to its less accurate asymptotic error.

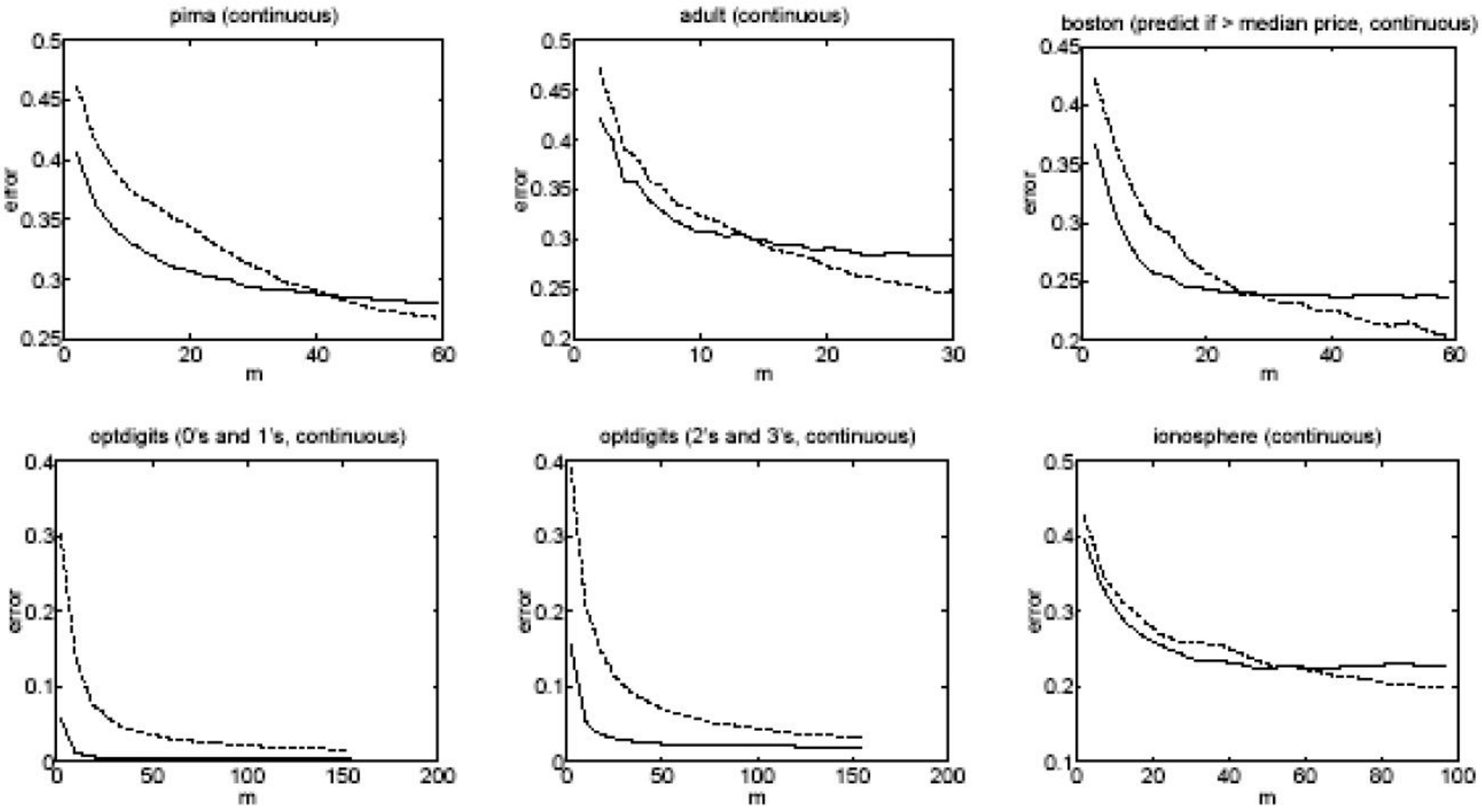
- Which you should choose? Depends on problem & data
- For problem with d features
  - Naïve Bayes needs fewer samples (O(log d)) to converge to its asymptotic error while LR requires O(d) samples. Why because NB learns parameter estimates independently, not coupled.

- If conditional independence assumption holds,
  - both have similar error

- If it does not hold,
  - LR has smaller error than NG

Experimental comparison of Naïve Bayes and logistic regression (Ng-Jordan’01)

UCI Machine Learning Repository 15 datasets, 8 continuous features, 7 discrete features



More in Paper…

—— Naïve Bayes

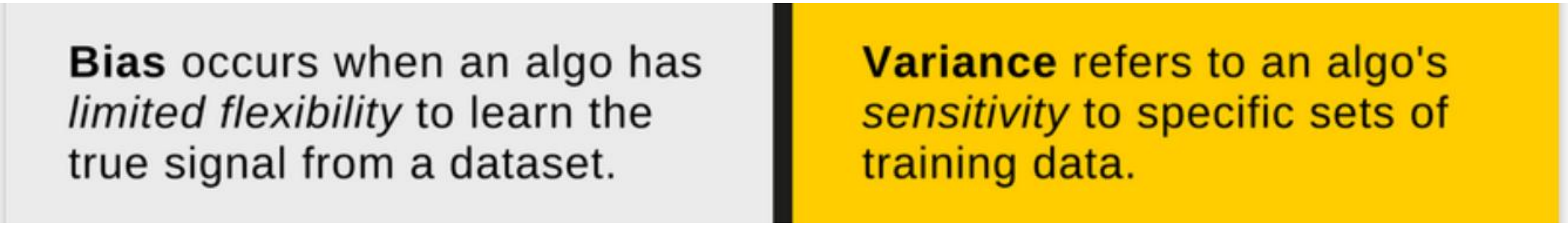
----- Logistic Regression

The Bias versus Variance tradeoff

- one of most important concepts to understand for supervised machine learning and predictive modeling
- Practical implications about model complexity, under/over fitting.
- There are many supervised learning algorithms to choose for predictive modeling: linear regression, logistic regression, GNB, neural network, random forest.
- They differ in several ways including the amount of bias vs variance which are two types of prediction error

Variance… is a measure of how much a predictive model varies about its mean for different training datasets

Bias .. Is a measure of how much the mean of a predictive model inferences on test data for different training sets, differ from the optimal predictor



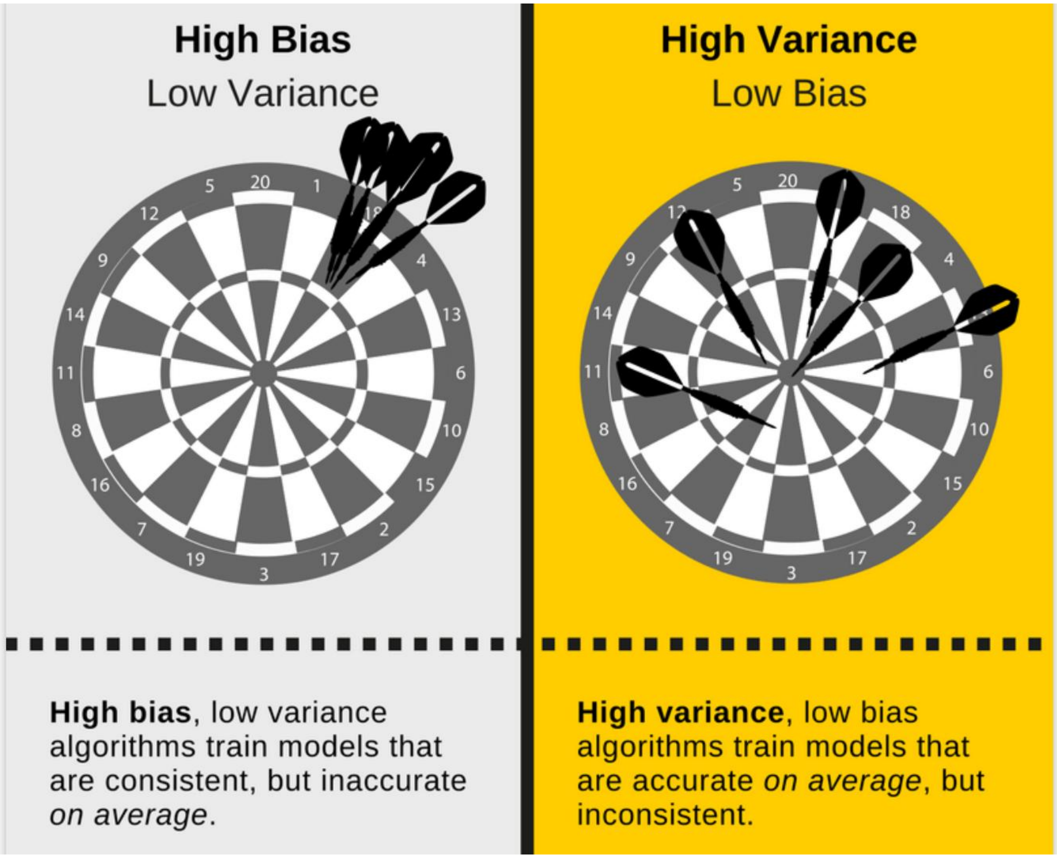
Bias occurs when an algo has *limited flexibility* to learn the true signal from a dataset.

Variance refers to an algo's *sensitivity* to specific sets of training data.

- **Consider model fitting not as a one time activity but a repeatable task.**

The Bias versus Variance tradeoff

- **Imagine that you have collected not 1 but 5 different training datasets (selected randomly from the population).**
- **Then you choose one algorithm to train 5 different fitted models, one for each training set.**
- **The performance of those 5 models on a separate set of test data reveals important characteristics of the chosen algorithm:**



High bias, low variance algorithms train models that are consistent, but inaccurate *on average*.

High variance, low bias algorithms train models that are accurate *on average*, but inconsistent.

The Bias versus Variance tradeoff

- **Why is there a tradeoff?**
  - **Low variance algorithms**
    - **tend to be less complex (statistically). They have fewer parameters. They tend to employ simple, rigid underlying structure:**
    - **Exs: Linear regression, Naïve Bayes, parametric algorithms**
  - **Low bias algorithms**
    - **tend to be more complex (statistically). They have more model parameters. They tend to employ flexible underlying structure which can adapt to the training data:**
    - **Exs: Random forest, nearest neighbor, non-linear algorithms (neural networks), non-parmetric algorithms**
- **Within each algorithm there is also a bias vs variance tradeoff:**
  - **Regression can be regularized to further reduce complexity**
  - **Random forest can employ pruned decision trees (reduced height) to reduce complexity**
- **Therefore a solid, proper model training approach is fundamental to successful use of machine learning.**

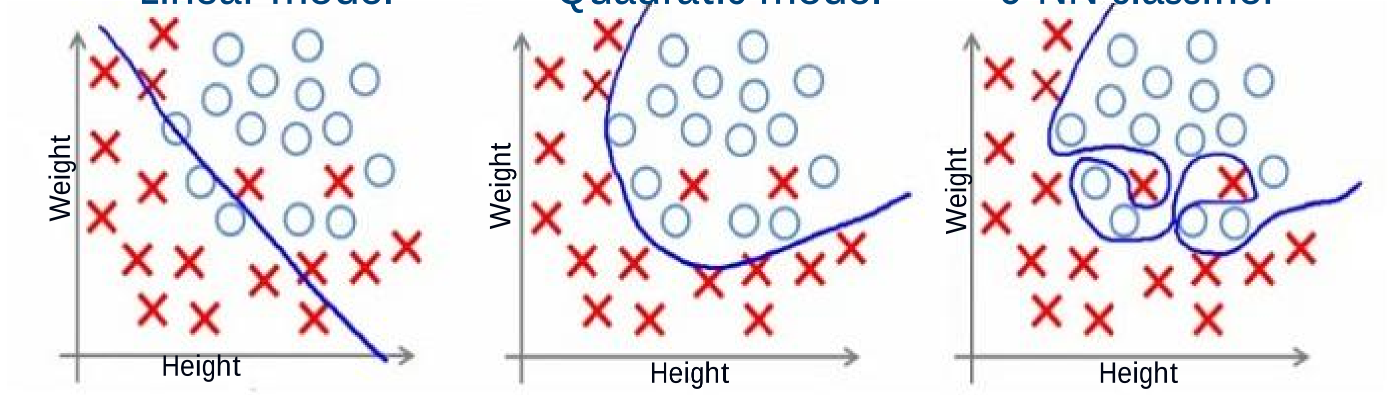
The Bias versus Variance tradeoff

- **Algorithms that are not complex enough or a training approach that is over regularized:**
  - **produce models that underfit and don’t learn the signal from the training data.**

- **Algorithms that are too complex or a training approach that is under regularized:**
  - **produce models that overfit and memorize the peculiarities (idiosyncrasies) of the training sample and its noise instead of learning the true generalizable signal from the data.**

Example: Classify Football player: X=no, O=yes

Linear model Quadratic model 0-NN classifier



Under-fitting **Appropriate-fitting** **Over-fitting**

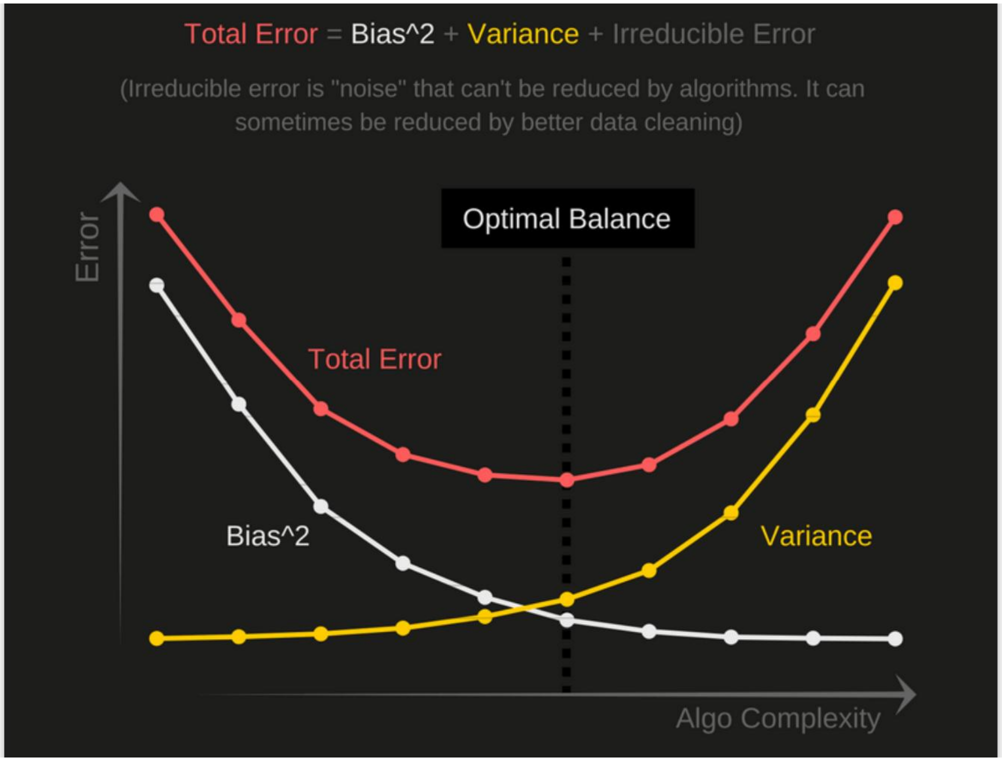
(too simple to explain the variance)

(forcefitting -- too good to be true)

Unlikely to generalize well

The Bias versus Variance tradeoff

- In order to get good prediction, a balance between bias and variance that minimizes the total error must be sought:



- Generally the more iterations of GD, the greater the model complexity (model is tuned to training sample).
- Optimal balance occurs when the total error on the test data is lowest.
- Error on training data tends to be driven asymptotically towards zero, but is generally not of interest for a predictive model.

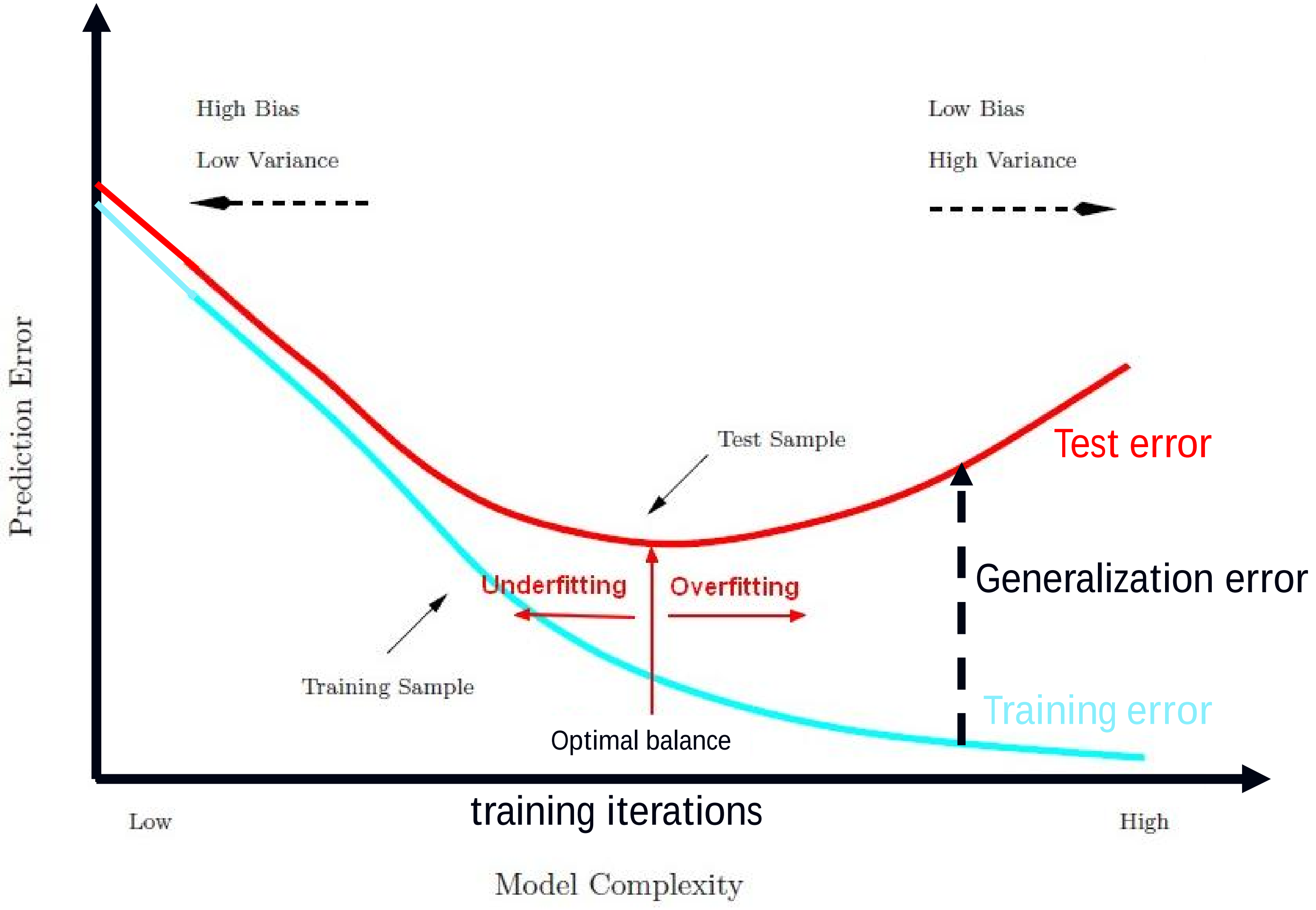
Finding the balance through proper model training workflow

Hallmarks of a proper ML model training workflow include:

1. **Separate training and test sets**
2. **Utilizing models with appropriate complexity**
3. **Employing regularization to optimize model complexity to the data**
4. **Appropriate performance metrics**
5. **Diagnostics that focus on test error and generalization error (relative to training error)**
6. **Employing early stopping criteria**
7. **Systematic use of cross validation rather than one train/test partition.**

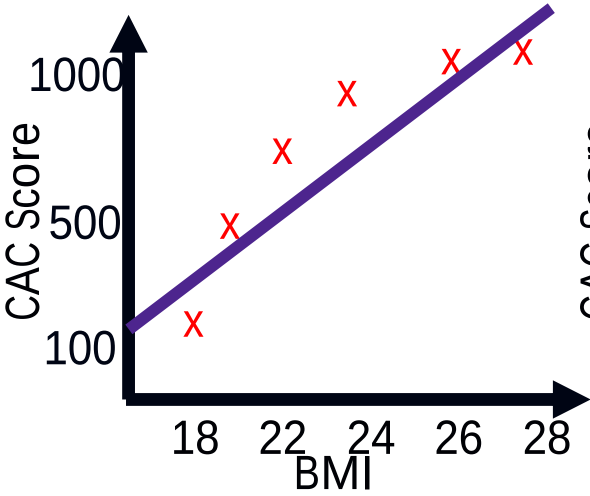
The Bias versus Variance tradeoff

- **Similar graph, different terms**



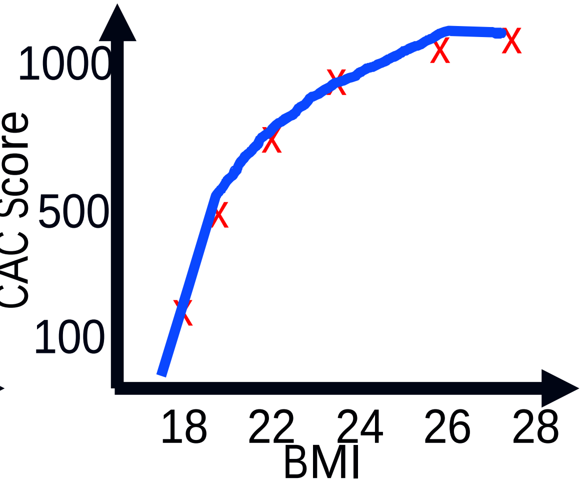
Underfitting and Overfitting: linear regression

Predict a patients coronary arteries calcium score (CAC) given their BMI



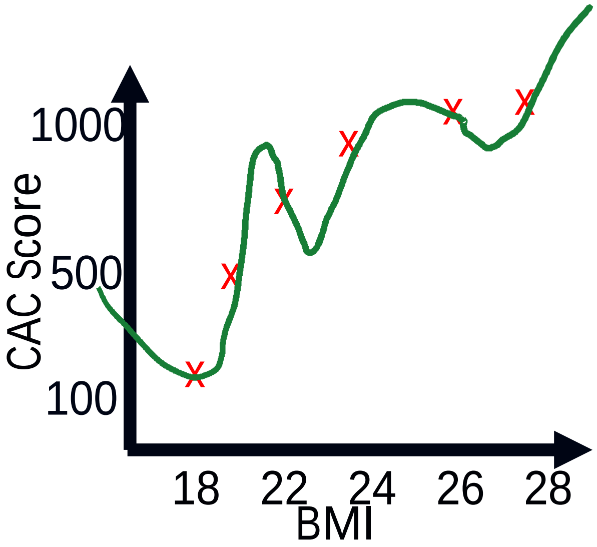
$\boldsymbol{h(x;\theta) = \theta_0 + \theta_1 x}$

Underfit, high bias, low variance



$\boldsymbol{h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2}$

Model feels “about right”



$\boldsymbol{h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5}$

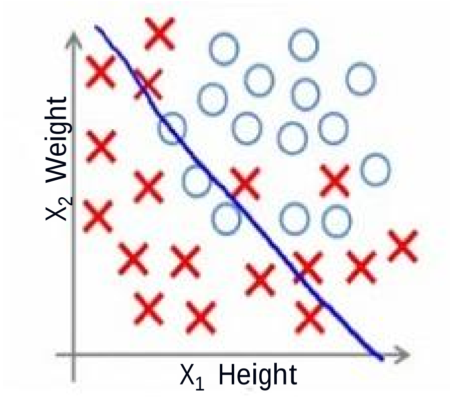
Overfit, low bias, high variance

- **In overfitting, the model can fit the training data really well, $\mathbf{J(\boldsymbol{\theta}) = 0}$, but fail to generalize to new samples not in the training set.**
- **In ML the emphasis is all about the test error, not training error. The construction of a predictive model. Contrast with many statistical analyses.**

Underfitting and Overfitting: logistic regression

Example: Classify Football player: X=no, O=yes

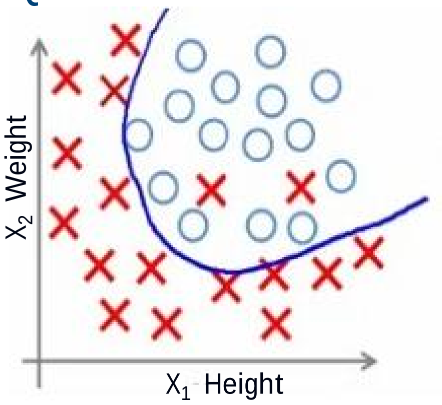
Linear model



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

Underfit, high bias, low variance

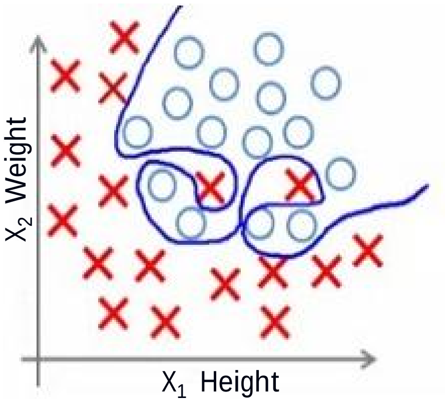
Quadratic model



$$h(x;\theta) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

Model feels “about right”

Higher order classifier



$$h(x;\theta) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + +\theta_6 x_1^3 x_2 + \cdots)$$

Overfit, low bias, high variance

- **In logistic regression we see the use of the sigmoid that converts the regression to a classification, but again we see three fundamental cases.**
- **In general we do not expect always zero error even on training data. We expect it to be low (Can you say why?) but the features may not perfectly separate classes and noise in measurements.**
- **To minimize test error, we aim for a predictive model (in the middle) that is a trade off between low variance (left) and low bias (right).**

Unlikely to generalize well

Approaches to address overfitting

1. Feature selection

- Manually select which features to keep
  - May not know in advance which features are valuable; if you throw out valuable features you will get sub-optimal performance
- Model selection …
  - Requires training multiple models
  - Model selection approach

2. Regularization

- Retain all features but shrink the magnitude of the parameters θ_j
- Comes in different flavors, we will discuss L_2 regularization
- This form works well when we have many features m>>1 and each of them contributes at least a small amount to an accurate prediction . Conversely few or none of them have zero effect.

Why don’t we simply plot the modeled hypothesis and simply pick the model that looks about right?

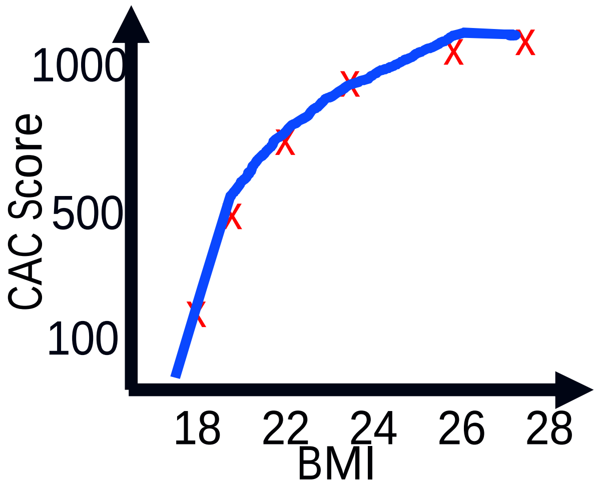
most problems >>2 features, difficult to visualize

How would you address underfitting?

Generally this is not a problem, add more model flexibility or complexity

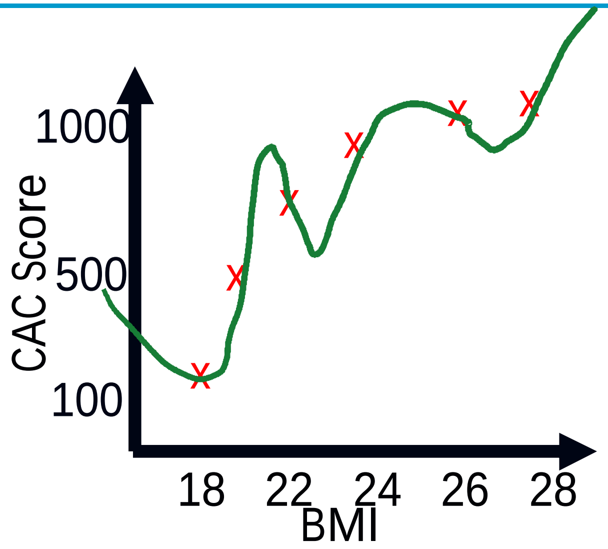
E.g. add more features: measured features, polynomial, etc..

Regularization: Intuition



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

Model feels "about right"



$h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$

Overfit, low bias, high variance

- **If we could drive θ_3, θ_4, θ_5 to be close to zero then our model will behave like a quadratic.**
- **So we add a penalty term to our cost function to make them small:**

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

First sum iterates over *m* samples

Second sum iterates over *n* features (but not the bias term, θ_0)

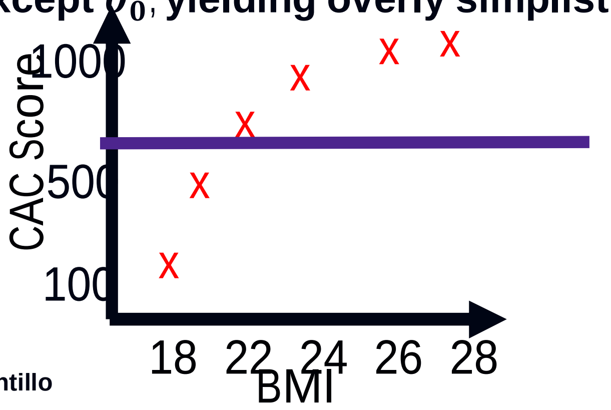
Regularization: Intuition

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

λ is the regularization parameter: controls the trade off between fitting the data well and keeping the parameters small, through a simpler model.

- **Setting λ allows to directly regulate model’s statistical complexity.**
- **Increasing λ increases regularization**
  - **makes models less prone to overfitting**
  - **Results in smoother models**

  - **If we set it loo large $\lambda = 10^{12}$ then we will push model into underfitting regime, where all fitted parameters are zero except θ_0, yielding overly simplistic $h(x;\theta) = \theta_0$**



Apply regularization to multiple linear regression

- **Recall, non-regularized, multiple linear regression for *n* features:**

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- **Regularized linear regression**
  - **Add term to penalized large coefficient weights (L$_2$ regularization)**

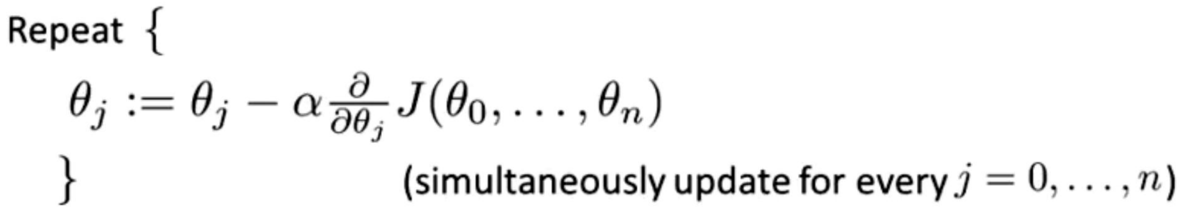
$$\boldsymbol{J(\theta)} = \frac{\mathbf{1}}{\mathbf{2m}} \left[\sum_{i=1}^{m} \left(h\left(x^{(i)}; \theta\right) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

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

Apply regularization to multiple linear regression

To fit $\boldsymbol{\theta}$ we solve for $\min_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})$:

- **Non-regularized gradient descent:**



Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

} (simultaneously update for every $j = 0, \ldots, n$)

- **Regularized gradient descent:**

Repeat until convergence {

$$\boldsymbol{\theta_0} = \boldsymbol{\theta_0} - \boldsymbol{\alpha}\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}}\left(h\left(\boldsymbol{x}^{(i)};\theta\right) - \boldsymbol{y}^{(i)}\right)\boldsymbol{x}_0^{(\boldsymbol{i})}$$

$$\boldsymbol{\theta}_j = \boldsymbol{\theta}_j - \boldsymbol{\alpha}\left[\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}}\left(h\left(\boldsymbol{x}^{(i)};\theta\right) - \boldsymbol{y}^{(i)}\right)\boldsymbol{x}_j^{(\boldsymbol{i})} \quad + \frac{\lambda}{m}\theta_j\right]$$

}

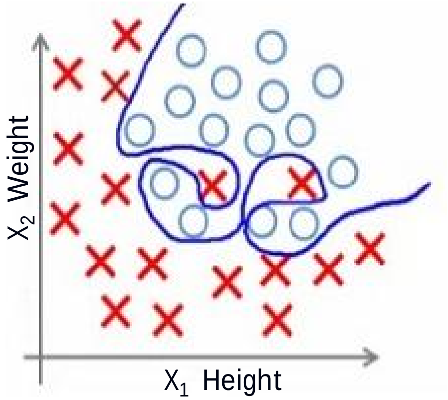
Factoring out $\boldsymbol{\theta}_j$ shows that we are shrinking $\boldsymbol{\theta}_j$ by a multiplier<1 on every GD iteration: $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j\left(\mathbf{1} - \boldsymbol{\alpha}\frac{\boldsymbol{\lambda}}{\boldsymbol{m}}\right) - \boldsymbol{\alpha}\left[\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}}\left(h\left(\boldsymbol{x}^{(i)};\theta\right) - \boldsymbol{y}^{(i)}\right)\boldsymbol{x}_j^{(\boldsymbol{i})}\right]$

Apply regularization to Logistic regression (classification)

- **Recall non-regularized logistic regression:**
  - **Hypothesis:** $\boldsymbol{h(x;\theta)} = \frac{\mathbf{1}}{\mathbf{1+e^{-\theta^T x}}}$
  - **Parameters:** $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \dots, \theta_n]$
  - **Cost function:** $\boldsymbol{J(\theta)} = -\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{y^{(i)} log\left(h\left(x^{(i)};\theta\right)\right) + \left(1-y^{(i)}\right) log\left(1-h\left(x^{(i)};\theta\right)\right)}$

- **Regularized logistic regression:**
  - **Add a term to penalized large coefficient weights (L$_2$ regularization)**

Higher order classifier



$$\boldsymbol{h(x;\theta) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + +\theta_6 x_1^3 x_2 + \cdots)}$$

Overfit, low bias, high variance

Goal: find $\min_{\theta} J(\boldsymbol{\theta})$ **of our logistic regression cost function while penalizing large parameters:**

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

Apply regularization to Logistic regression (classification)

To fit $\boldsymbol{\theta}$ we solve for $\min_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})$:

- **Non-regularized gradient descent:**

$$\text{Repeat } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$
$$\}$$

(simultaneously update for every $j = 0, \ldots, n$)

- **Regularized gradient descent:**

Repeat until convergence {

$$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_0 - \boldsymbol{\alpha} \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \left(h\left(\boldsymbol{x}^{(\boldsymbol{i})}; \theta\right) - \boldsymbol{y}^{(\boldsymbol{i})}\right) \boldsymbol{x}_0^{(\boldsymbol{i})}$$

$$\boldsymbol{\theta}_j = \boldsymbol{\theta}_j - \boldsymbol{\alpha} \left[\frac{\mathbf{1}}{m} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \left(h\left(\boldsymbol{x}^{(\boldsymbol{i})}; \theta\right) - \boldsymbol{y}^{(\boldsymbol{i})}\right) \boldsymbol{x}_j^{(\boldsymbol{i})} \quad + \frac{\lambda}{m} \theta_j \right]$$

}

On the surface looks the same as linear regression, but h() is not the same:

$$\boldsymbol{h}(\boldsymbol{x}; \boldsymbol{\theta}) = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

Model selection

- By now you see there are many design decisions about the hyperparameters:
  features to include, form of hypothesis, regularization parameters

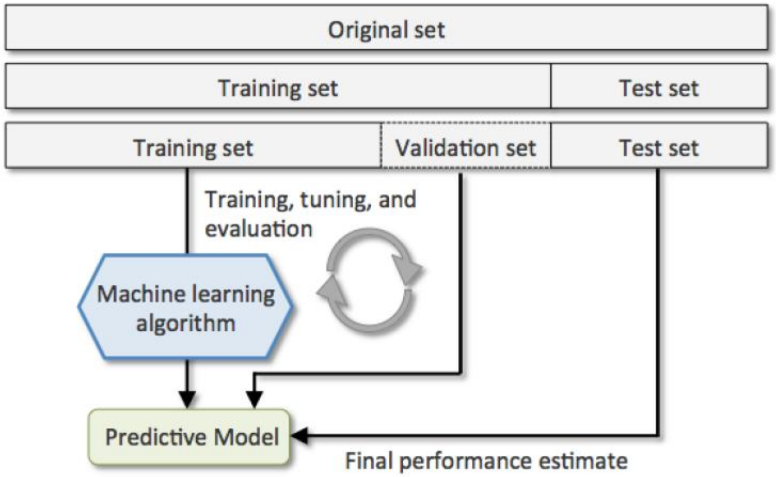
- Choosing between them is known as the model selection task.
- Commonly used methods:
  - Cross-validation
  - Risk minimization
  - Complexity regularization

- Focus on cross-validation

Cross-validation requires partitioning your data

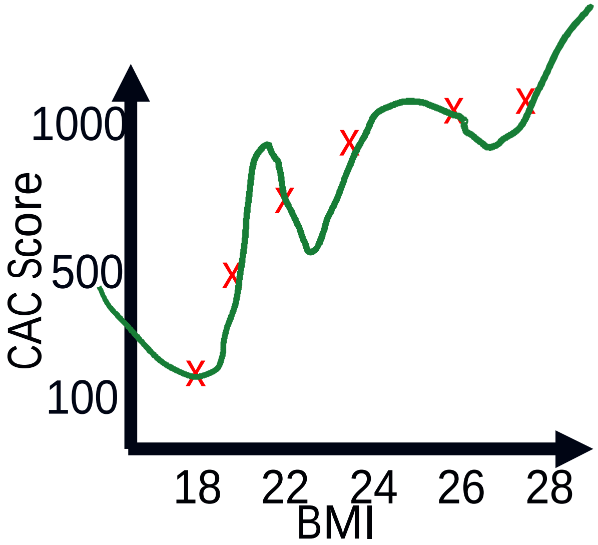
- Randomly shuffle your data
- Partition data (inputs and labels) into disjoint sets:
  - train (inputs and labels),
  - validation (inputs and labels),
  - test (inputs and labels)

- Purposes are distinct
  - Training is used for fitting each candidate model's parameters
  - Validation is used for comparing candidate models
  - Test is used for estimating model performance in real world, used only once


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

Raschka, 2015

Model selection: choosing the form of the ML model



- Compute error on each partition separately

Training error:

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

Validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

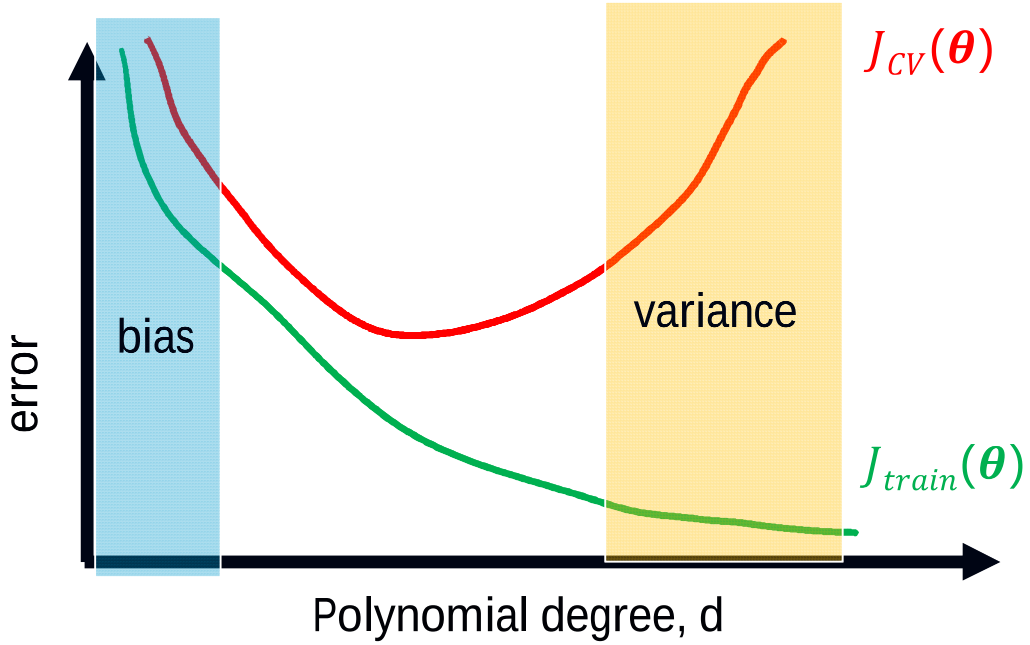
- Suppose you decide your candidate models are:
  1. $h(x;\theta) = \theta_0 + \theta_1 x$
  2. $h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2$
  3. $h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
  4. $h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
  5. $h(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$

Then the procedure is:

- Fit all your models' parameters, θ, to the train data partition by optimizing J_{train}
- Choose between them by computing cost on validation J_{cv} and picking the one with the min J_{cv}
- Estimate real-world performance by computing J_{test}

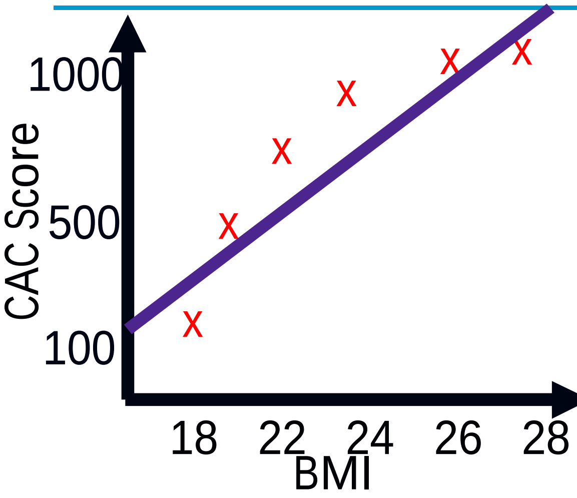
Model selection: diagnosing bias vs variance

- If J_{cv} or J_{test} is higher than you would like, how can you tell if this is a bias or a variance problem?

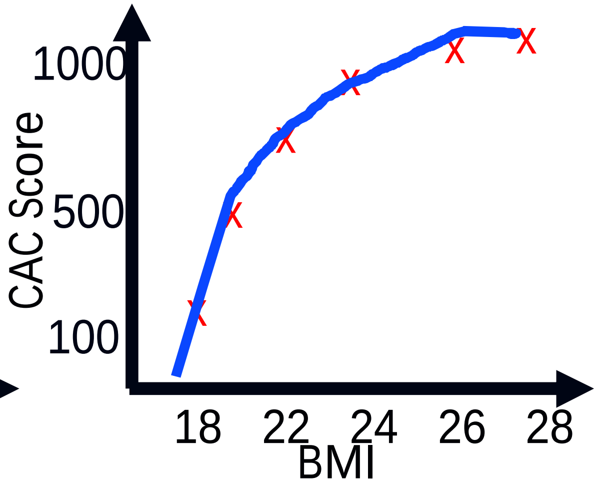


- **The left is high bias.**
  - Model is underfit
  - Jtrain is high
  - Jcv ≈ Jtrain
- **The Right is high variance.**
  - Model is overfit
  - Jtrain is low
  - Jcv >> Jtrain

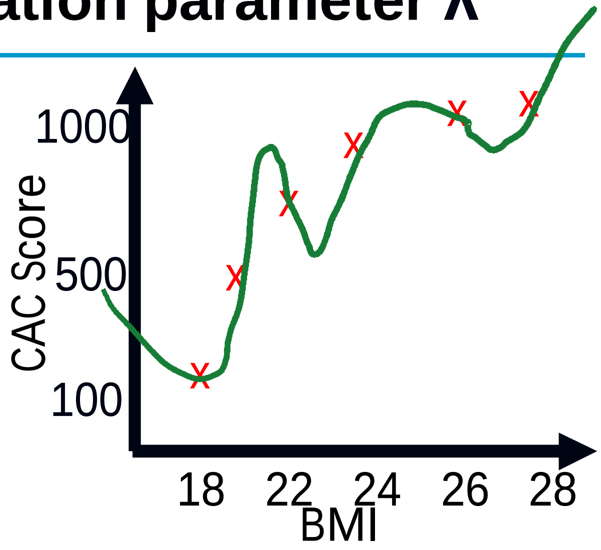
Model selection: choosing the regularization parameter λ



λ too large, e.g. λ=10⁶
Underfit, high bias



λ is just about right



λ is too small
Overfit, low bias, high variance

Reasonable search strategy for λ :

1. **Consider λ=0**
2. **Consider λ=0.01**
3. **Consider λ=0.02**
4. **Consider λ=0.04**

...

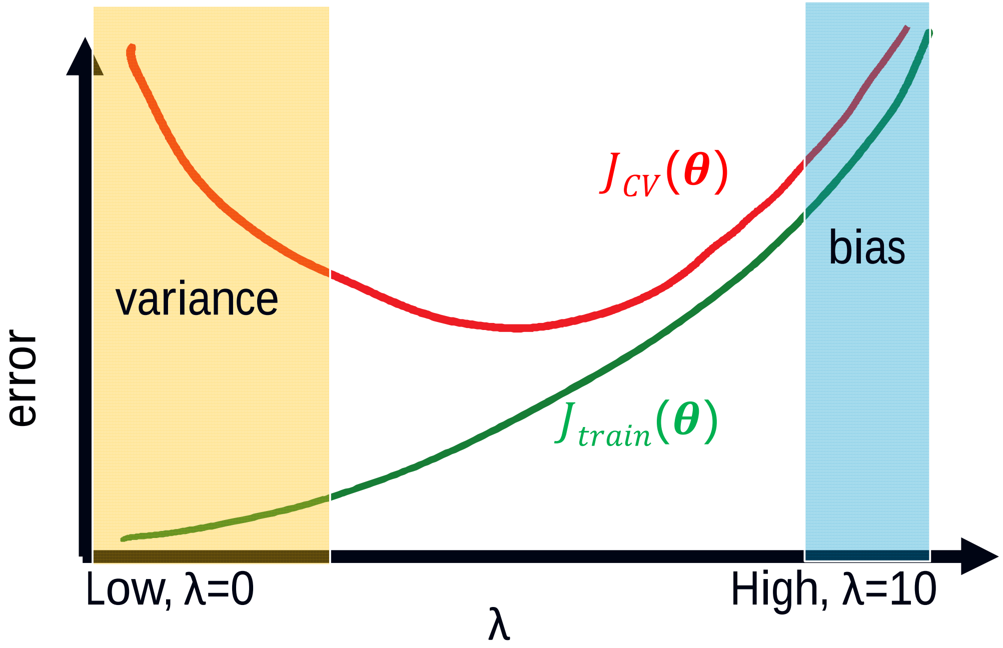
12. **Consider λ=10.24**

Then the procedure is:

- **For each λ train the model by optimizing J_{train}**
- **Choose between them by computing validation J_{cv} and picking the one with the min J_{cv}**
- **Estimate real-world performance by computing J_{test}**

Model selection: choosing the regularization parameter λ

- If J_{cv} or J_{test} is higher than you would like, how can you tell if this is a bias or a variance problem?



- The left is high variance.
  - Model is overfit
  - Jtrain is low
  - Jcv >> Jtrain

- The Right is high bias.
  - Model is underfit
  - Jtrain is high
  - Jcv ≈ Jtrain

Practical advice

Lets say your regression model isn't producing performance as good as you would like (J_{test} is to high). What are your options and likley impact?

- **Acquire more training samples**
  - **Can fix high variance, but at some added cost**
- **Use fewer features (down select)**
  - **Using a simpler model helps reduce high variance, without added cost**
- **Acquire additional features and increase model flexibility**
  - **More flexible, complex model addresses high bias.**
  - **If features are not available, must reacquire data at some cost**
- **Try adding polynomial features**
  - **This too increase model flexibility and addresses high bias, but is relatively cheap**
- **Decrease λ**
  - **This makes model more flexible, more able to choose parameters freely.**
  - **Thus it helps decrease high bias**
- **Increase λ**
  - **This makes model less flexible, drives parameters to be small.**
  - **Thus it helps decrease high variance.**

Pop Quiz!

1. What is the difference between regression and classification?
   - Reg: predicts continuous val'd outputs, class: categorical
2. What is the difference between generative & discriminative models?
   - Generative models the input for each class, discrim: directly models dec bndy
3. What is difference btwn supervised and unsupervised learning?
   - Ground truth is provided for supervised, no labels for unsupervised
4. What is the bias vs variance trade off?
   - Compromise between mean prediction accuracy (bias) and sensitivity to training data
5. What is an example of a generative model?
   - Naïve Bayes
6. What is model selection?
   - The choice of hyperparameters. E.g. degree of polynomial, features to include, regularization amount

References

For further information on machine learning :

- Pattern Recognition and Machine Learning, Christopher Bishop.
- Machine Learning, Tom Mitchell.
- The Elements of Statistical Learning: Data Mining, Inference and Prediction, Trevor Hastie, Robert Tibshirani, Jerome Friedman.

Questions

Albert.Montillo@UTSouthwestern.edu