Machine Learning

Recap of last class

- Markov Decision Problems
- Expected utility of a state
- Bellman equation
- Value iteration
- Policy iteration

Agenda

- Introduce definition of machine learning
- Consider types of problems amenable to ML
- Describe general categories of ML techniques
- Introduce different types of learning
- Understand how to avoid overlearning

Early Definition

- Machine learning: "Field of study that gives computers the ability to learn without being explicitly programmed”

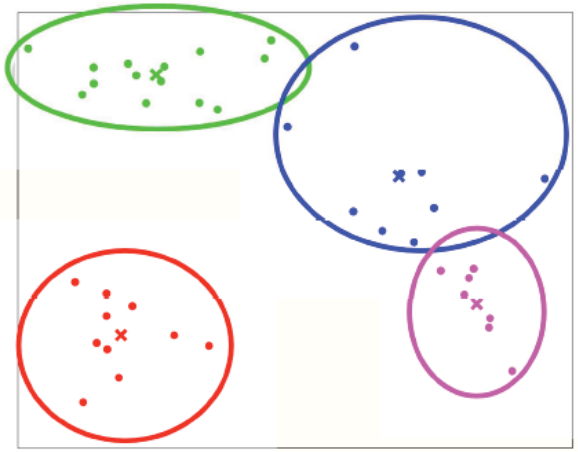
-- Arthur Samuel (1959)

ML Problems

- Clustering
- Data reduction
- Classification
- Regression

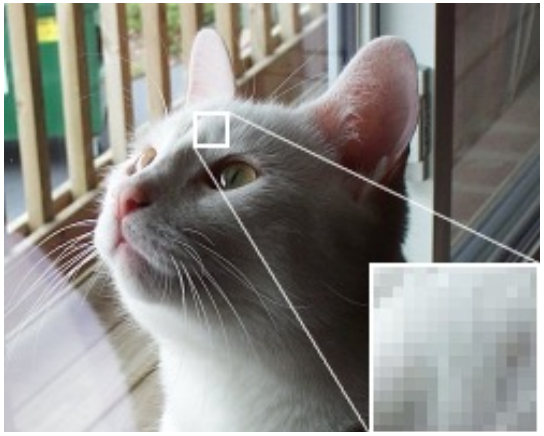
Clustering

- Given a set of data points, group the points in clusters according to a criterion.
  - e.g., divide the set of photos into four groups of people

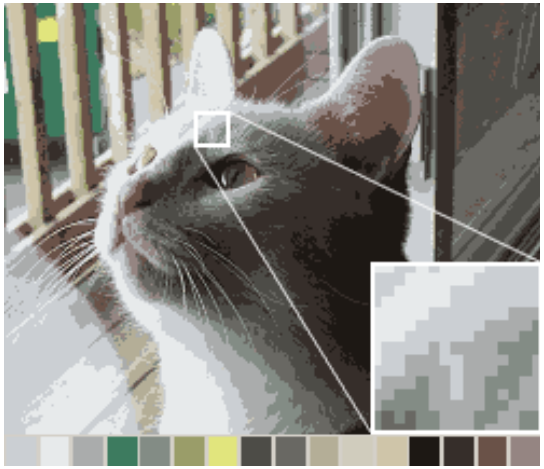


Dimensionality Reduction

- Given a set of points, determine how many dimensions are required to represent it. Potentially find the dimensions that are most important or generate new ones.
  - e.g., compress a colour image to store it in the minimum number of bits with sufficient quality to identify who is who



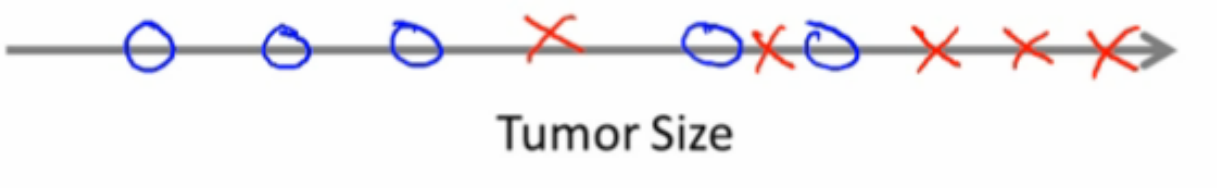
original: 24-bit RGB color



reduced to 4-bits (16 colors)

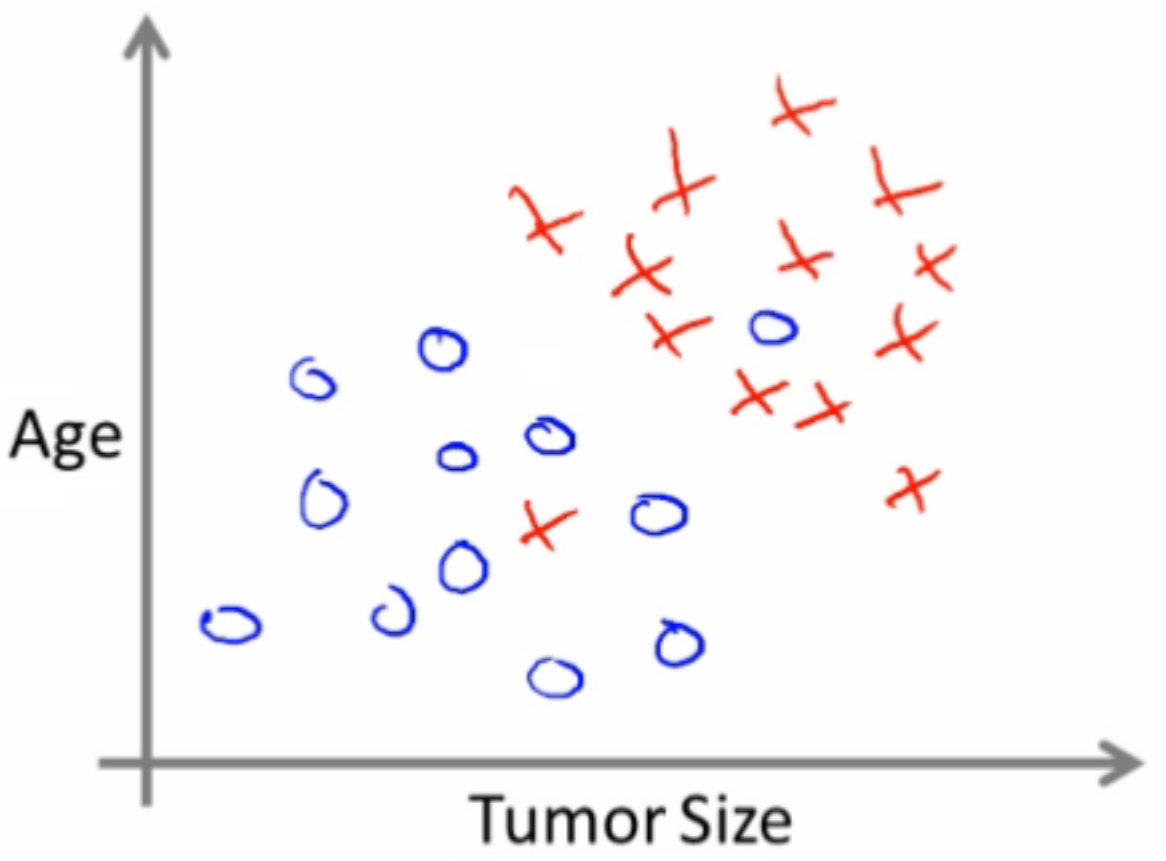
Classification

- Given labeled data (e.g., class labels), predict the class membership of a new data point.
  - e.g., is the sensor data of earth vibrations indicative of an earthquake or a bomb test?
  - Is the tumor malignant or benign?



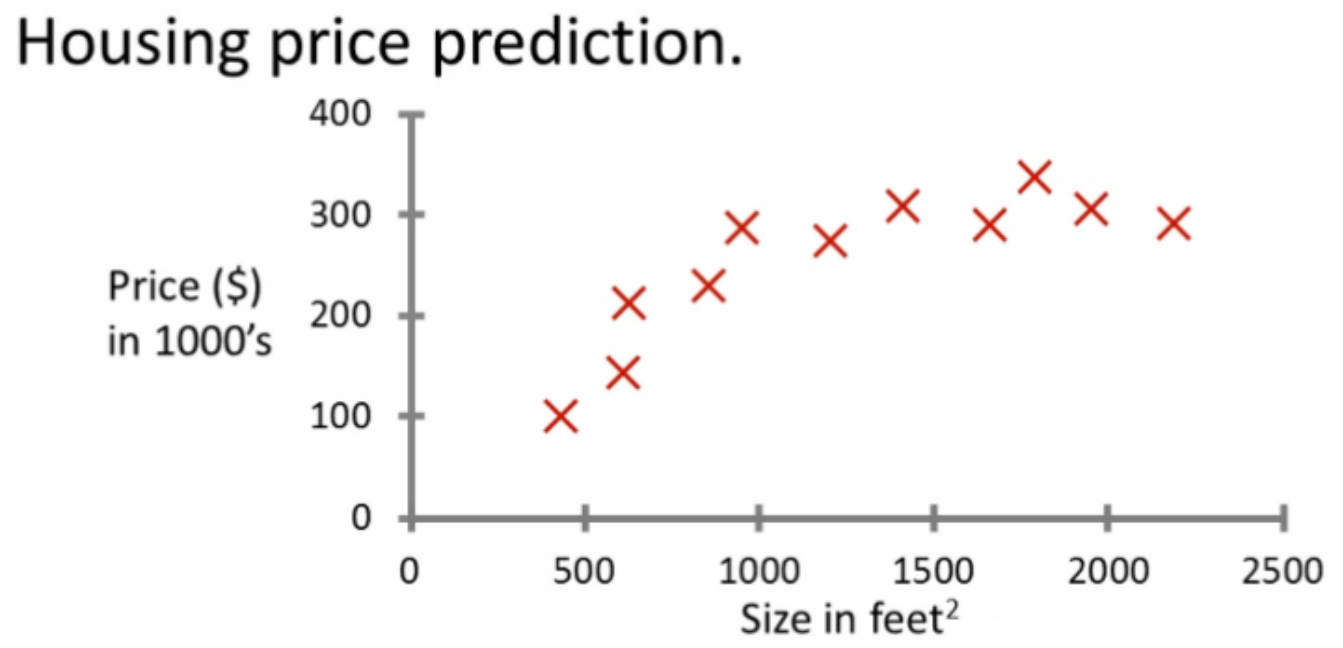
Credit for this and several of the following graphics:
http://holehouse.org/mlclass/01_02_Introduction_regression_analysis_and_gr.html

Classification with multiple attributes



Regression (Prediction)

- Given the values associated with points, predict the associated value of a new point
  - e.g., when will the value of Facebook stock hit $300/share?
  - What will a 750 sq ft house sell for?



Credit for this and following graphics: http://holehouse.org/mlclass/01_02_Introduction_regression_analysis_and_gr.html

ML Techniques

- Clustering:
  - k-Means, vector quantization
- Dimensionality Reduction:
  - PCA, SOM, autoencoder
- Classification, regression, prediction:
  - parametric models: Bayes', MAP, ML (and EM algorithm)
  - non-parametric models: k-NN, SVM, decision trees, BPNN

Learning Schemes

- supervised learning
  - learns from labeled examples
- reinforcement learning
  - learns how to act based on observations
- unsupervised learning
  - groups data into categories; no labeled examples available

Supervised Learning

- given examples ($x, f(x)$), learn ($x, h(x)$) where $h(x) \approx f(x)$
  - regression: $h(x)$ is a continuous value
  - classification: $h(x)$ is a class label
- goal: minimize error on training (and testing) data
- learning can be:
  - incremental: update hypothesis when new example seen
  - batch: hypotheses generated after considering a training set of n examples
- methods: BPNN, SVM, Bayes, k-NN, decision trees, boosting

Linear Regression

- simplest approach to estimating a function
- given data pairs (**x**,y)
- x may be a vector; assume y is a linear function of **x**
- estimate $y \approx h_{\boldsymbol{\theta}}(\mathbf{x})$ where $h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \ldots$
- need to determine $\boldsymbol{\theta}$ so as to minimize error, i.e., difference between $h_{\boldsymbol{\theta}}(\mathbf{x})$ and y
- typically use Least Mean Squares (LMS)
- can often solve by linear algebra

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(x_i) - y_i)^2$$

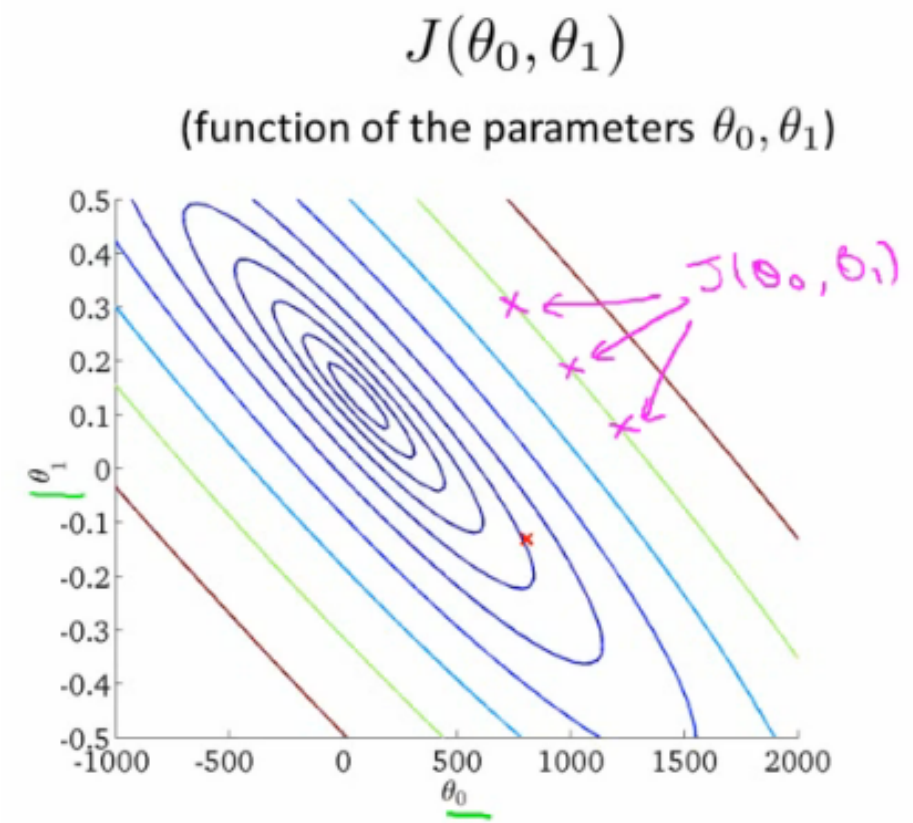
Polynomial Regression

- form of linear regression on which hypothesis is assumed to be a polynomial function of x
- given data pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$
- for simplicity, we'll assume x is a scalar
- assume y is a degree-*d* polynomial function of x
- estimate $y \approx h_w(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d$
- again, can solve for **w** with linear algebra

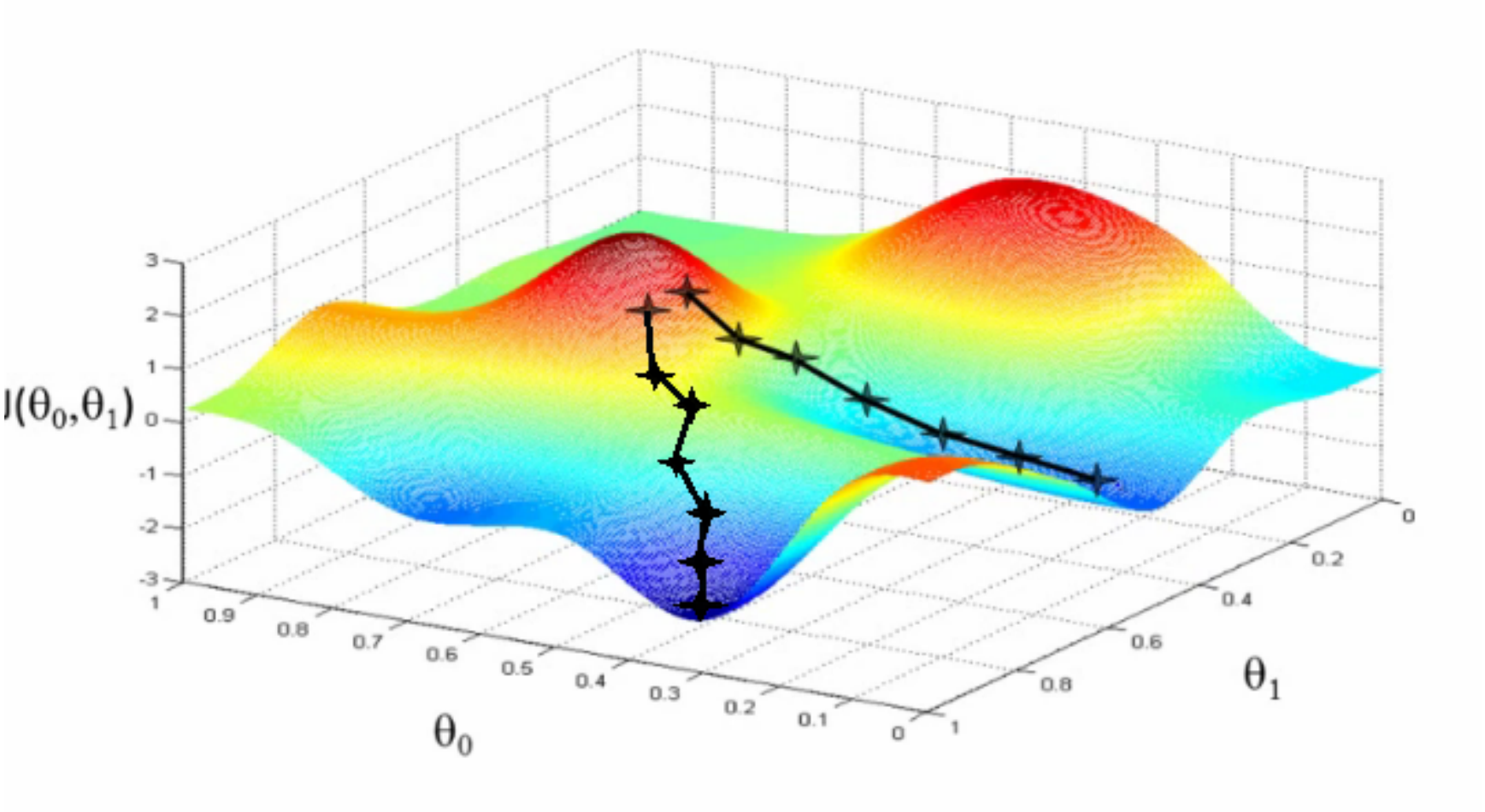
Minimizing Cost Function

- Sometimes, linear algebra solutions are too complex to solve for the parameters
- Instead, can try to minimize cost function by "navigating" the error contours
- Consider case of two parameters

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

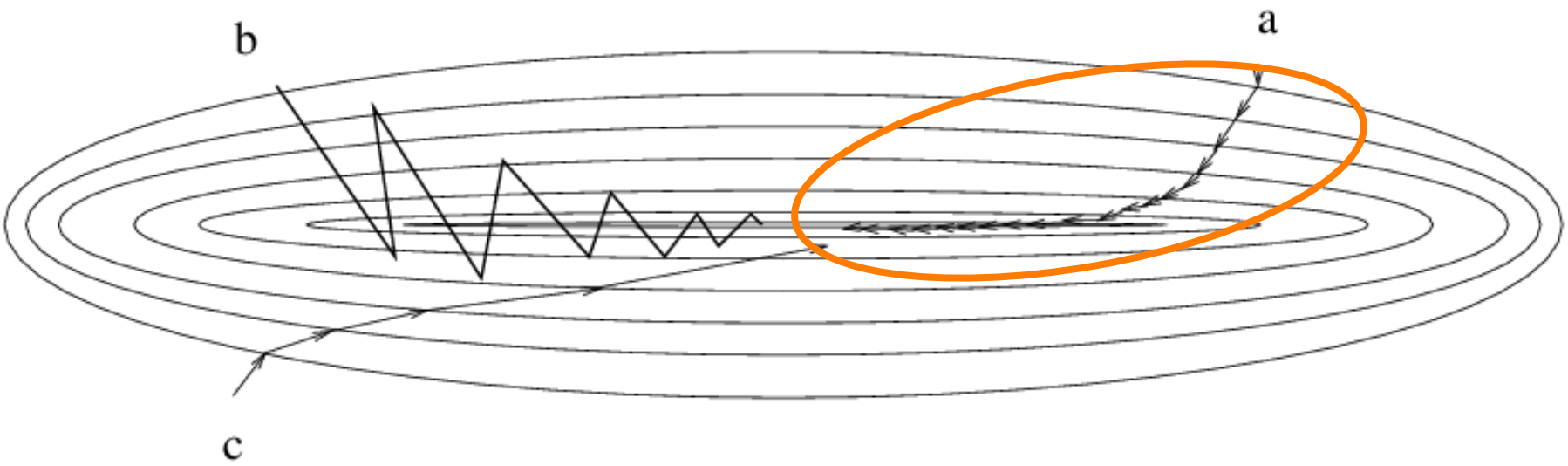


Gradient Descent



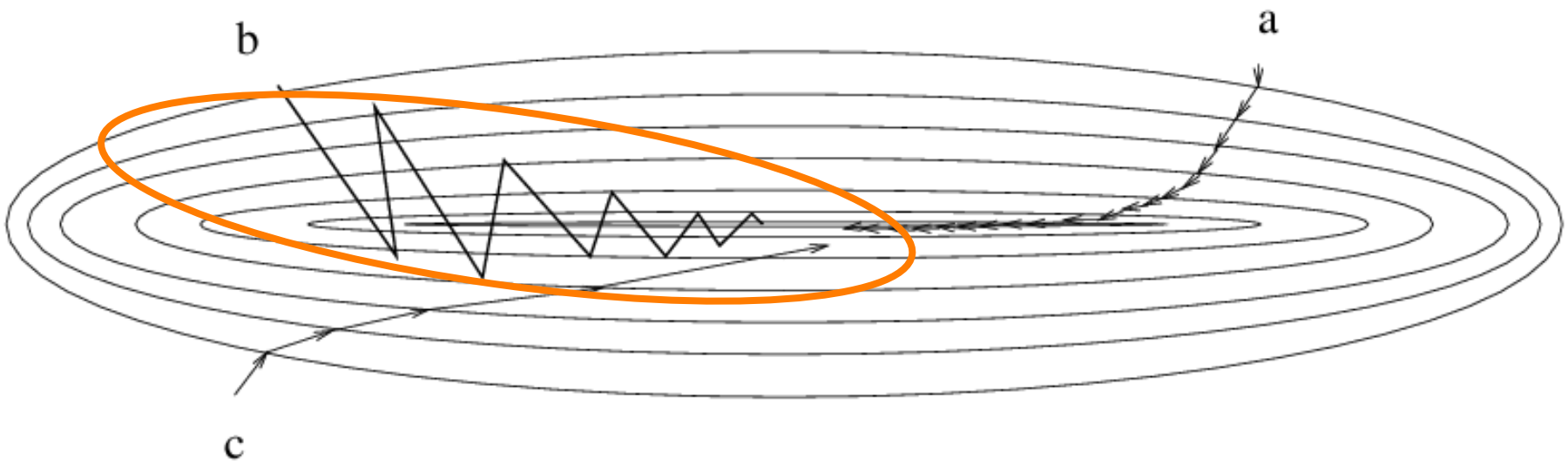
Where you start can lead to different solutions

Gradient Descent Too Slow



- gradient descent needs infinitesimal steps
  => this could take forever

Gradient Descent Too Fast

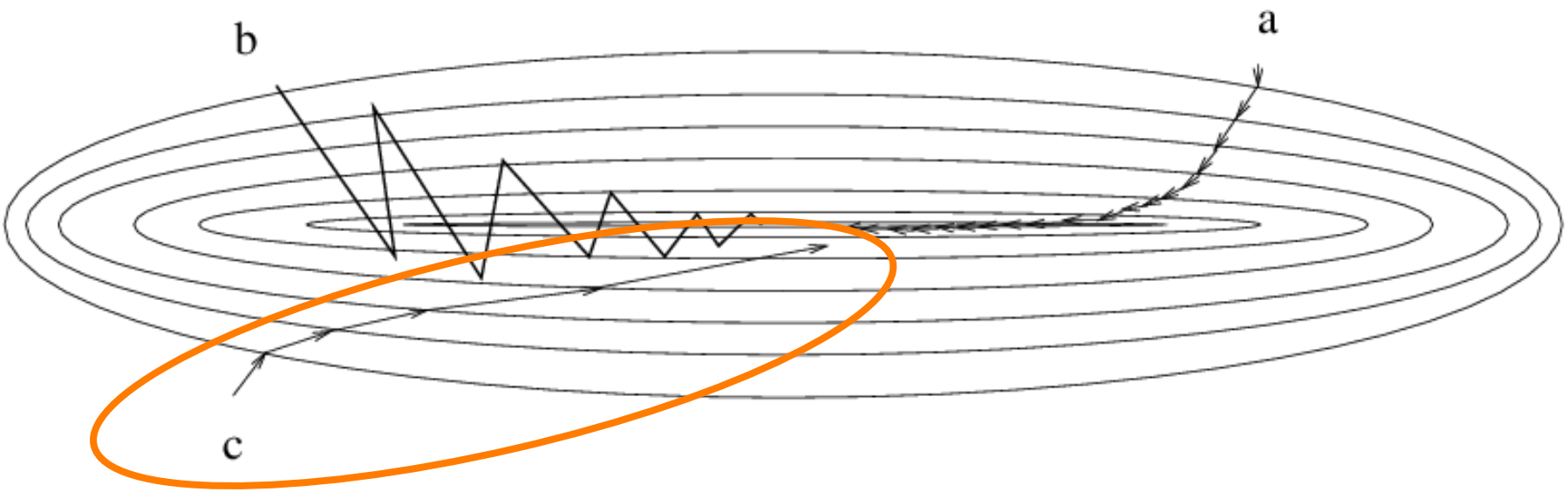


- in practice, use a larger step size -- *learning rate*: α
  => but too large a value leads to oscillations in error space

Local Minima

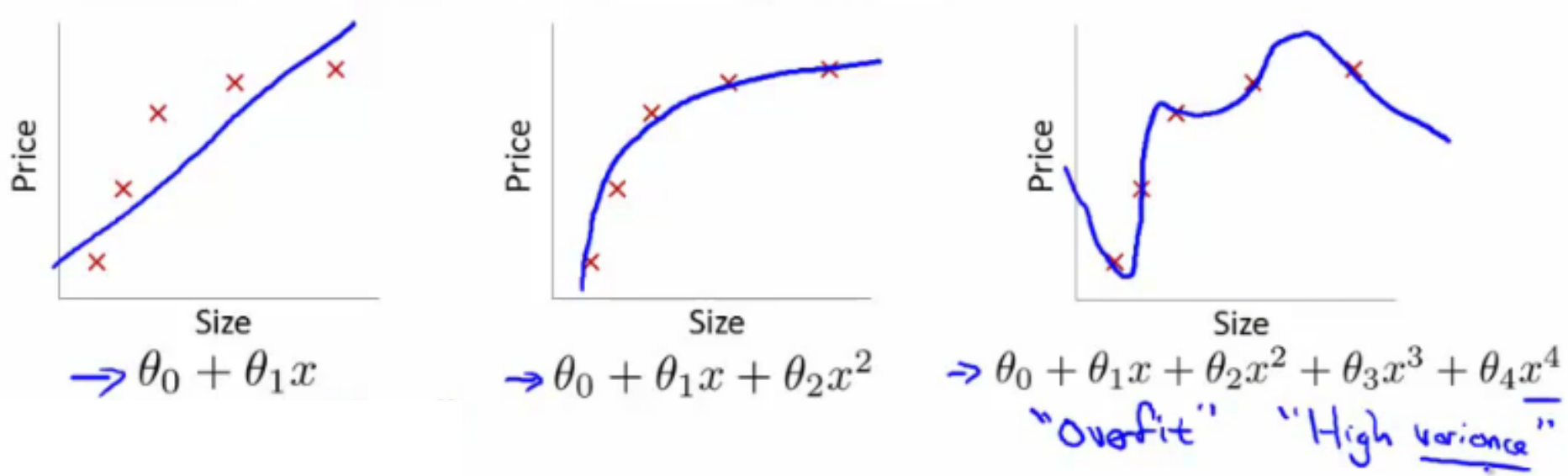
- in general, error surface is full of hills and valleys
- gradient descent can get trapped in local minimum
- solutions?

Gradient Descent just right



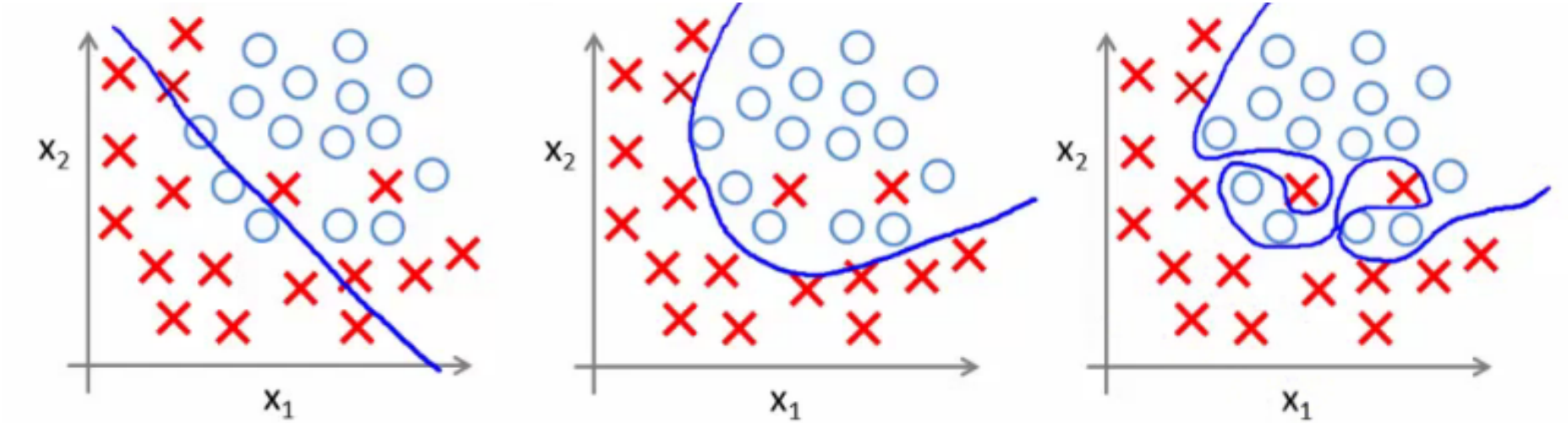
- Dampen oscillations by introducing a momentum factor γ
  - e.g., $\Delta\theta(t + 1) = \alpha\nabla J + \gamma\, \Delta\theta(t)$
- Bonus: This also helps avoid getting trapped in local minima

Fitting the Data with Linear or Polynomial Regression



(credit: Stanford ML course)

Same issue with logistic regression



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

(g = sigmoid function)

UNDERFITTING
(high bias)

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

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

OVERFITTING
(high variance)

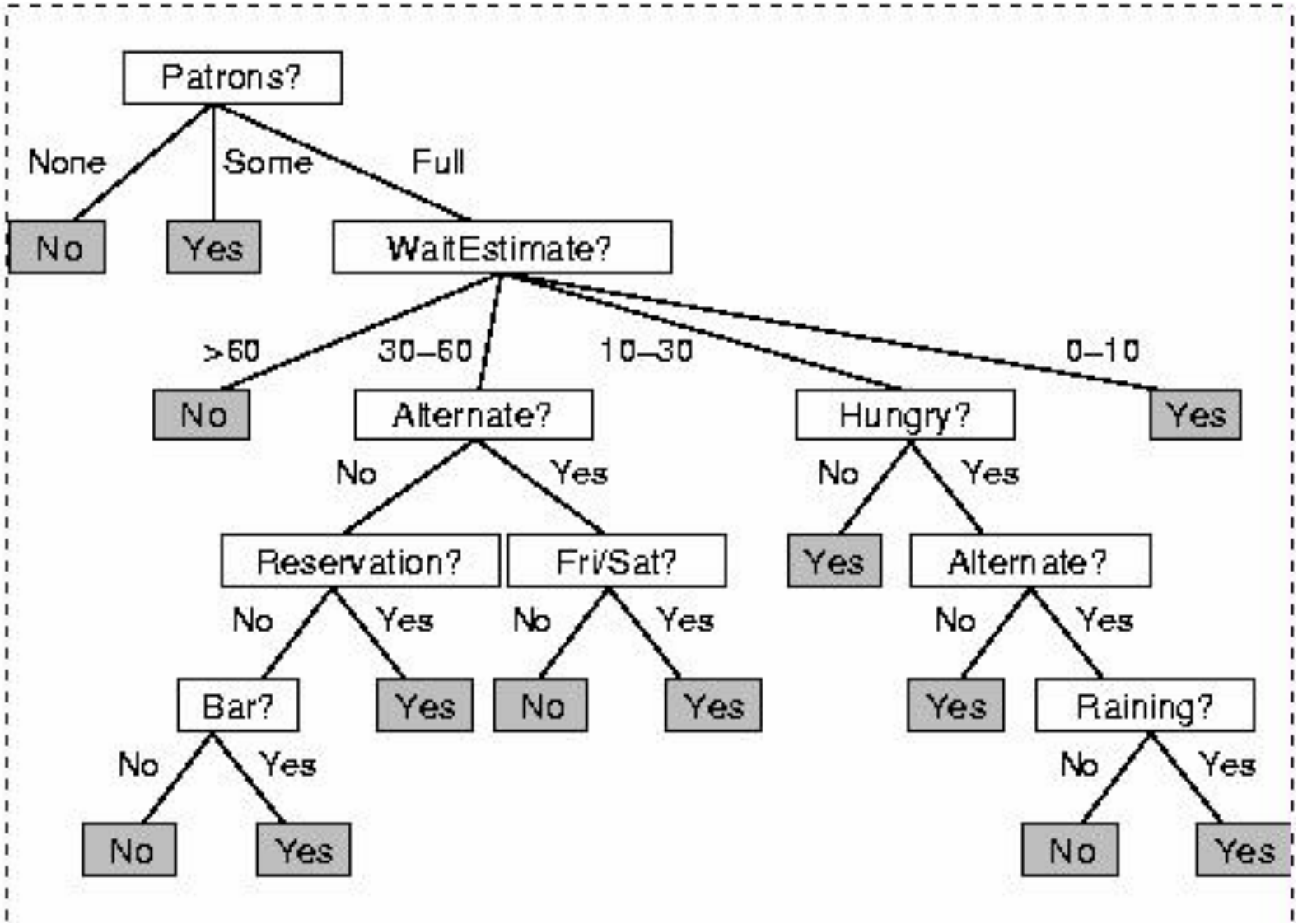
(credit: Stanford ML course)

Overlearning

- don't want learning system to be overtrained
- Solution: partition data examples into three sets:

Data Set	Used for	Analogy with

Supervised Learning of a Decision Tree



Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	$$$	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	$	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	$$$	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	$	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	$	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$$$	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	$	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Burger*	*30–60*	*Yes*

How about an LTA?

- one path for each training example
- going down path tests each attribute in turn
- leaf assigned the classification of example

Fixing the 2^n problem

- form more compact representation
  - look for redundancy of attributes
  - but not always possible, e.g.,
    - parity function
    - majority function

Types of Learning
Based on Information Available

- supervised learning:
  - given labeled sets of (x,y)
  - goal: minimize error on training (and testing) data
  - e.g., BPNN, SVM, Bayes, k-NN, decision trees, boosting
- reinforcement learning:
  - given actions and outcomes
  - goal: learn to maximize reward over long term
  - e.g., TD, Q-learning
- unsupervised learning:
  - only given x
  - goal: learn associations within (groupings of) the data
  - e.g., k-means, EM, PCA, SOM

Next Class

- Inductive Learning (Ch 18.3)