

Machine Learning Crash Course



Computer Vision
James Hays

Slides: Isabelle Guyon,
Erik Sudderth,
Mark Johnson,
Derek Hoiem



Robert Borowicz



Bao Vu



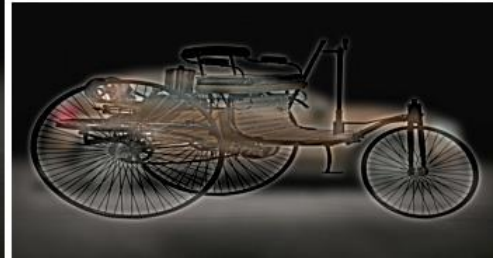
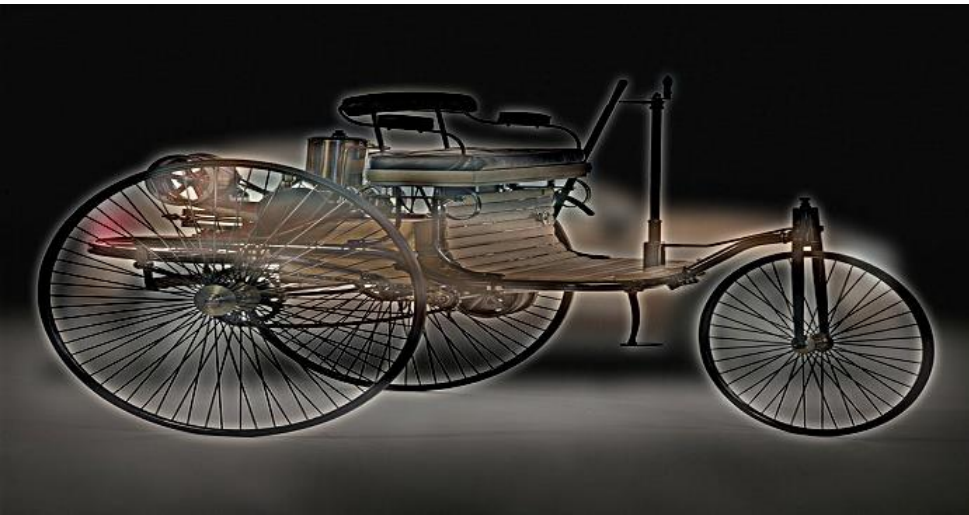
Creston Bunch



Pranathi Tupakula



Murali Raghu Babu Balusu



Sandhya Sridhar



M S Suraj

Machine Learning Crash Course



Computer Vision
James Hays

Slides: Isabelle Guyon,
Erik Sudderth,
Mark Johnson,
Derek Hoiem

Machine Learning Problems

Supervised Learning

Unsupervised Learning

Discrete
Continuous

classification or categorization

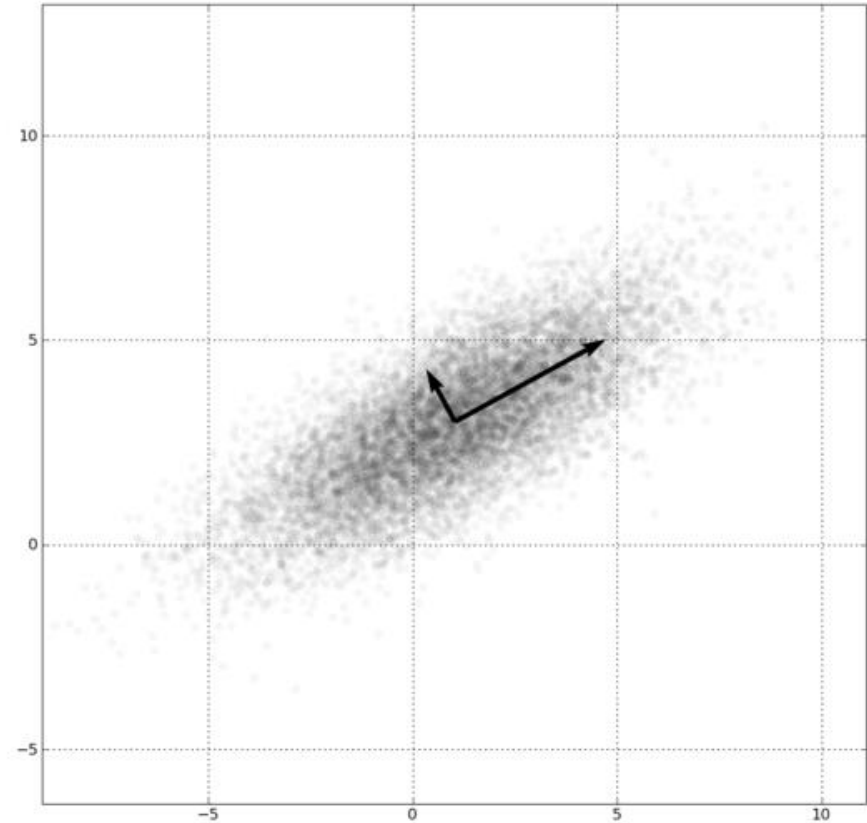
clustering

regression

dimensionality reduction

Dimensionality Reduction

- **PCA, ICA, LLE, Isomap**
- PCA is the most important technique to know. It takes advantage of correlations in data dimensions to produce the best possible lower dimensional representation based on linear projections (minimizes reconstruction error).
- PCA should be used for dimensionality reduction, not for discovering patterns or making predictions. Don't try to assign semantic meaning to the bases.



Machine Learning Problems

Supervised Learning

Unsupervised Learning

Discrete
Continuous

classification or
categorization

clustering

regression

dimensionality
reduction

How do we cluster?

- K-means
 - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
 - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
 - Estimate modes of pdf
- Spectral clustering
 - Split the nodes in a graph based on assigned links with similarity weights

Clustering for Summarization

Goal: cluster to minimize variance in data given clusters

- Preserve information

$$\mathbf{c}^*, \boldsymbol{\delta}^* = \underset{\mathbf{c}, \boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} (\mathbf{c}_i - \mathbf{x}_j)^2$$

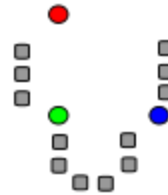
Cluster center

Data

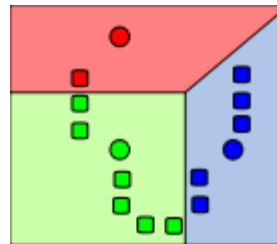
Whether x_j is assigned to c_i

K-means algorithm

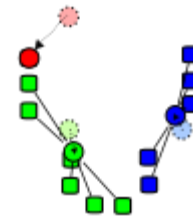
1. Randomly select K centers



2. Assign each point to nearest center

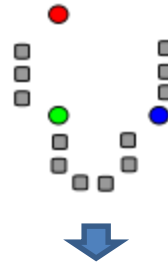


3. Compute new center (mean) for each cluster



K-means algorithm

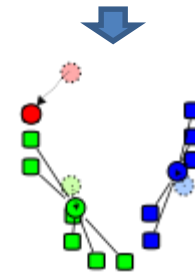
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2

Machine Learning Problems

Supervised Learning

Unsupervised Learning

Discrete

classification or
categorization

clustering

Continuous

regression

dimensionality
reduction

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

$f(\text{apple image}) = \text{“apple”}$

$f(\text{tomato image}) = \text{“tomato”}$

$f(\text{cow image}) = \text{“cow”}$

The machine learning framework

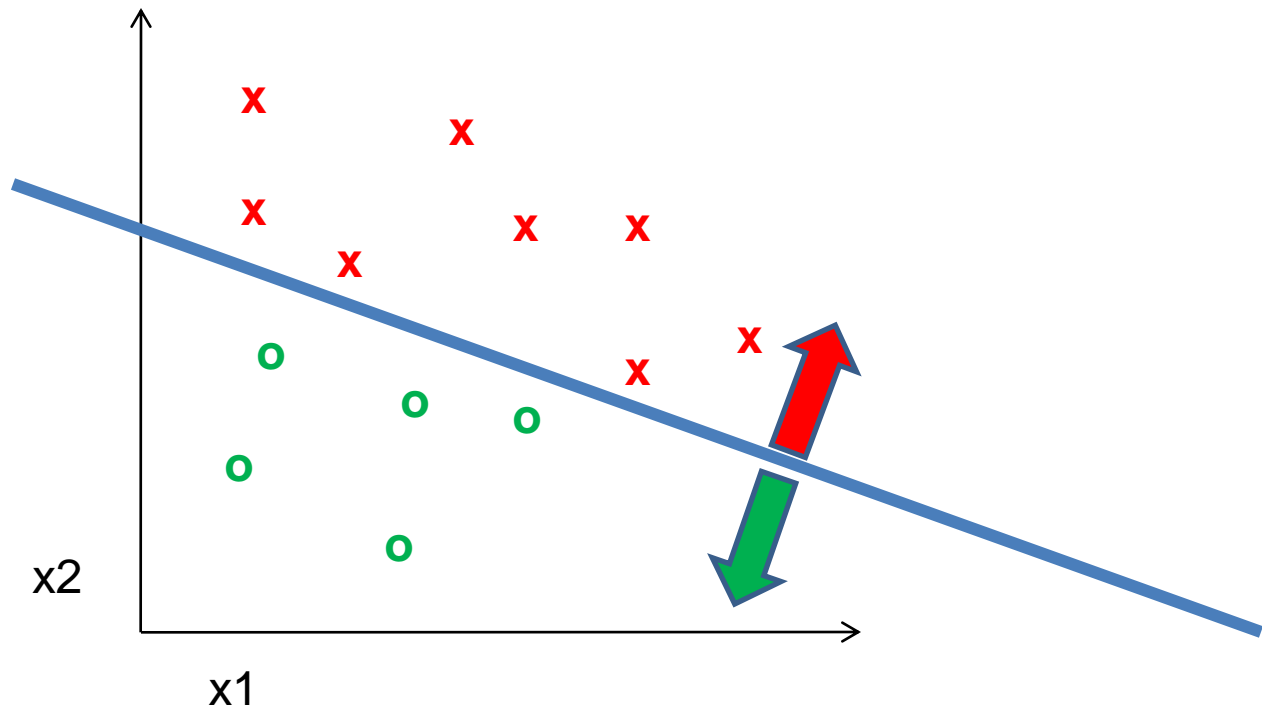
$$y = f(\mathbf{x})$$

output prediction function Image feature

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* \mathbf{x} and output the predicted value $y = f(\mathbf{x})$

Learning a classifier

Given some set of features with corresponding labels, learn a function to predict the labels from the features



Steps

Training

Training Images

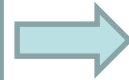


Image Features



Training



Learned model

Training Labels



Testing



Test Image

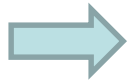
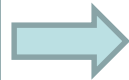


Image Features



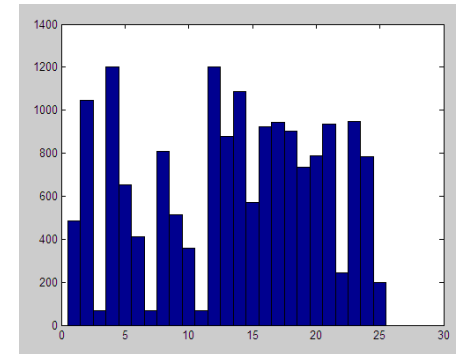
Learned model



Prediction

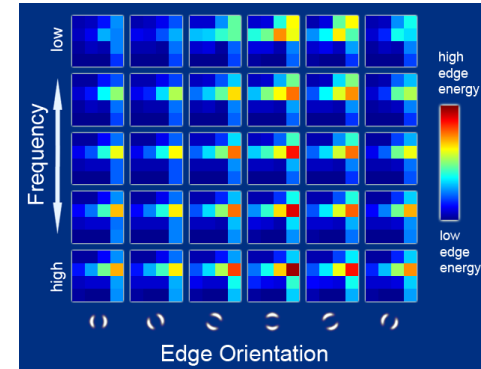
Features

- Raw pixels



- Histograms

- GIST descriptors



- ...

One way to think about it...

- Training labels dictate that two examples are the same or different, in some sense
- Features and distance measures define visual similarity
- Classifiers try to learn weights or parameters for features and distance measures so that visual similarity predicts label similarity

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Deep Convolutional Network
- Etc.

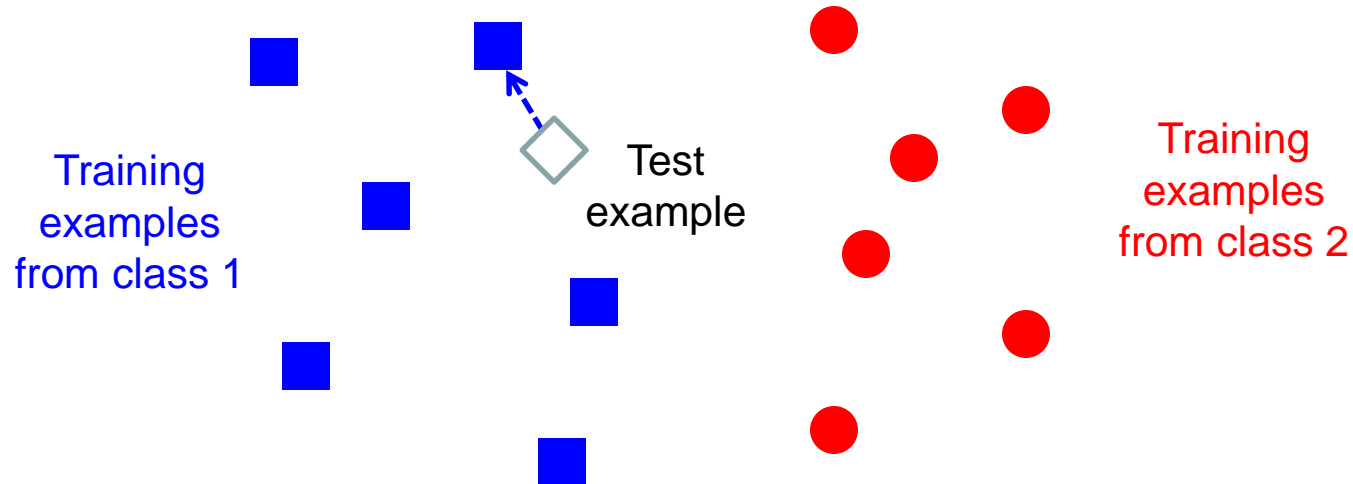
Which is the best one?

Claim:

The decision to *use* machine learning is more important than the choice of a *particular* learning method.

*Deep learning seems to be an exception to this, at the moment, probably because it is learning the feature representation.

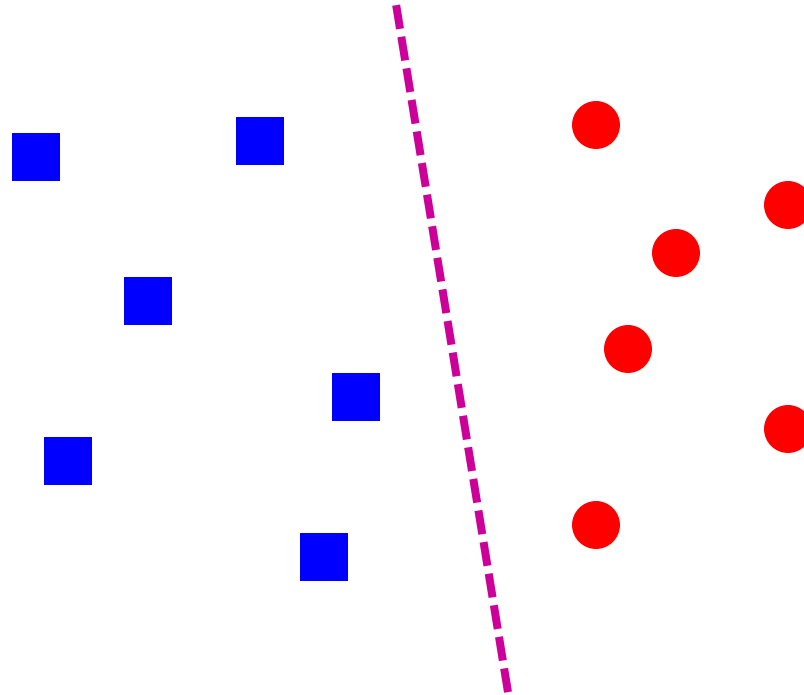
Classifiers: Nearest neighbor



$f(\mathbf{x}) = \text{label of the training example nearest to } \mathbf{x}$

- All we need is a distance function for our inputs
- No training required!

Classifiers: Linear



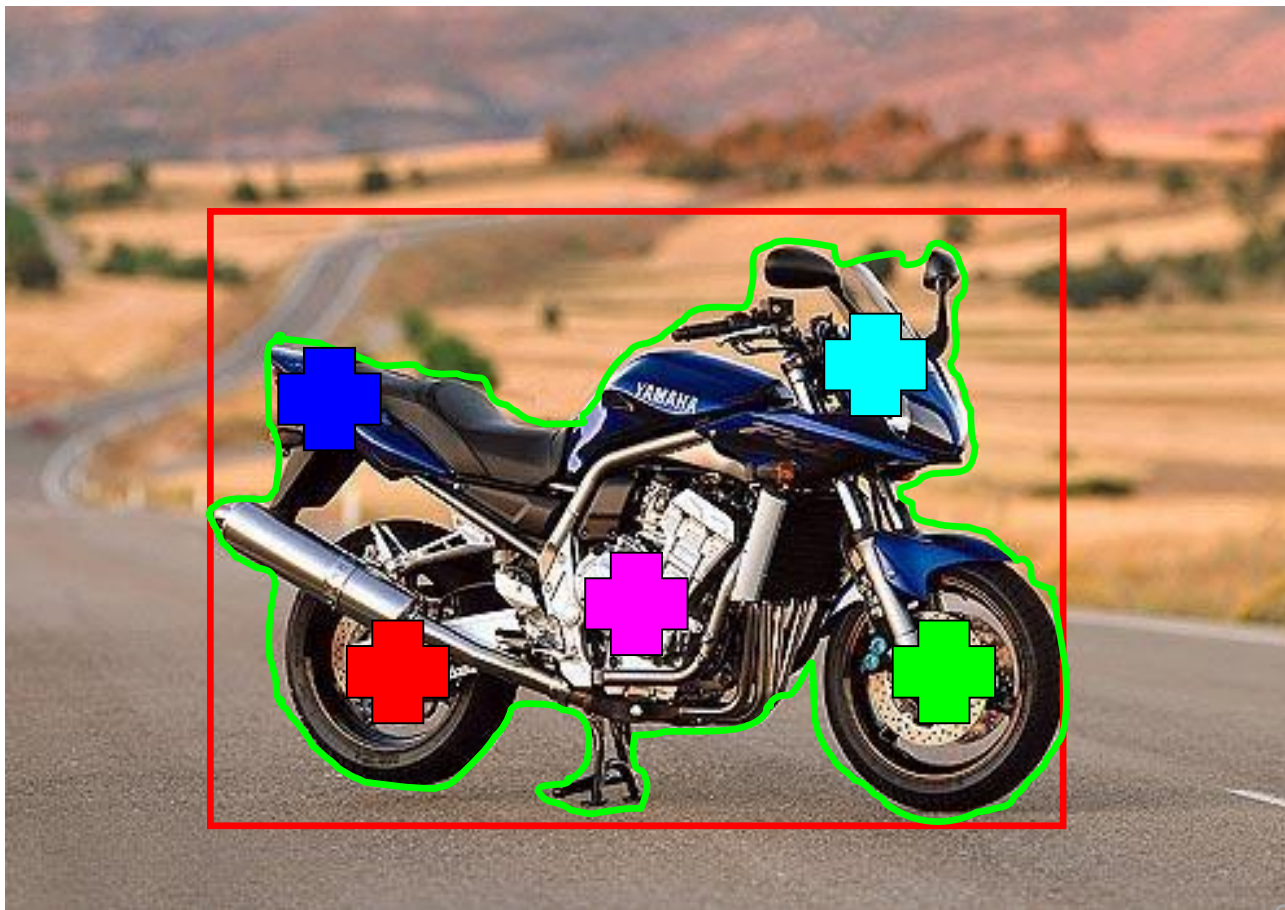
- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Recognition task and supervision

- Images in the training set must be annotated with the “correct answer” that the model is expected to produce

Contains a motorbike



Spectrum of supervision

Less

More



Unsupervised



“Weakly” supervised



Fully supervised

Definition depends on task

Generalization



Training set (labels known)



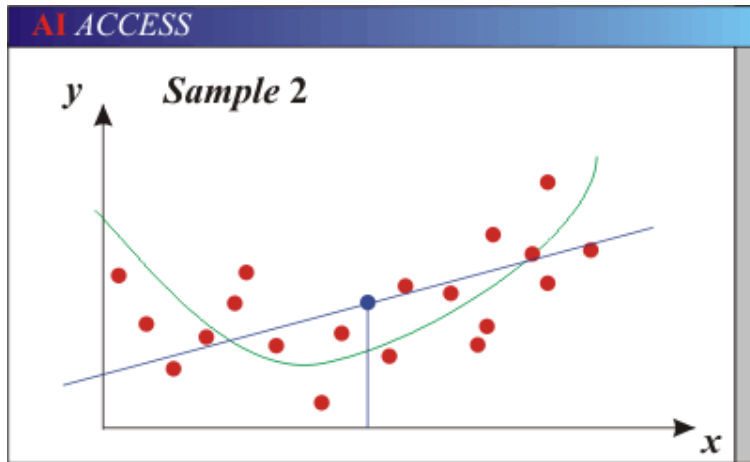
Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

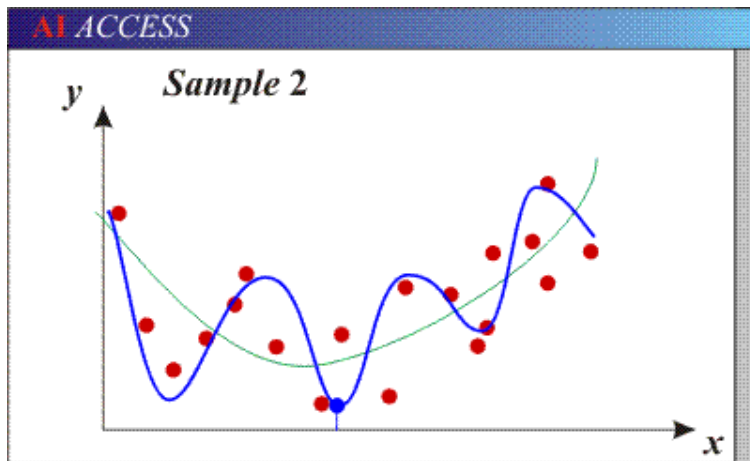
Generalization

- Components of generalization error
 - **Bias:** how much the average model over all training sets differ from the true model?
 - Error due to inaccurate assumptions/simplifications made by the model.
 - **Variance:** how much models estimated from different training sets differ from each other.
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
 - High bias (few degrees of freedom) and low variance
 - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
 - Low bias (many degrees of freedom) and high variance
 - Low training error and high test error

Bias-Variance Trade-off



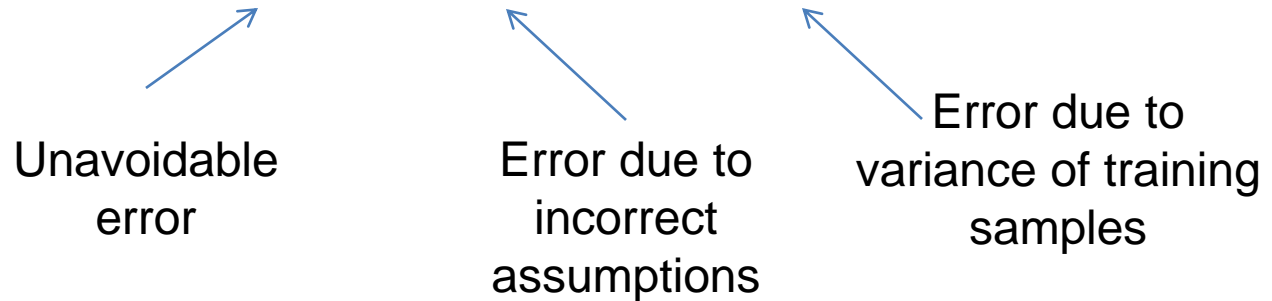
- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Bias-Variance Trade-off

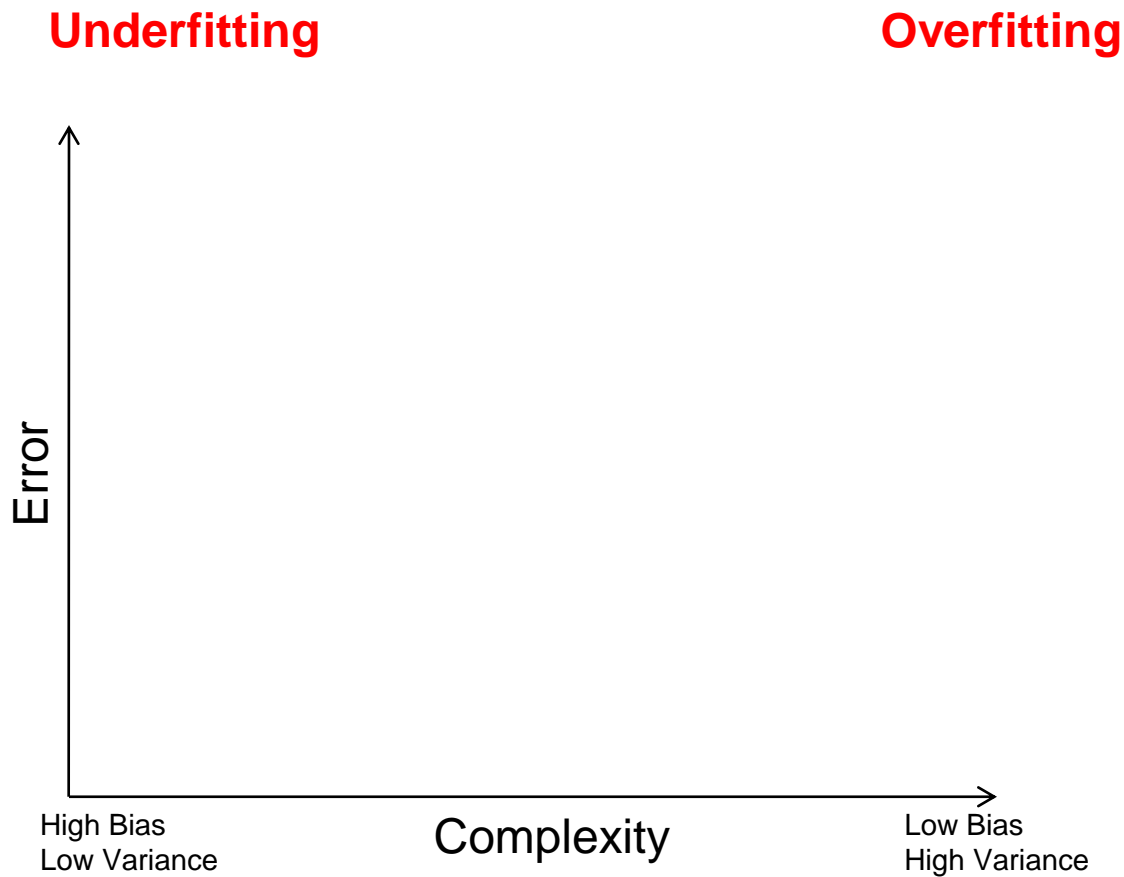
$$E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$



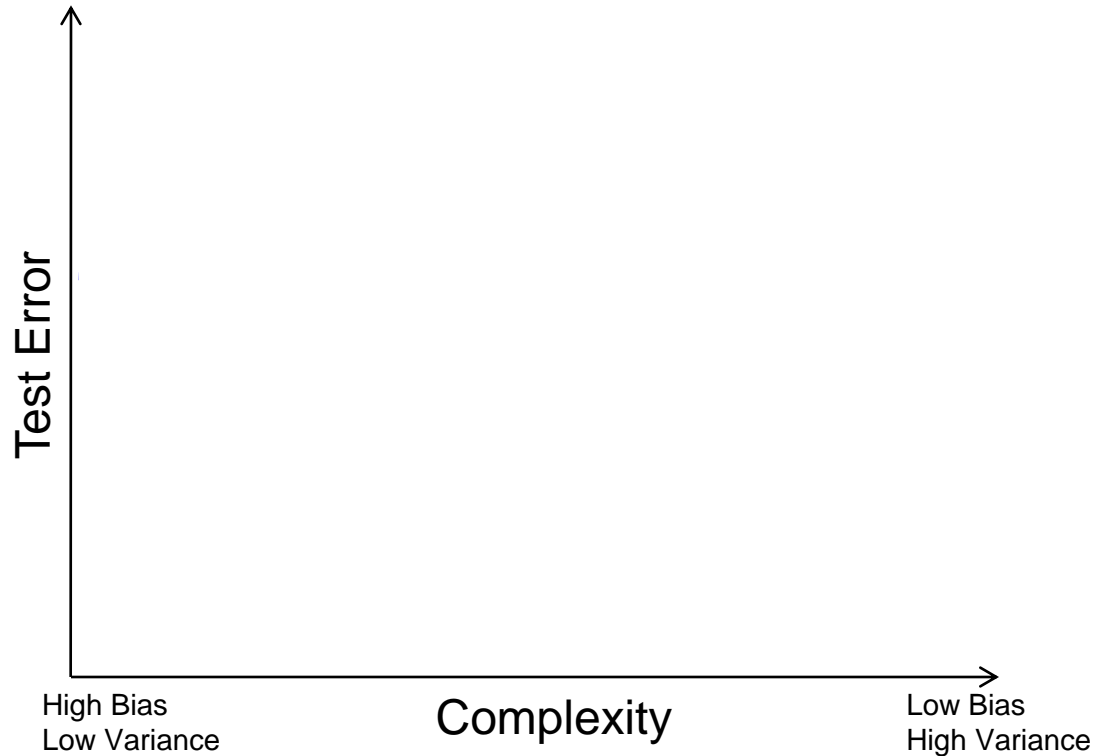
See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

• <http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

Bias-variance tradeoff

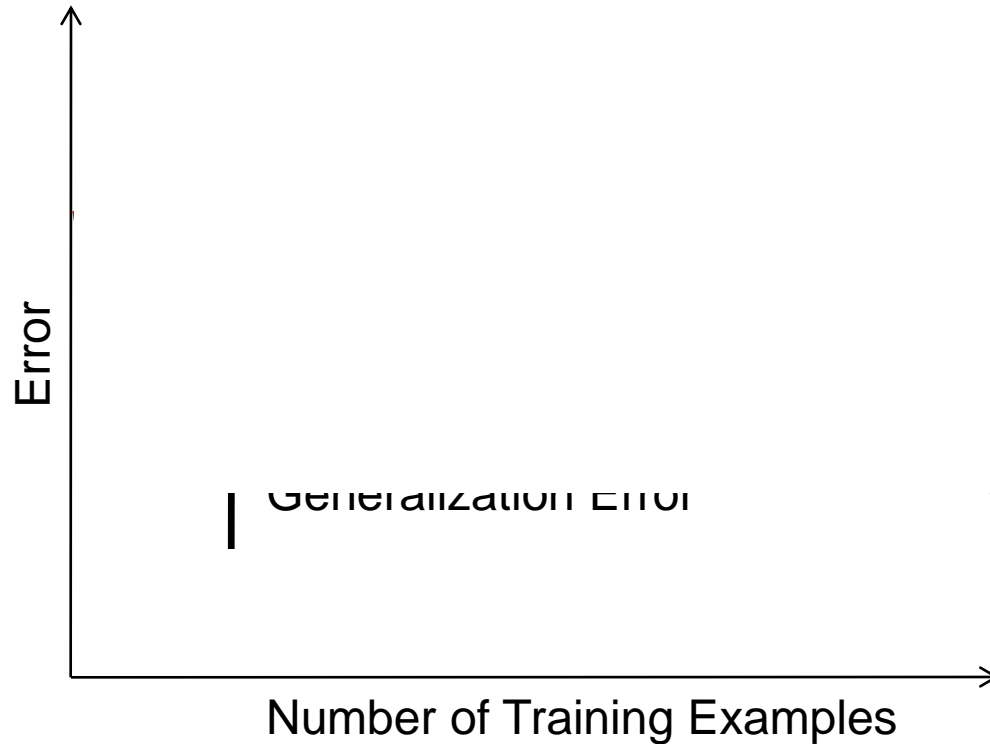


Bias-variance tradeoff



Effect of Training Size

Fixed prediction model



Remember...

- No classifier is inherently better than any other: you need to make assumptions to generalize
- Three kinds of error
 - Inherent: unavoidable
 - Bias: due to over-simplifications
 - Variance: due to inability to perfectly estimate parameters from limited data



How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data

Very brief tour of some classifiers

- **K-nearest neighbor**
- **SVM**
- **Boosted Decision Trees**
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- RBMs
- Etc.

Generative vs. Discriminative Classifiers

Generative Models

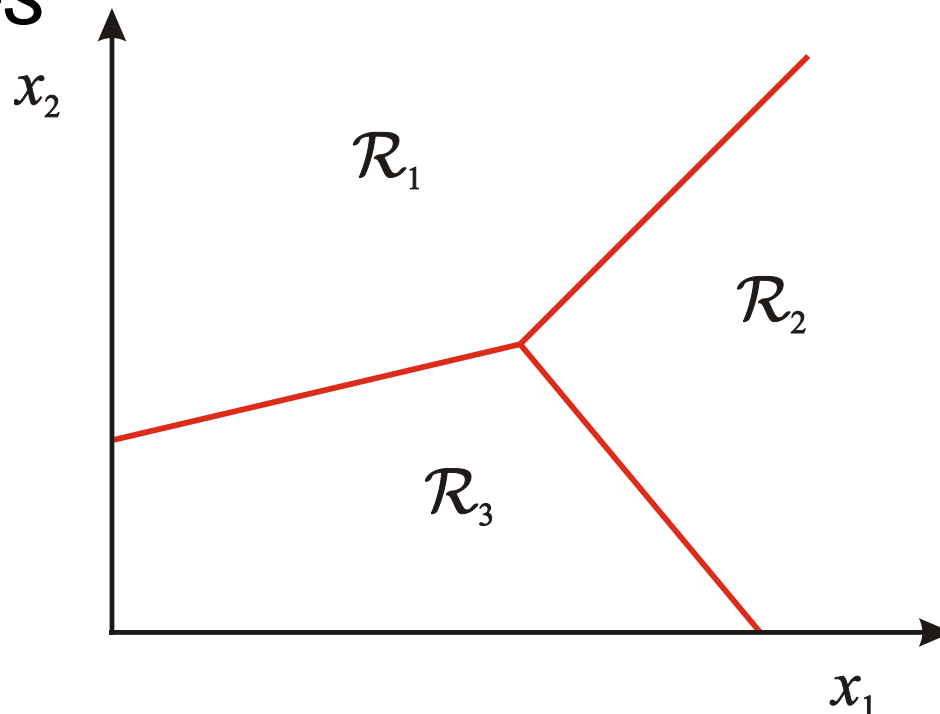
- Represent both the data and the labels
- Often, makes use of conditional independence and priors
- Examples
 - Naïve Bayes classifier
 - Bayesian network
- Models of data may apply to future prediction problems

Discriminative Models

- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
 - Logistic regression
 - SVM
 - Boosted decision trees
- Often easier to predict a label from the data than to model the data

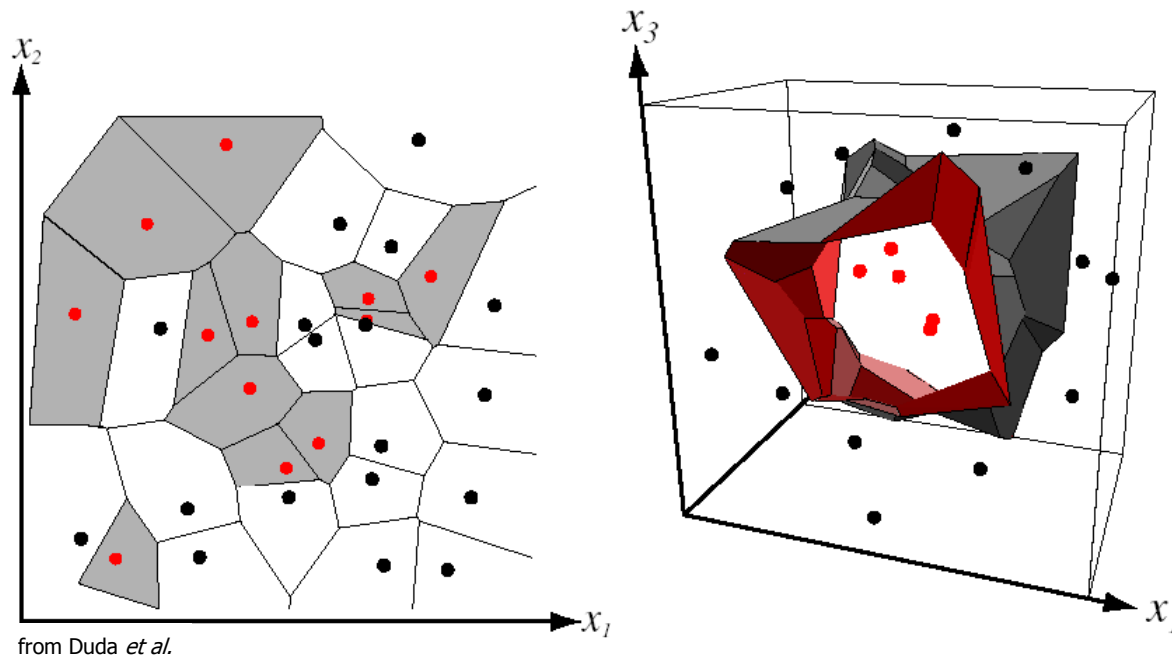
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



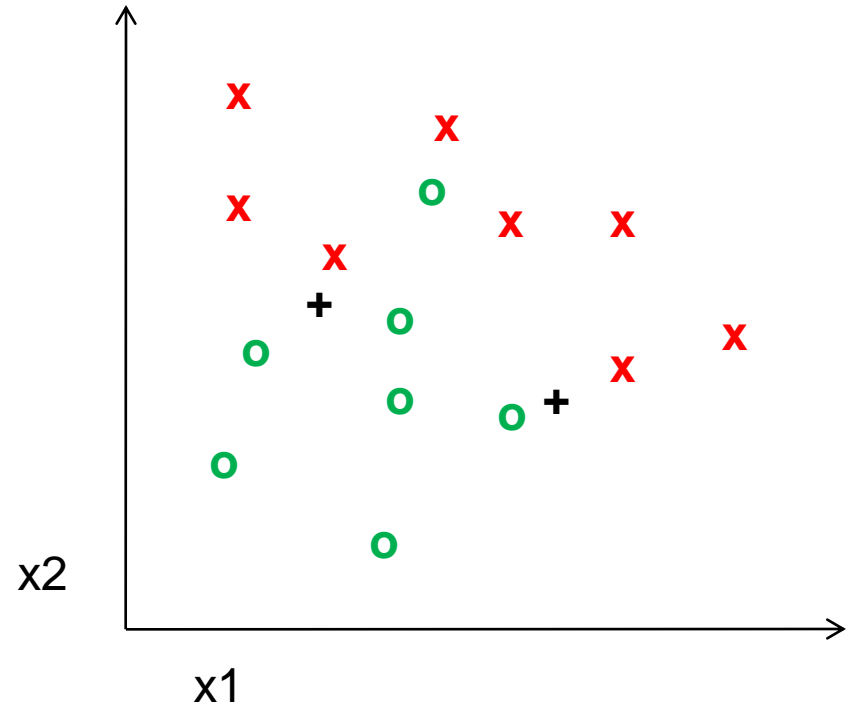
Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point

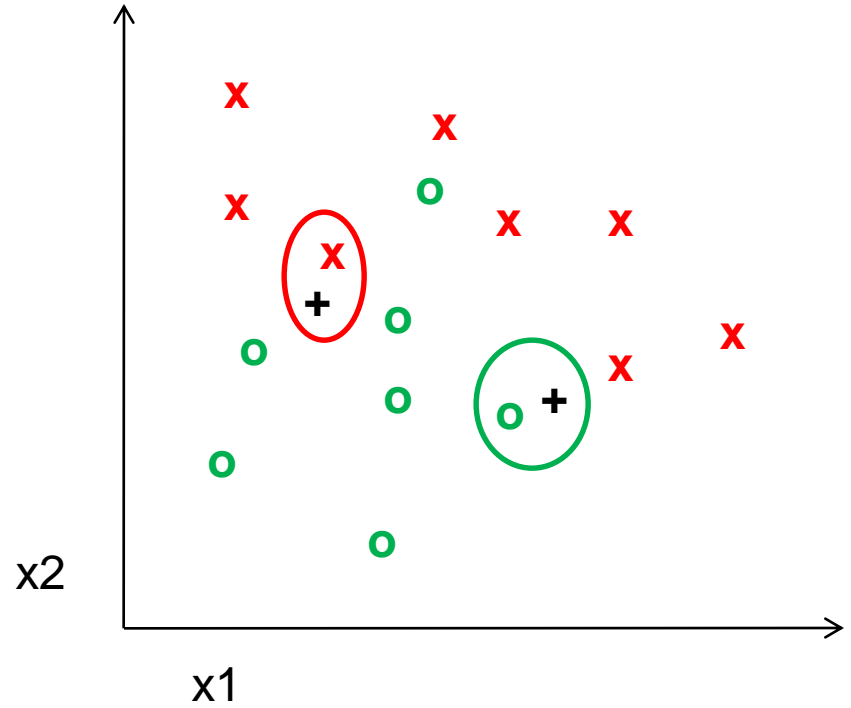


Voronoi partitioning of feature space
for two-category 2D and 3D data

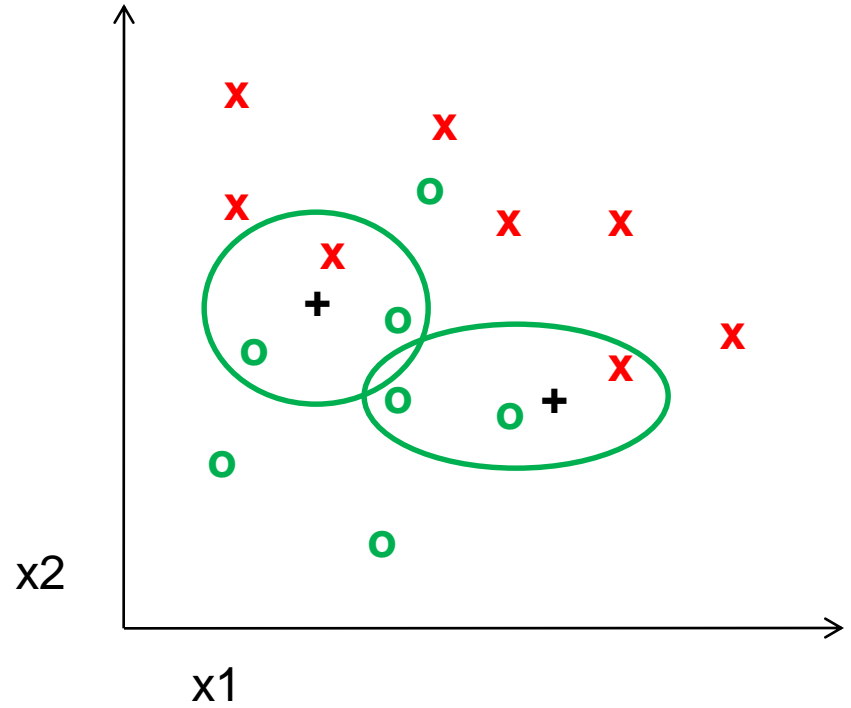
K-nearest neighbor



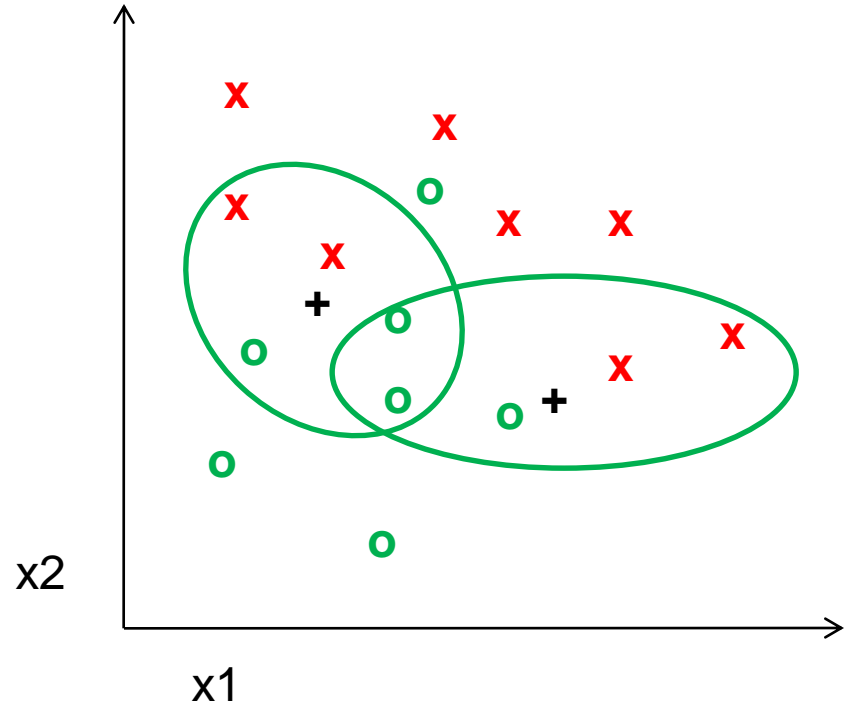
1-nearest neighbor



3-nearest neighbor



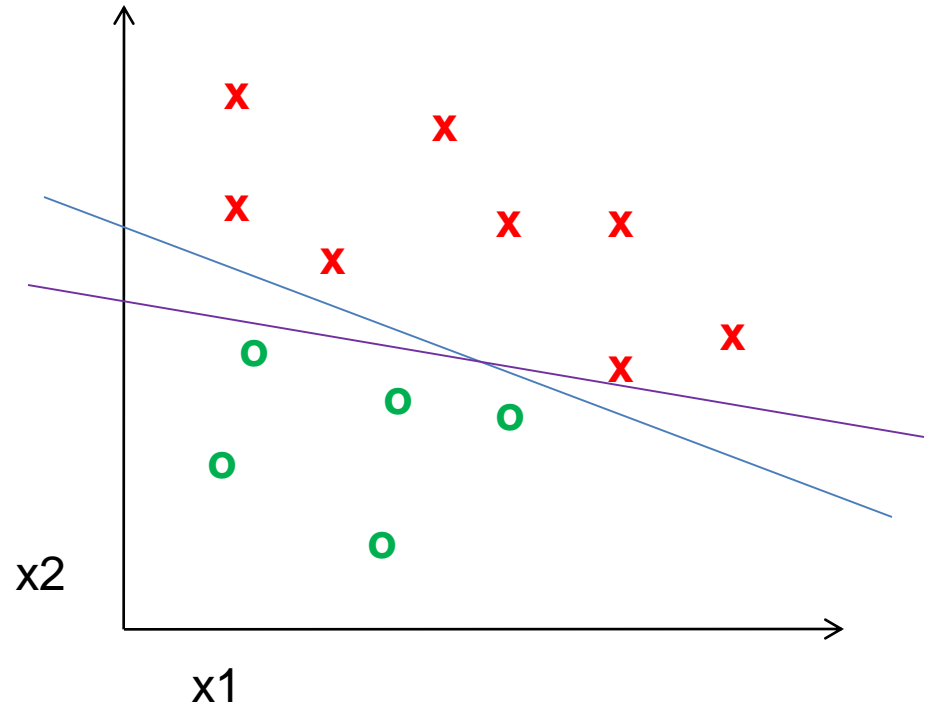
5-nearest neighbor



Using K-NN

- Simple, a good one to try first
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error

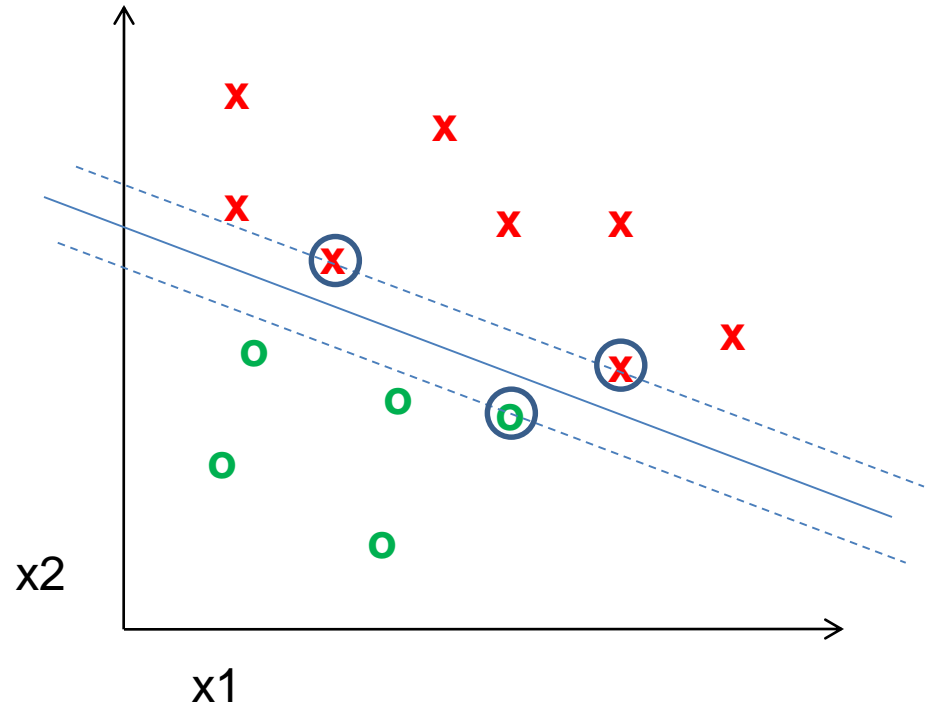
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

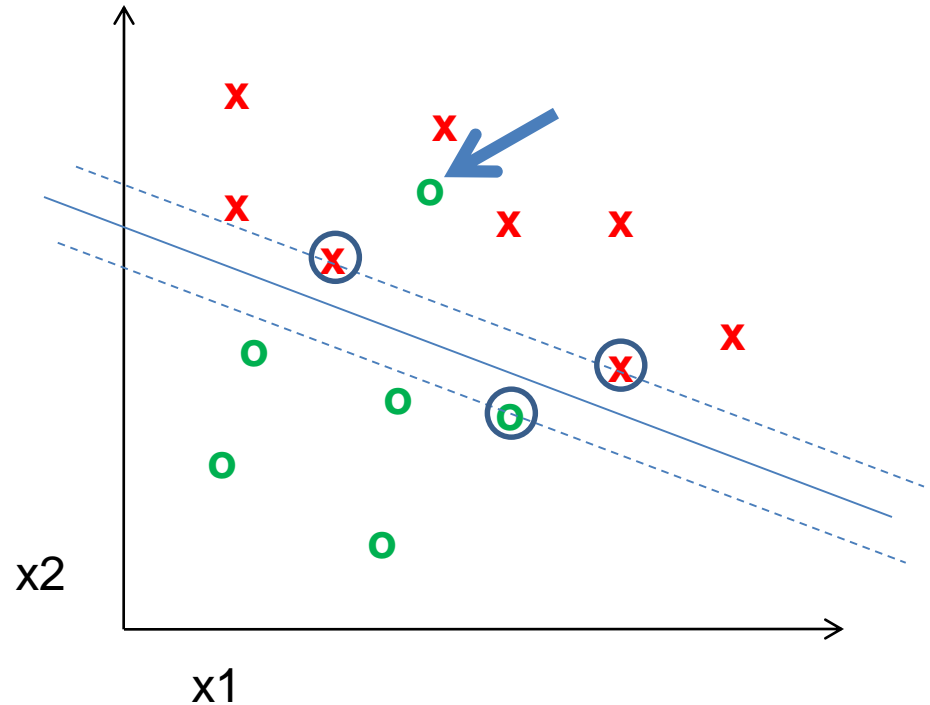
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
 - Training: learn an SVM for each class vs. the others
 - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
 - Training: learn an SVM for each pair of classes
 - Testing: each learned SVM “votes” for a class to assign to the test example

SVMs: Pros and cons

- Pros

- Many publicly available SVM packages:
<http://www.kernel-machines.org/software>
- Kernel-based framework is very powerful, flexible
- SVMs work very well in practice, even with very small training sample sizes

- Cons

- No “direct” multi-class SVM, must combine two-class SVMs
- Computation, memory
 - During training time, must compute matrix of kernel values for every pair of examples
 - Learning can take a very long time for large-scale problems

What to remember about classifiers

- No free lunch: machine learning algorithms are tools, not dogmas
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Making decisions about data

- 3 important design decisions:
 - 1) What data do I use?
 - 2) How do I represent my data (what feature)?
 - 3) What classifier / regressor / machine learning tool do I use?
- These are in decreasing order of importance
- Deep learning addresses 2 and 3 simultaneously (and blurs the boundary between them).
- You can take the representation from deep learning and use it with any classifier.