Machine Learning (*cont.*)

Applied to Computer Vision

Adín Ramírez Rivera
adin@ic.unicamp.br

Artificial Intelligence (AI)
2nd. Semester 2016

Adapted from J. Hays, I. Guyon, E. Sudderth, M. Johnson, D. Hoiem

Machine Learning Problems

	Supervised Learning	Unsupervised Learning
Discrete	Classification or categorization	Clustering
Continuous	Regression	Dimensionality Reduction

How do we generate the clusters?

- k-means
  - Iteratively re assign each point to the closest cluster depending on their center
- Agglomerative Clustering
  - Each point is its own cluster, and iteratively we mix the closest ones
- Mean-shift Clustering
  - Estimate the modes of the PDF
- Spectral Clustering
  - Divide the graph nodes based on the edges' weights
- The lower in the list, the algorithms tend to transtively group the points (even when they are not close in the feature space)

Machine Learning Problems

	Supervised Learning	Unsupervised Learning
Discrete	Classification or categorization	Clustering
Continuous	Regression	Dimensionality Reduction

Framework

- Apply a prediction function to the representation of the image to obtain a desired output
- For example

$f\Big($  $\Big) =$ apple

$f\Big($  $\Big) =$ tomato

$f\Big($  $\Big) =$ cow

Function f

Prediction function

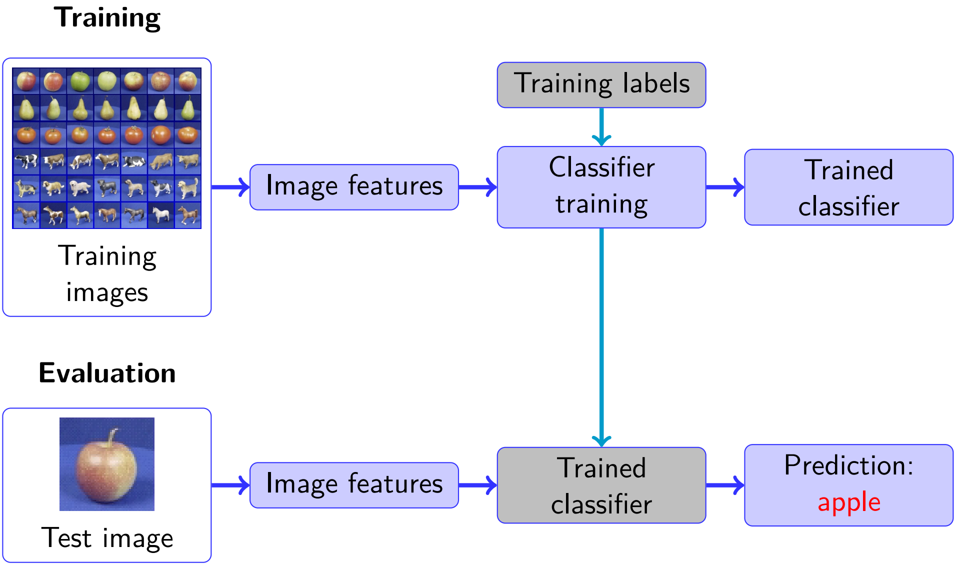
$$\boxed{y} = \boxed{f}(\boxed{\mathbf{x}})$$

Features

Output

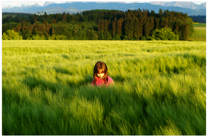
- Training: given a set of label training data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, we estimate the prediction function f through the minimization of the error in the training set
- Evaluation: apply f to each element of the testing set (not yet seen) $\mathbf{x}$ and obtain the prediction $y = f(\mathbf{x})$

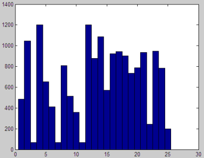
Steps

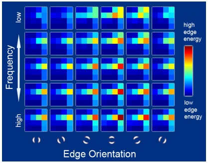

Training
Training images
Image features
Training labels
Classifier training
Trained classifier
Evaluation
Test image
Image features
Trained classifier
Prediction: apple

Features

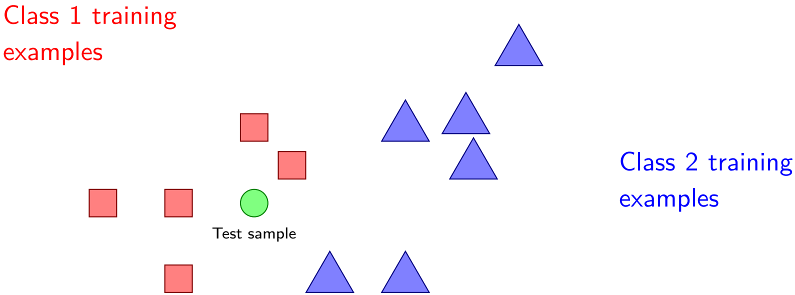
- Pixel values (raw)
- Histograms
- SIFT Descriptors
- HOG Descriptors
- GIST Descriptors
- etc.





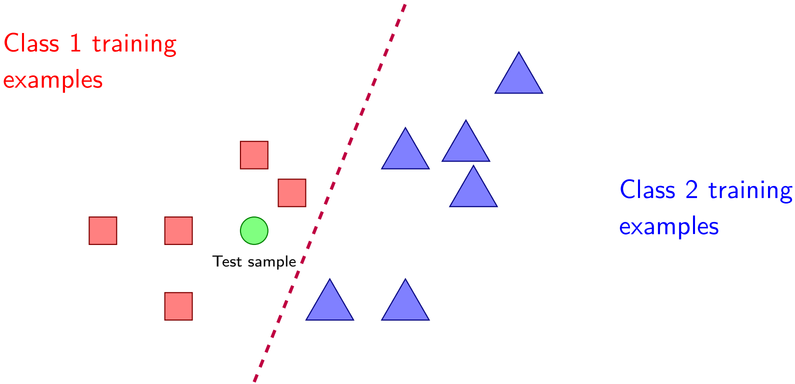


Nearest Neighbor



- $f(x) =$ label of the closest sample to x
- All we need are distance functions for the samples
- No need for training (there is no model)

Linear



- Find a lineal function that separates the classes

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

- Classify according to the side of the barrier in which the samples lies

Many Classifiers

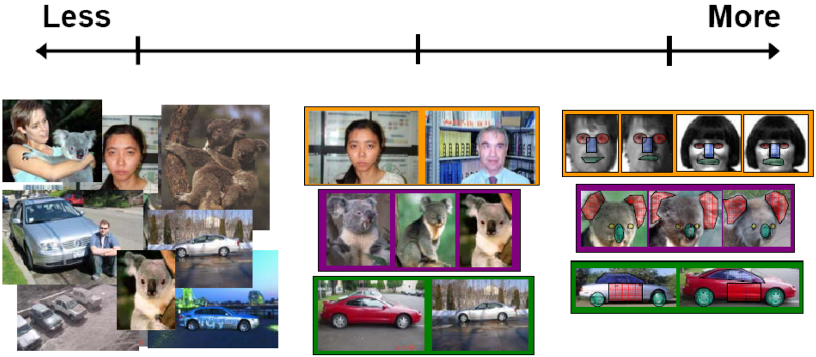
- Support Vector Machines (SVM)
- Neural Networks (hot topic)
- Naïve Bayes
- Bayesian Networks
- Logistic regression
- Random forest
- Boosted decision trees
- k-Nearest neighbors
- etc.
- Which one is best?

Recognition and supervision

- Images in the training set need to be labeled with the correct answer
- The model needs to recognize similar shapes

Supervision Spectrum

- Non supervised
- Weak supervised
- Totally supervised



Generalization

- Answers "how good the learned model generalizes to not yet seen data?"

Training set
(known labels)

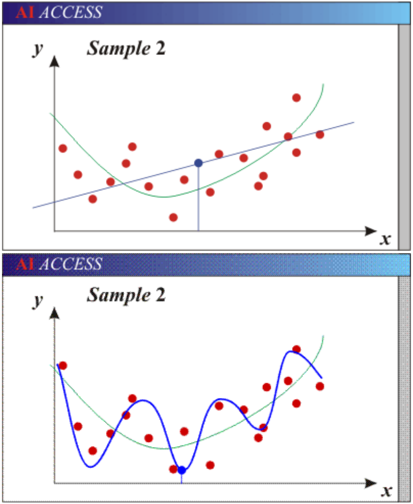
Testing set
(unknown labels)

Generalization

- Components of the generalization error
  - Bias how much does the mean model (from all the training set) differs from the true model
  - Variance how much does the trained models differ among each other when trained with different set of data
- Underfit: the model is too simple to represent all the relevant features of the given class
  - High bias and low variance
  - High training and testing error
- Overfit: the model is too complex and adjusts to the irrelevant features (noise) of the data
  - Low bias and high variance
  - Low training error and high test error

Compromise between variance and bias

- Models with few parameters are imprecise because they have high bias (have no flexibility)
- Models with too many parameters are imprecise because they have high variance (too much sensitivity to samples)



Compromise between variance and bias

- Expected mean square error

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

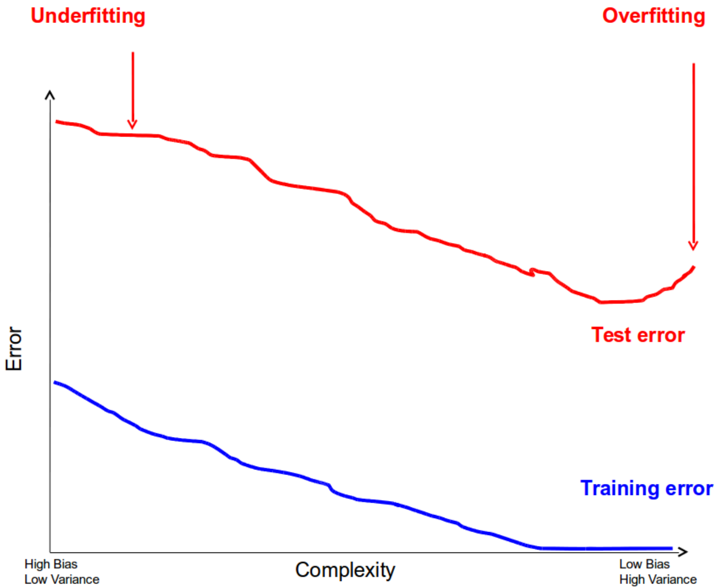
Unavoidable error

Error due to wrong assumptions

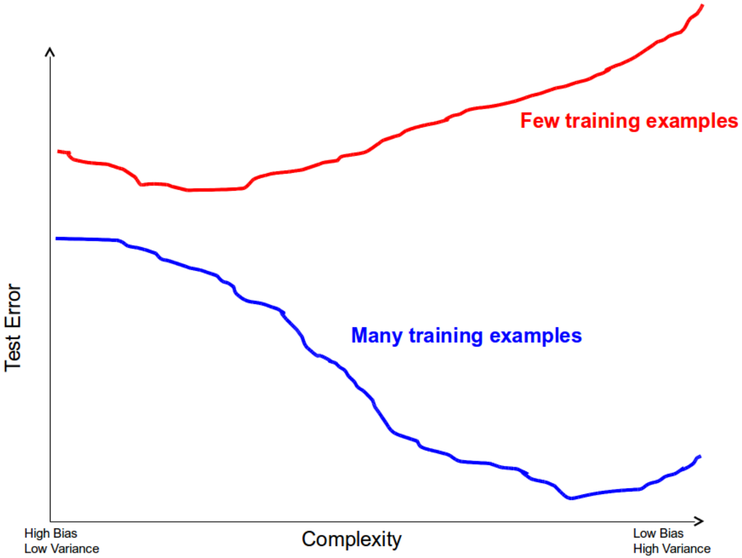
Error due to variance of the training samples

- More details http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf
- Also “Neural Networks,” Bishop

Complexity vs. Error

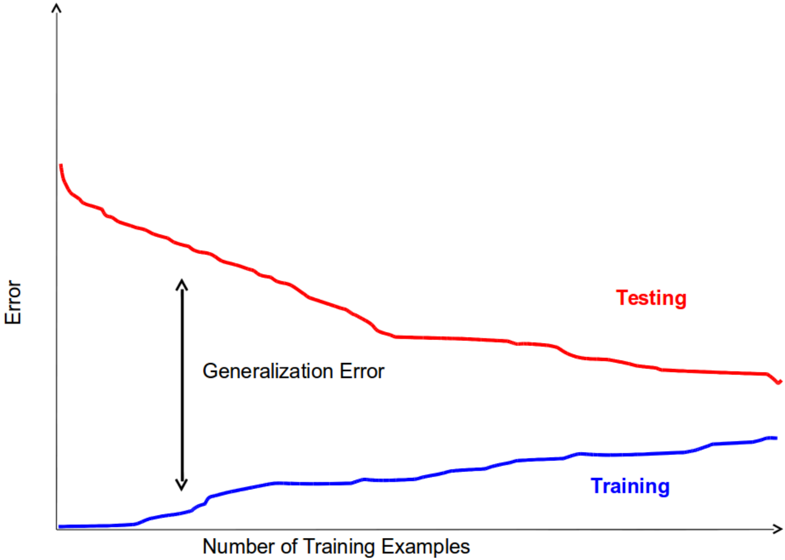


Complexity vs. Test Error



Training Sample Size Effect

Fixed Prediction Model



Key Points

- There is no classifier that is inherintly better than another
  - We made assumptions to generalize
- There is no free lunch!

- Three error types
  - Inherent: can't be avoided
  - Bias: due to over simplification
  - Variance: due to inability to estimate the correct parameters from the data

How to reduce the variance?

- Pick a simple classifier
- Regularize the parameters
- Get more training data

Generative vs. Discriminative

Generative Models

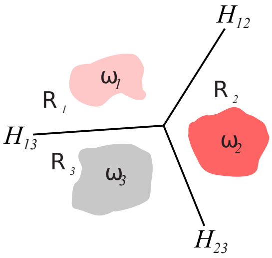
- Represent the data and labels
- Often use conditional independence and priors
- Examples
  - Bayes Naive Classifier
  - Bayesian Networks
- Data models can be applied to future prediction problems

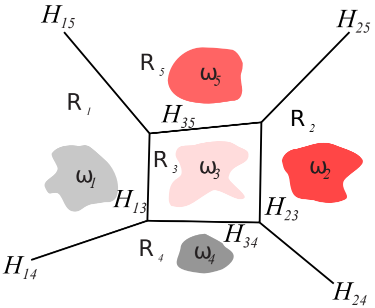
Discriminative Models

- Learn to predict the labels of the data directly
- Often asume a barrier (e.g., lineal)
- Examples
  - Logistic Regression
  - Support Vector Machines
  - Boosted Decision Trees
- Easier to predict a label than to model the data

Clasification

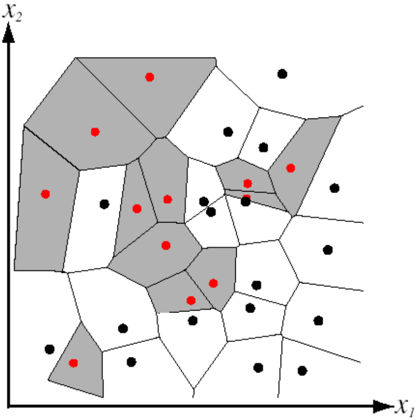
- Assign an input vector to one or more classes
- Any decision rule divides the input space into decision regions separated by decision boundaries

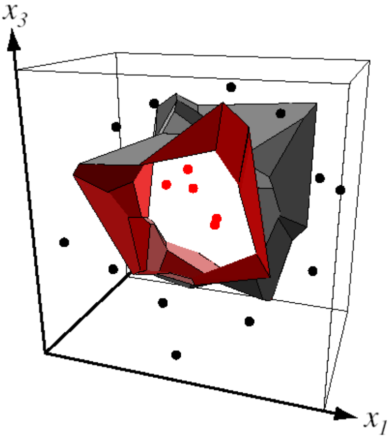




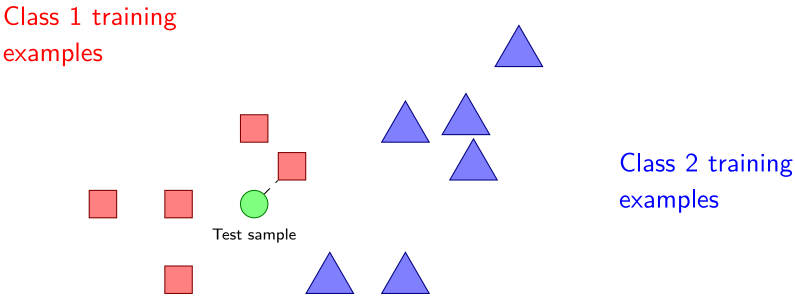
Nearest Neighbor Classifiers

- Assign the label according to the closest training data
- We can partition the space using a Voronoi diagram

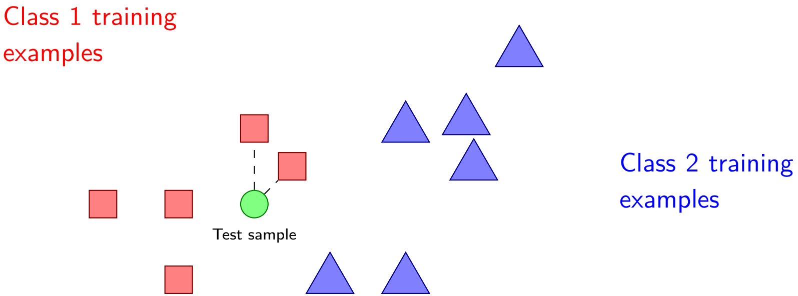




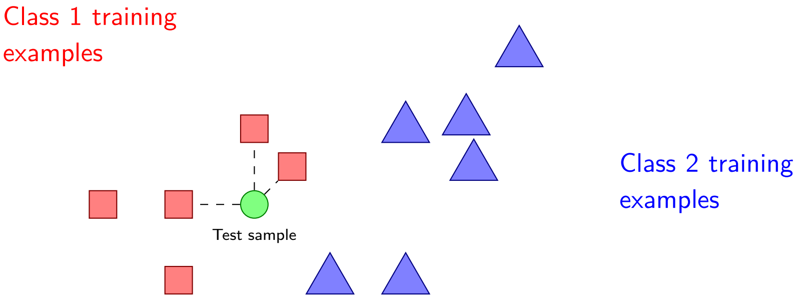
Nearest Neighbor



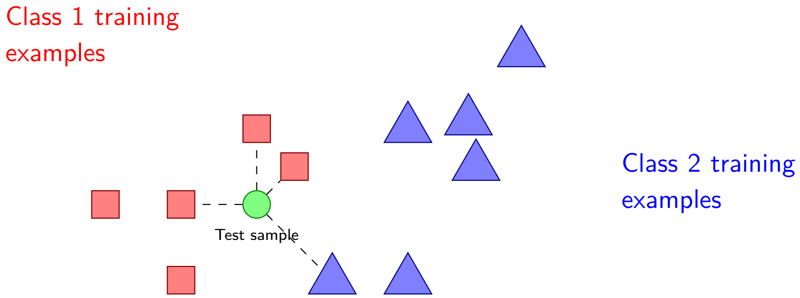
Nearest Neighbor



Nearest Neighbor



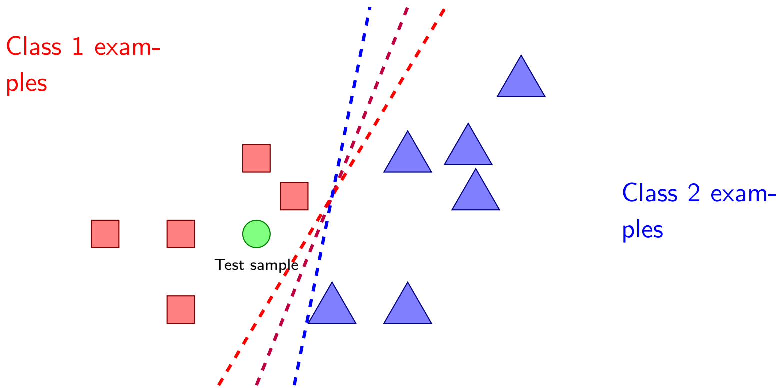
Nearest Neighbor



Using k-NN

- Simple, and a good baseline
- With infinite samples, 1-NN probably has an error at much as the double optimal Bayes error

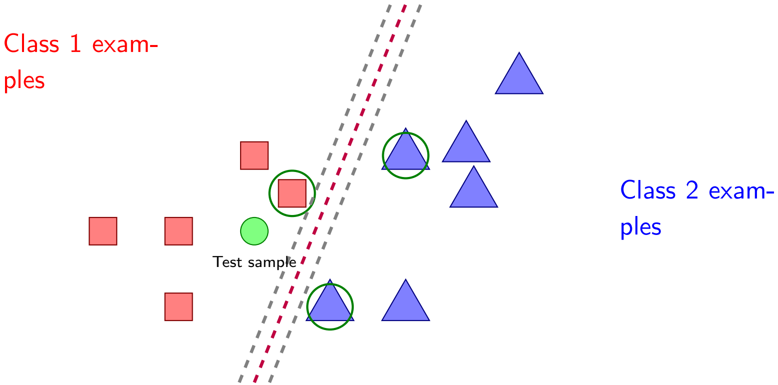
Linear Support Vector Machine



- Find a lineal function that separate the classes

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

Linear Support Vector Machine



- Find a lineal function that separate the classes

$$f(\mathbf{x}) = \text{sign}(\mathbf{wx} + b)$$

Non-linear SVM

- SVM works for linear separable data
- What about non-linear separable data?
- Solution: map the data to a higher dimensional space

General Idea

The original space can be transformed into a higher dimensional one where the data is separable

Kernel Trick

- Kernel Trick: instead of explicitly computing the transformation $\varphi(\mathbf{x})$, we define a kernel K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_j),$$

  where, K satisfies the Mercer's condition
- Then, we have a decision boundary in the original feature space

$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i)\varphi\mathbf{x} + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- More details: C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Example of Non-linear Kernel

- Consider the mapping $\varphi(x) = (x, x^2)$
- How is the generated space?
- Solution

$$K(x, y) = \varphi(x)\varphi(y)$$
$$\varphi(x)\varphi(y) = (x, x^2)(y, y^2)$$
$$K(x, y) = xy + x^2y^2$$

- We found a non-linear boundary from the original mapping

Bag of Features Kernels

- Histogram Intersection Kernel

$$I(h_1, h_2) = \sum_{i=1}^{N} \min\left(h_1(i), h_2(i)\right)$$

- Generalized Gaussian Kernel

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right),$$

  where, D can be the L_1 distance (inverse), Euclidean, χ^2, etc.

- More details: J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study, IJCV 2007

Summary of SVM

- Pick a representation of the images (bag of words, histograms, etc.)
- Pick a kernel according to the representation
- Compute the matrix of the kernel between each pair of samples
- Train the SVM using the previous matrix to find the support vectors and weights
- During testing
  - Compute the values of the kernel for the test data and each support vector
  - Combine them using the learned weights to obtain the decision value

Multi-class SVM

- There is no native multi-class SVM
- In practice, we obtain a multi-class SVM by combining several two-class SVM
- One vs. all
  - Train: learn an SVM per class vs. the rest
  - Test: apply each SVM to each test sample, and assign the class with best decision value
- One vs. one
  - Train: learn an SVM per each pair of classes
  - Test: each SVM votes per class according to the decision

SVM

- Good
  - Several SVM software packages (http://www.kernel-machines.org/software)
  - The frameworks based on kernels are potent and flexible
  - SVM work well in practice, despite having “small” training sets
- Bad
  - There is no multi-class formulation, and we need to combine SVM using some strategy
  - Computation and memory
    - During training time, we need to compute a complex matrix per each element pair
    - Learing can take time for complex problems

What to remember about classifiers?

- There is no free lunch: the learning algorithms are tools, and not dogmas
- Test simple classifiers for baseline
- It is best to have smart features and simple classifiers, than the opposite
- Use more complex classifiers with more data (compromise between variance and bias)

Extra References

- General
  - Tom Mitchell, Machine Learning, McGraw Hill, 1997
  - Christopher Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995
- Adaboost
  - Friedman, Hastie, and Tibshirani, Additive logistic regression: a statistical view of boosting, Annals of Statistics, 2000
- SVMs
  - http://www.support-vector.net/icml-tutorial.pdf