MACHINE LEARNING AND PATTERN RECOGNITION

Srping 2005, Lecture 1a:

Introduction and Basic Concepts

Yann LeCun
The Courant Institute,
New York University
http://yann.lecun.com

Before we get started...

- Course web site: http://www.cs.nyu.edu/ yann/2005s-V22-0480-006/index.html
- Evaluation: Assignements [40%] + final exam [30%] + final project [30%].
- Course mailing list (see course's web site).
- Text Books: mainly "Pattern Classification" by Duda, Hart, and Stork, but a number of other books can be used reference material: "Neural Networks for Pattern Recognition" by Bishop, and "Element of Statistical Learning" by Hastie, Tibshirani and Friedman.
- ... but we will mostly use my own material, augmented by tutorial papers and resarch papers.
- formal prerequisite: linear algebra. You might want to brush up on probability theory, multivariate calculus (partial derivatives ...), optimization (least square method...), and the method of Lagrange multipliers for constrained optimization. We will review those topics in class.
- Programming projects: can be done in any language, but I **STRONGLY** recommend to use Lush (http://lush.sf.net). Skeleton code in Lush will be provided for most projects.

What is Learning?

- Learning is acquiring and improving performance through experience.
- Pretty much all animals with a central nervous system are capable of learning (even the simplest ones).
- What does it mean for a computer to learn? Why would we want them to learn? How do we get them to learn?
- We want computers to learn when it is too difficult or too expensive to program them directly to perform a task.
- Get the computer to program itself by showing examples of inputs and outputs.
- In reality: we will write a “parameterized” program, and let the learning algorithm find the set of parameters that best approximates the desired function or behavior.

Different Types of Learning

- Supervised Learning: given training examples of inputs and corresponding outputs, produce the "correct" outputs for new inputs. Example: character recognition.
- Reinforcement Learning (similar to animal learning): an agent takes inputs from the environment, and takes actions that affect the environment. Occasionally, the agent gets a scalar reward or punishment. The goal is to learn to produce action sequences that maximize the expected reward (e.g. driving a robot without bumping into obstacles). I won't talk much about that in this course.
- Unsupervised Learning: given only inputs as training, find structure in the world: discover clusters, manifolds, characterize the areas of the space to which the observed inputs belong (e.g.: clustering, probability density estimation, novelty detection, compression, embedding).

Related Fields

- Statistical Estimation: statistical estimation attempts to solve the same problem as machine learning. Most learning techniques are statistical in nature.
- Pattern Recognition: pattern recognition is when the output of the learning machine is a set of discrete categories.
- Neural Networks: neural nets are now one many techniques for statistical machine learning.
- Data Mining: data mining is a large application area for machine learning.
- Adaptive Optimal Control: non-linear adaptive control techniques are very similar to machine learning methods.
- Machine Learning methods are an essential ingredient in many fields: bio-informatics, natural language processing, web search and text classification, speech and handwriting recognition, fraud detection, financial time-series prediction, industrial process control, database marketing....

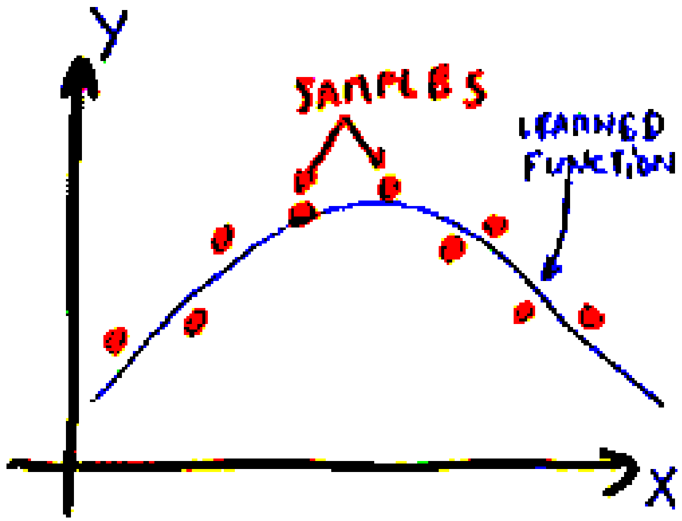
Applications

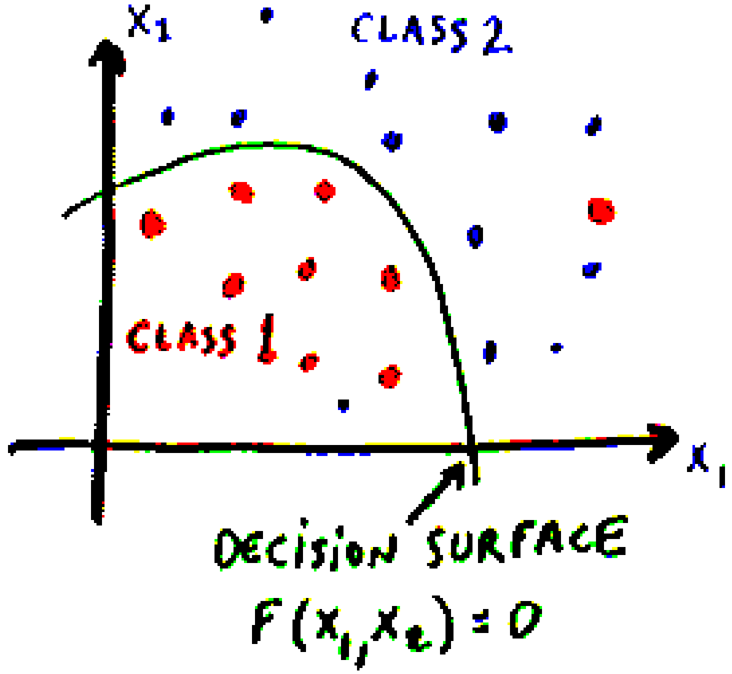
- handwriting recognition, OCR: reading checks and zipcodes, handwriting recognition for tablet PCs.
- speech recognition, speaker recognition/verification
- security: face detection and recognition, event detection in videos.
- text classification: indexing, web search.
- computer vision: object detection and recognition.
- diagnosis: medical diagnosis (e.g. pap smears processing)
- adaptive control: locomotion control for legged robots, navigation for mobile robots, minimizing pollutant emissions for chemical plants, predicting consumption for utilites...
- fraud detection: e.g. detection of "unusual" usage patterns for credit cards or calling cards.
- database marketing: predicting who is more likely to respond to an ad campaign.
- (...and the antidote) spam filtering.
- games (e.g. backgammon).
- Financial prediction (many people on Wall Street use machine learning).

Demos / Concrete Examples

- Handwritten Digit Recognition: supervised learning for classification
- Handwritten Word Recognition: weakly supervised learning for classification with many classes
- Face detection: supervised learning for detection (faces against everything else in the world).
- Object Recognition: supervised learning for detection and recognition with highly complex variabilities
- Robot Navigation: supervised learning and reinforcement learning for control.

Two Kinds of Supervised Learning

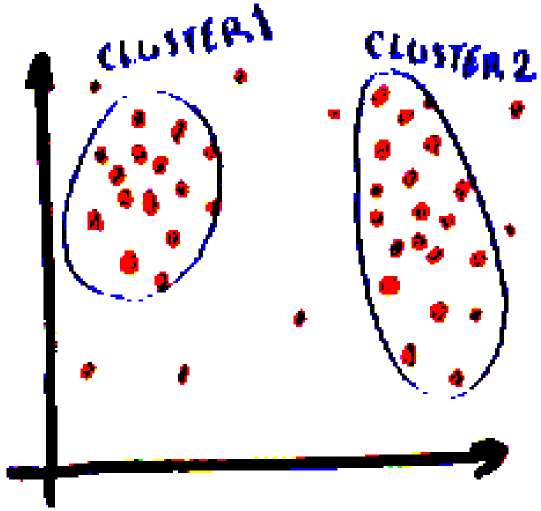


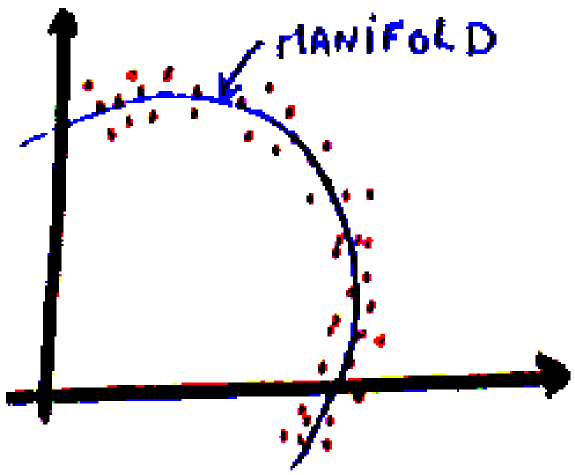


- Regression: also known as "curve fitting" or "function approximation". Learn a continuous input-output mapping from a limited number of examples (possibly noisy).
- Classification: outputs are discrete variables (category labels). Learn a decision boundary that separates one class from the other. Generally, a "confidence" is also desired (how sure are we that the input belongs to the chosen category).

Unsupervised Learning

Unsupervised learning comes down to this: if the input looks like the training samples, output a small number, if it doesn't, output a large number.



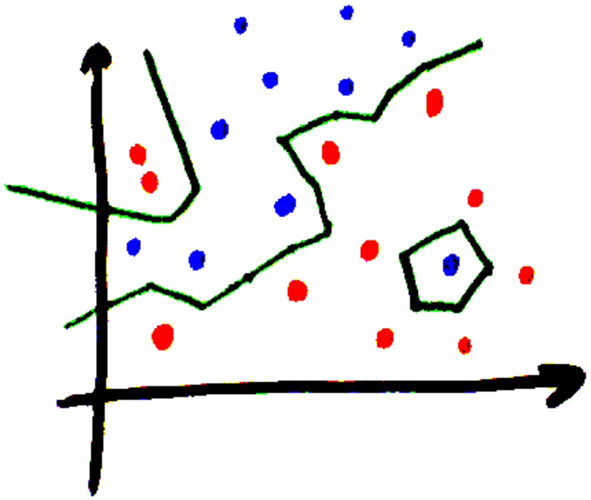


- This is a horrendously ill-posed problem in high dimension. To do it right, we must guess/discover the hidden structure of the inputs. Methods differ by their assumptions about the nature of the data.
- A Special Case: Density Estimation. Find a function f such $f(X)$ approximates the probability density of X, $p(X)$, as well as possible.
- Clustering: discover "clumps" of points
- Embedding: discover low-dimensional manifold or surface near which the data lives.
- Compression/Quantization: discover a function that for each input computes a compact "code" from which the input can be reconstructed.

Learning is NOT Memorization

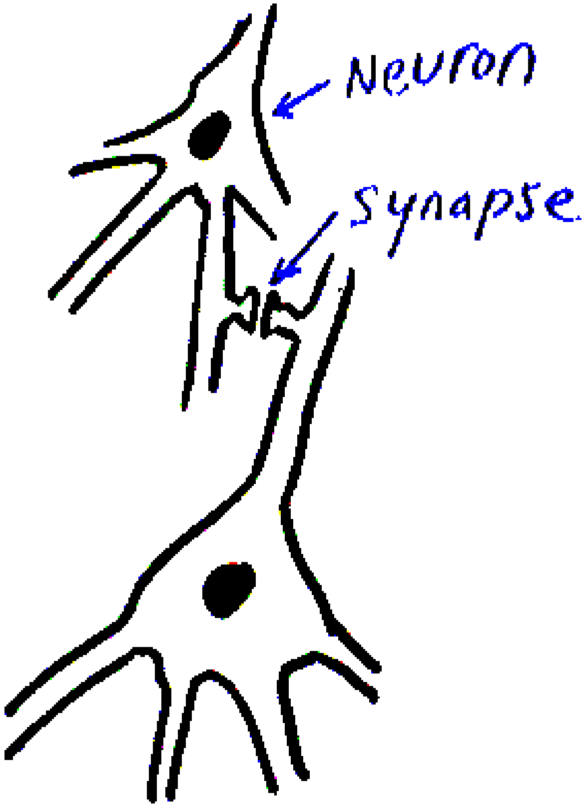
- rote learning is easy: just memorize all the training examples and their corresponding outputs.
- when a new input comes in, compare it to all the memorized samples, and produce the output associated with the matching sample.
- PROBLEM: in general, new inputs are different from training samples.
- The ability to produce correct outputs or behavior on previously unseen inputs is called GENERALIZATION.
- rote learning is memorization without generalization.
- The big question of Learning Theory (and practice): how to get good generalization with a limited number of examples.

A Simple Trick: Nearest Neighbor Matching



- Instead of insisting that the input be exactly identical to one of the training samples, let's compute the "distances" between the input and all the memorized samples (aka the prototypes).
- 1-Nearest Neighbor Rule: pick the class of the nearest prototype.
- K-Nearest Neighbor Rule: pick the class that has the majority among the K nearest prototypes.
- PROBLEM: What is the right distance measure?
- PROBLEM: This is horrendously expensive if the number of prototypes is large.
- PROBLEM: do we have any guarantee that we get the best possible performance as the number of training samples increases?

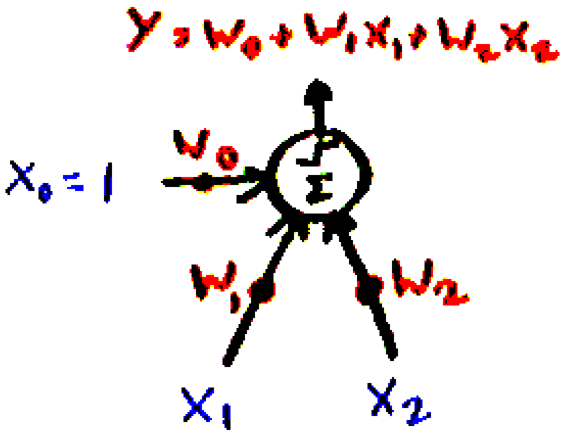
How Biology Does It

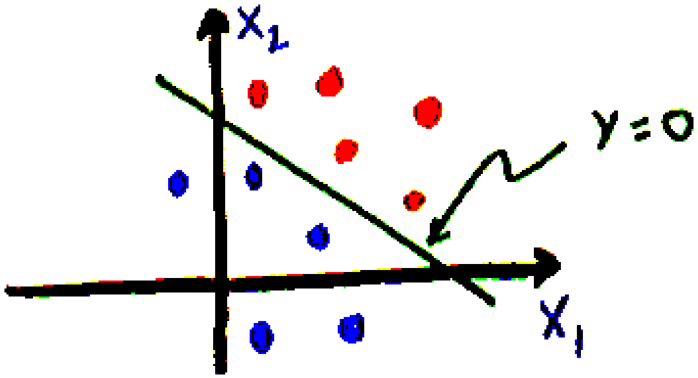


- The first attempts at machine learning in the 50's, and the development of artificial neural networks in the 80's and 90's were inspired by biology.
- Nervous Systems are networks of neurons interconnected through synapses
- Learning and memory are changes in the "efficacy" of the synapses
- HUGE SIMPLIFICATION: a neuron computes a weighted sum of its inputs (where the weights are the synaptic efficacies) and fires when that sum exceeds a threshold.
- Hebbian learning (from Hebb, 1947): synaptic weights change as a function of the pre- and post-synaptic activities.
- orders of magnitude: each neuron has 10^3 to 10^5 synapses. Brain sizes (number of neurons): house fly: 10^5; mouse: 5.10^6, human: 10^{10}.

The Linear Classifier

Historically, the Linear Classifier was designed as a highly simplified model of the neuron (McCulloch and Pitts 1943, Rosenblatt 1957):





$$y = f(\sum_{i=0}^{i=N} w_i x_i)$$

With f is the threshold function: $f(z) = 1$ iff $z > 0$, $f(z) = -1$ otherwise. x_0 is assumed to be constant equal to 1, and w_0 is interpreted as a bias.
In vector form: $W = (w_0, w_1 ....w_n), X = (1, x_1 ...x_n)$:

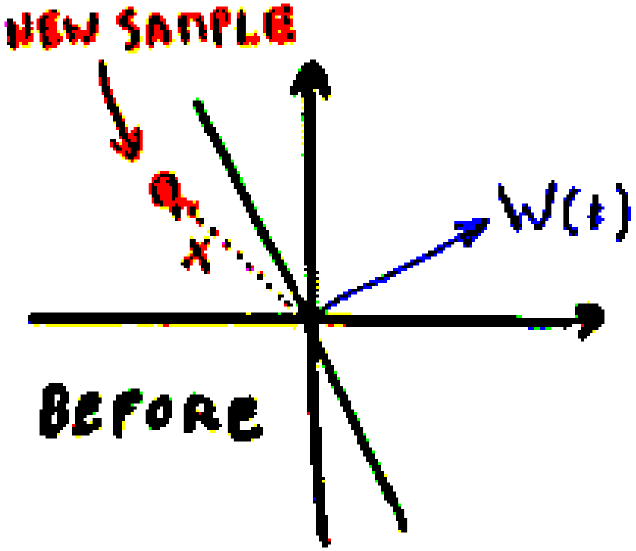
$$y = f(W'X)$$

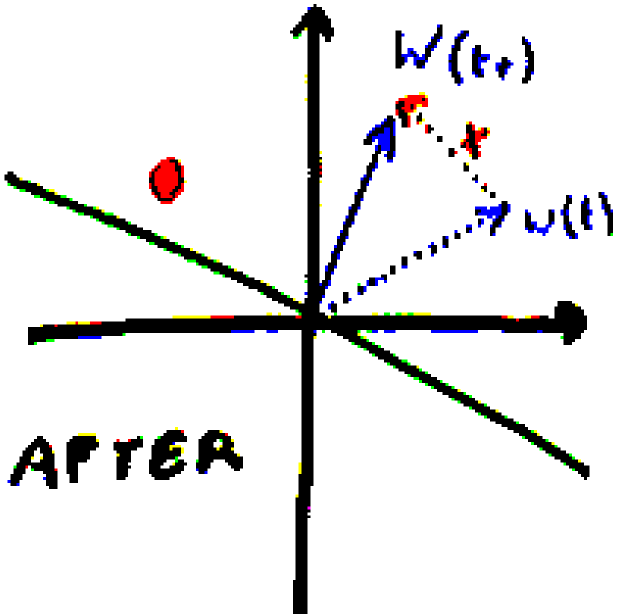
The hyperplane $W'X = 0$ partitions the space in two categories. W is orthogonal to the hyperplane.

Vector Inputs

With vector-based classifiers such as the linear classifier, we must represent objects in the world as vectors.
Each component is a measurement or a feature of the the object to be classified.
For example, the grayscale values of all the pixels in an image can be seen as a (very high-dimensional) vector.

A Simple Idea for Learning: Error Correction





We have a **training set** $\mathcal{S}$consisting of P input-output pairs: $\mathcal{S} = (X^1, y^1), (X^2, y^2), ...(X^P, y^P)$.
A very simple algorithm:
- show each sample in sequence repetitively
- if the output is correct: do nothing
- if the output is -1 and the desired output +1: increase the weights whose inputs are positive, decrease the weights whose inputs are negative.
- if the output is +1 and the desired output -1: decrease the weights whose inputs are positive, increase the weights whose inputs are negative.

More formally, for sample p:

$$w_i(t+1) = w_i(t) + (y_i^p - f(W'X^p))x_i^p$$

This simple algorithm is called the Perceptron learning procedure (Rosenblatt 1957).