CSE 250B: Machine Learning

Lecture 0: Overview

Machine learning versus Algorithms

In both fields, the goal is to develop

procedures that exhibit a desired input-output behavior.

- **Algorithms**: the input-output mapping can be precisely defined.
  Input: Graph G.
  Output: MST of G.
- **Machine learning**: the mapping cannot easily be made precise.
  Input: Picture of an animal.
  Output: Name of the animal.

Instead, we simply provide examples of (input,output) pairs and ask the machine to *learn* a suitable mapping itself.

Inputs and outputs

Basic terminology:

- The input space, $\mathcal{X}$.
  E.g. 32×32 RGB images of animals.
- The output space, $\mathcal{Y}$.
  E.g. Names of 100 animals.

x:

y: "bear"

After seeing a bunch of examples (x, y), pick a mapping

$$f : \mathcal{X} \to \mathcal{Y}$$

that accurately replicates the input-output pattern of the examples.

Learning problems are often categorized according to the type of *output space*: (1) discrete, (2) continuous, (3) probability values, or (4) more general structures.

Discrete output space: classification

Binary classification:

- Spam detection
  $\mathcal{X} = \{\text{email messages}\}$
  $\mathcal{Y} = \{\text{spam}, \text{not spam}\}$
- Credit card fraud detection
  $\mathcal{X} = \{\text{descriptions of credit card transactions}\}$
  $\mathcal{Y} = \{\text{fraudulent}, \text{legitimate}\}$

Multiclass classification:

- Animal recognition
  $\mathcal{X} = \{\text{animal pictures}\}$
  $\mathcal{Y} = \{\text{dog}, \text{cat}, \text{giraffe}, \ldots\}$
- News article classification
  $\mathcal{X} = \{\text{news articles}\}$
  $\mathcal{Y} = \{\text{politics}, \text{business}, \text{sports}, \ldots\}$

Continuous output space: regression

- A parent's concerns
  How cold will it be tomorrow morning?
  $\mathcal{Y} = [-273, \infty)$
- For the asthmatic
  Predict tomorrow's air quality (max over the whole day)
  $\mathcal{Y} = [0, \infty)$ $\quad$ (< 100: okay, > 200: dangerous)
- Insurance company calculations
  In how many years will this person die?
  $\mathcal{Y} = [0, 200]$

What are suitable predictor variables ($\mathcal{X}$) in each case?

Conditional probability functions

Here $\mathcal{Y} = [0, 1]$ represents probabilities.

- Dating service
  What is the probability these two people will go on a date if introduced to each other?
  If we modeled this as a classification problem, the binary answer would basically always be "no". The goal is to find matches that are slightly less unlikely than others.
- Credit card transactions
  What is the probability that this transaction is fraudulent?
  The probability is important, because – in combination with the amount of the transaction – it determines the overall risk and thus the right course of action.

Structured output spaces

The output space consists of structured objects, like sequences or trees.

Dating service

Input: description of a person
Output: rank-ordered list of all possible matches

$\mathcal{Y}$ = space of all permutations

Example:
$x = Tom$
$y = (Nancy, Mary, Chloe, \ldots)$

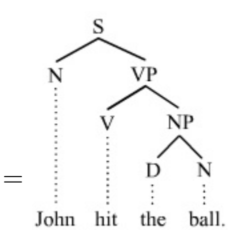
Language processing

Input: English sentence
Output: parse tree showing grammatical structure

$\mathcal{Y}$ = space of all trees

Example:
$x =$ "John hit the ball"

$y =$

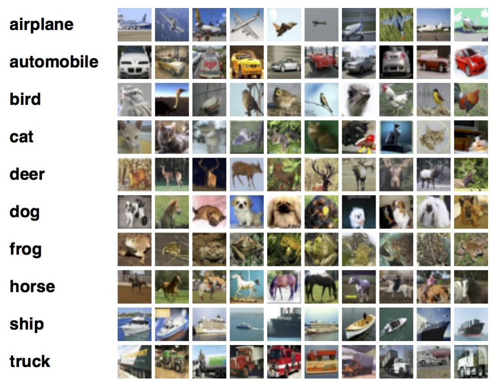


Course outline

1. Nonparametric methods
2. Classification using parametrized models
3. Combining classifiers
4. Representation learning

Nonparametric methods: nearest neighbor

Training set: a collection of (x, y) pairs:

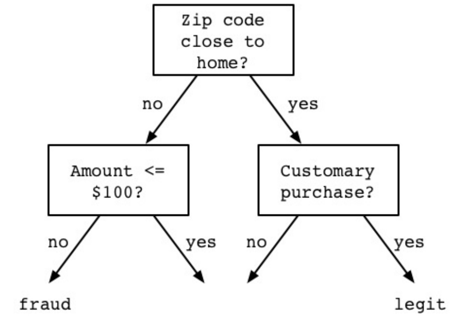


Given any x, find its nearest neighbor in the training set and predict that neighbor's y value.

Issues: (1) What distance function? (2) How to speed up search?

Nonparametric methods: decision tree

Credit card fraud detection: use training data to build a tree classifier



What do nearest neighbor and decision trees have in common?

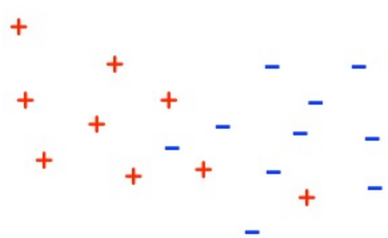
- Unbounded in size
- Can model arbitrarily complex functions

They are *nonparametric methods*.

Classification with parametrized models

Classifiers with a fixed number of parameters can represent a limited set of functions. Learning a model is about picking a good approximation.

Typically the x's are points in d-dimensional Euclidean space, $\mathbb{R}^d$:



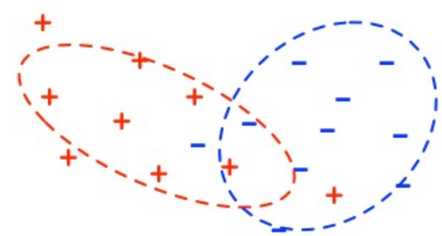
Two ways to classify:

- *Generative*: model the individual classes.
- *Discriminative*: model the decision boundary between the classes.

Generative models

Fit a probability distribution – like a multivariate Gaussian – to each class. Thereafter use this *summary* rather than the data points themselves.

To classify a new point: find the most probable class.



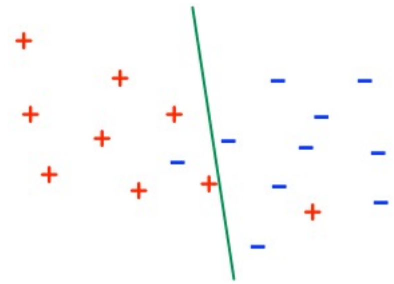
Examples: Naive Bayes, Fisher discriminant.

Under the hood: Bayes' rule, linear algebra (eigenvalues, eigenvectors).

Discriminative models

Approximate the boundaries between classes by simple – e.g. linear – functions.

To classify a new point: figure out which side of the boundary it lies on.



Examples: support vector machine, logistic regression.

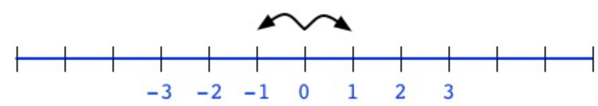
Under the hood: convex duality, optimization.

Generalization theory

- Complex, e.g. nonparametric, classifiers require a lot of training data to learn accurately.
- Simple, e.g. linear, classifiers require less.

What is the right notion of complexity? Are there formulas for how much data is enough? The answers are based on *large deviation theory*.

Example: *Symmetric random walk.*
A drunken man starts at the origin and at each time step either takes a step to the right or a step to the left. Where is he after n steps?



Answer: Not far from where he started! Most likely in $[-c\sqrt{n}, c\sqrt{n}]$ (for some constant c), and all positions in this interval are about equally likely.

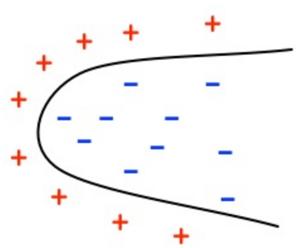
Under the hood: probability theory.

Richer classifiers via the kernel trick

We are good at finding linear classifiers in Euclidean space. But what if:

- The boundary between classes is far from linear?

  Example: quadratic, or higher-order polynomial, or even stranger.



- The data aren't even vectors of numbers?

  Example: documents, DNA sequences, parse trees.

The *kernel trick* handles these scenarios seamlessly, by mapping the data to a suitable Euclidean space in which linear classification is possible!

Richer output spaces

Many classification methods were developed for the binary (two-label) case. Usually the output space is larger than this.

- $\mathcal{Y}$ = several classes.

  Examples:

  x = image, y = name of object in image

  x = news article, y = category (sports, politics, business, ...)

- $\mathcal{Y}$ = structured objects.

  Examples:

  x = sentence in Swahili, y = transcription into English

  x = sentence in English, y = parse tree

Extend binary classification to handle such cases!

Under the hood: error-correcting codes, dynamic programming.

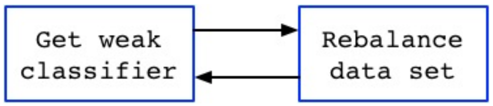
Composing simple classifiers

A common situation in classifier learning:

> *Easy to find* **weak classifiers** *– not very accurate, but better than random*

To increase accuracy, *compose* weak classifiers.

Example: *boosting*.



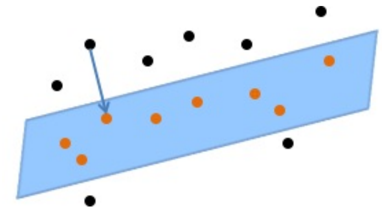
Final classifier is a linear combination of all these weak classifiers.

Generically improve the performance of *any* kind of classifier!

Representation learning

A handful of key primitives:

1. Dimensionality reduction and denoising.
   Given data in high-dimensional Euclidean space, project to a low-dimensional linear subspace while retaining as much of the signal as possible.

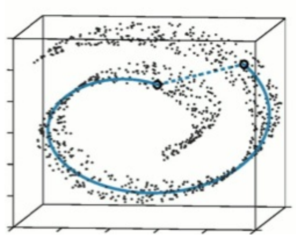


2. Embedding and manifold learning.
3. Metric learning.

Representation learning

A handful of key primitives:

1. Dimensionality reduction and denoising.
2. Embedding and manifold learning.
   Given data that lie in a non-Euclidean space, find an embedding into Euclidean space that preserves as much of the geometry as possible.



3. Metric learning.

Representation learning

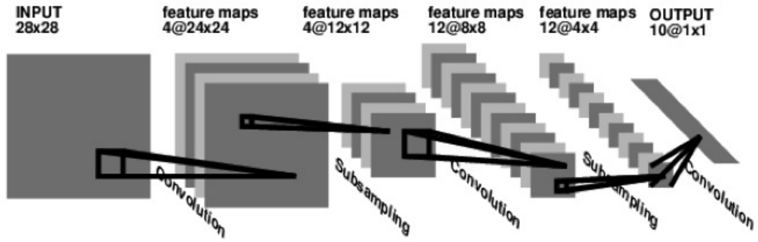
A handful of key primitives:

1. Dimensionality reduction and denoising.
2. Embedding and manifold learning.
3. Metric learning.
   Given data with only vague positional information, impose an Euclidean geometry that is suitable for classification.
   Example: $\mathcal{X} = \{$a collection of m books$\}$.
   A user supplies $\binom{m}{2}$ similarity ratings, such as:
   ("Pride and Prejudice", "Great Expectations"): similar
   ("Hamlet", "Great Expectations"): dissimilar
   Represent each book by a vector, respecting these ratings.

Under the hood: linear algebra, semidefinite programming.

Deep learning

Multi-layer neural nets achieve state-of-the-art performance across a range of benchmark problems in natural language processing, speech, and vision.



Under the hood: stochastic gradient descent, dictionary learning, autoencoders.

Class details

Website: `http://www.cs.ucsd.edu/~dasgupta/250B`.

Grading:

- Regular homeworks: 50%.
- Two midterms: 25% each.

Course outline

1. Nonparametric methods
2. Classification using parametrized models
    - Generative models
    - Discriminative models
    - Richer decision boundaries using the kernel trick
    - Richer output spaces
    - Generalization theory
3. Combining classifiers
    - Boosting, bagging, and random forests
    - Online learning
4. Representation learning
    - Linear projection
    - Embeddings
    - Metric learning
    - Deep learning