ECE 6254
Statistical Machine Learning
Spring 2018

David V. Anderson

(slides by Mark Davenport of Georgia Tech)

Georgia Institute of Technology

School of Electrical and Computer Engineering

Statistical machine learning

- How can we
  - learn effective models from data?
  - apply these models to practical inference and signal processing problems?
- Applications include: classification, prediction, regression, clustering, modeling, and data exploration/visualization
- Our approach: ***statistical inference***
- **Main subject of this course**
  - how to reason about and work with probabilistic models to help us make inferences from data

What is machine learning?

learn: gain or acquire knowledge of or skill in (something) by study, experience, or being taught

How do we learn that this is a tree?

Definition?

(A perennial plant with an elongated stem, or trunk, supporting leaves or branches.)

EXAMPLES!

A good definition of learning for this course:

"using a set of examples to *infer* something about an underlying process"

Why learn from data?

Traditional signal processing is "top down"

Given a model for our data, derive the optimal algorithm

A learning approach is more "bottom up"

Given some examples, derive a good algorithm

Sometimes a good model is ***really*** hard to derive from first principles

Examples of learning

The Netflix prize: Predict how a user will rate a movie

10% improvement = $1 million prize

- Some pattern exists
  - users do not assign ratings completely at random – if you like Godfather I, you'll probably like Godfather II
- It is hard to pin down the pattern mathematically
- We have lots and lots of data
  - we know how a user has rated other movies, and we know how other users have rated this (and other) movies

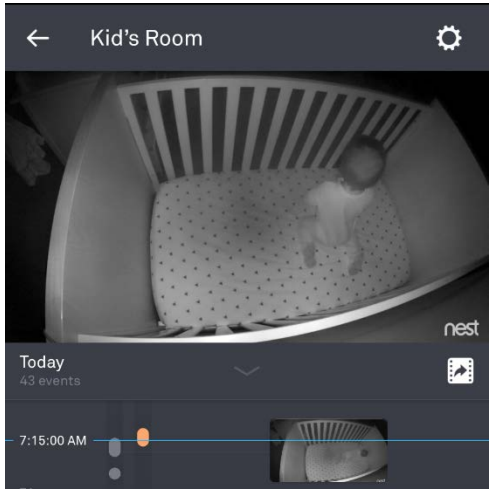
Examples of learning

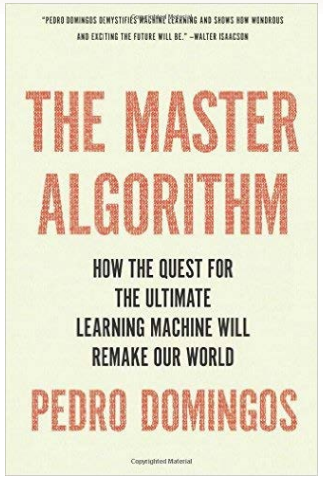
Handwritten digit recognition

A day in the life…

Waking up in the morning

PANDORA®





A day in the life…

Getting into the car to drive to work

A day in the life…

Once at work

Google

Google Translate

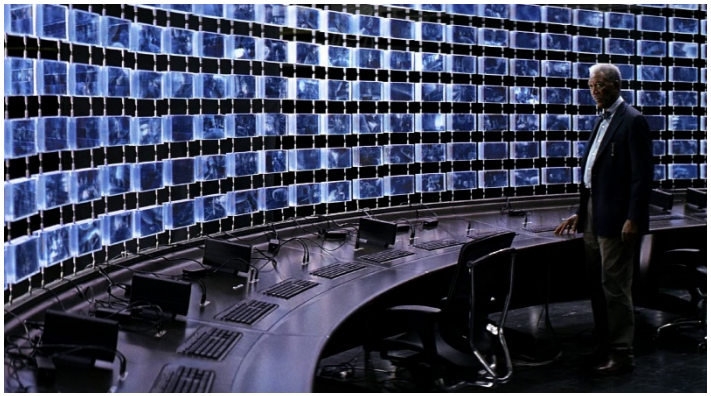
A day in the life…

Over the lunch break

yelp®

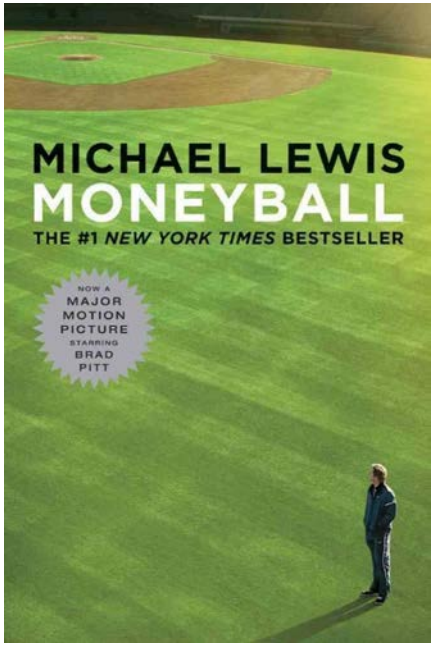
A day in the life…

Heading home for the day



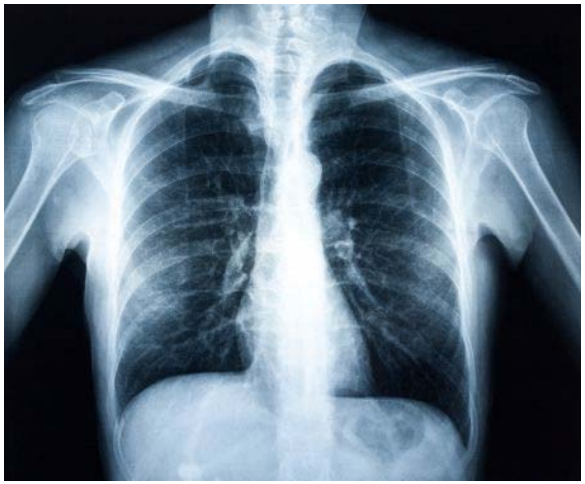
A day in the life…

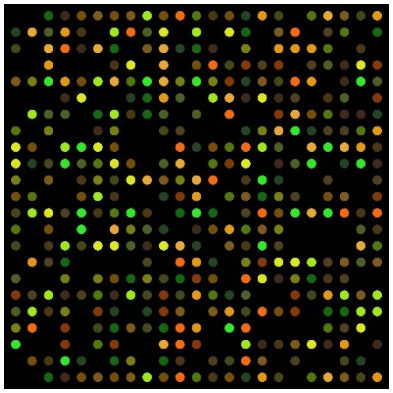
Finally home for the day



A day in the life…

Before getting into bed





match.com®

Supervised learning

We are given input data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

Each $\mathbf{x}_i$ represents a measurement or observation of some natural or man-made phenomenon

- may be called input, pattern, signal, feature vector, instance, or independent variable
- the coordinates may be called features, attributes, predictors, or covariates

In the supervised case, we are also given output data $y_1, \ldots, y_n$

- may be called output, label, response, or dependent variable

The data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are called the ***training data***

Supervised learning

We can think of a pair $(\mathbf{x}_i, y_i)$ as obeying a (possibly noisy) input-output relationship

The goal of supervised learning is usually to ***generalize*** the input-output relationship so that we can predict the output y associated with a previously unseen input $\mathbf{x}$

The primary supervised learning problems are

- classification: $y \in \{1, \ldots, m\}$
- regression: $y \in \mathbb{R}$

Unsupervised learning

The inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are not accompanied by labels

The goal of unsupervised learning is typically not related to future observations. Instead we just want to understand that structure in the data sample itself, or to infer some characteristic of the underlying probability distribution.

Examples of unsupervised learning problems include

- clustering
- density estimation
- dimensionality reduction/feature selection
- visualization

Other variants of learning

- semi-supervised learning
- active learning
- online learning
- reinforcement learning
- anomaly detection
- ranking
- transfer learning
- multi-task learning
- …

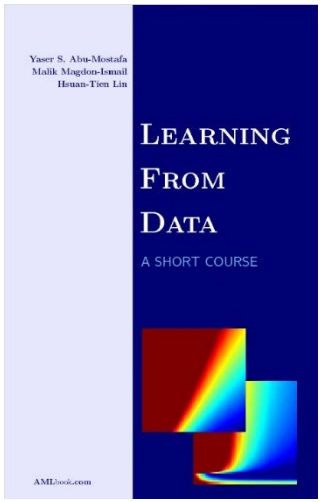
In general, most learning problems can be thought of as variants of traditional signal processing problems, but where we have ***no idea (a priori) how to model our signals***

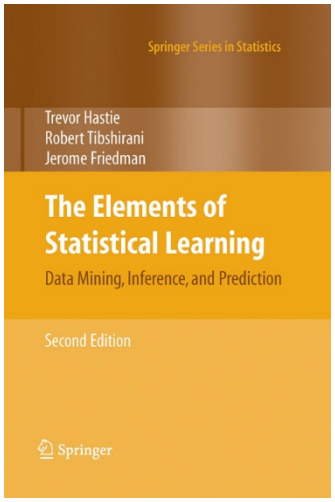
Prerequisites

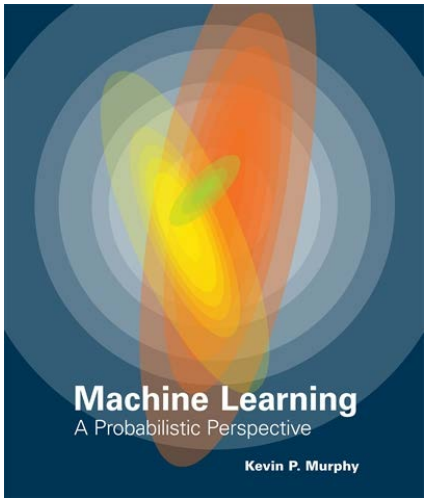
- Probability
  - random variables, expectation, joint distributions, independence, conditional distributions, Bayes rule, multivariate normal distribution, …
- Linear algebra
  - norms, inner products, orthogonality, linear independence, eigenvalues/vectors, eigenvalue decompositions, …
- Multivariable calculus
  - partial derivatives, gradients, the chain rule, …
- Python or similar programming experience (C or MATLAB)

Text

There is no formally required textbook for this course, but I will draw material primarily from these sources:







A list of other useful books and links to relevant papers will be posted on the course webpage

Lecture notes will also be posted on the course webpage

Grading

- Pre-test (5%)
- Homework (25%)
- Midterm exam (20%)
- Final exam (20%)
- Final project (25%)
- Participation (5%)

Distance learning

Welcome to our online students!

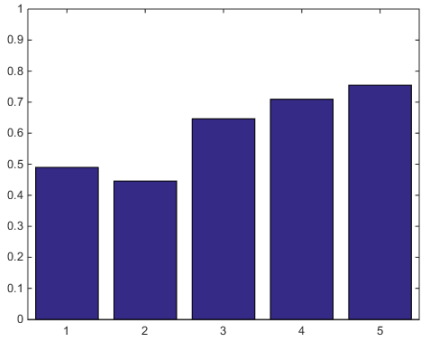
Recorded lectures will be available to ***all*** students (including on-campus students)

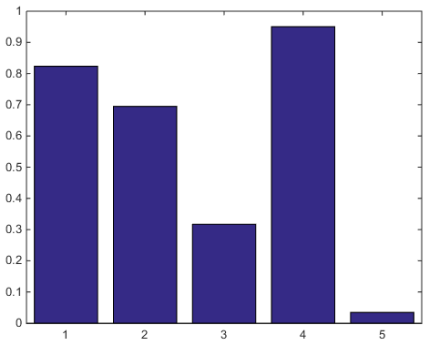
I need your help to make this a success

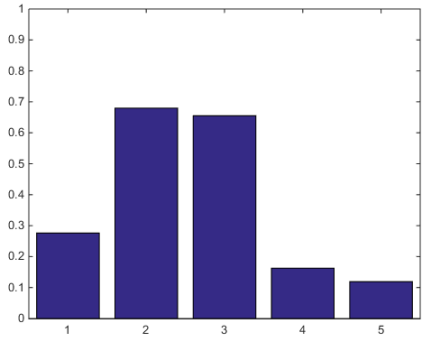
Online resources:

- Course website
- T-square
- Piazza

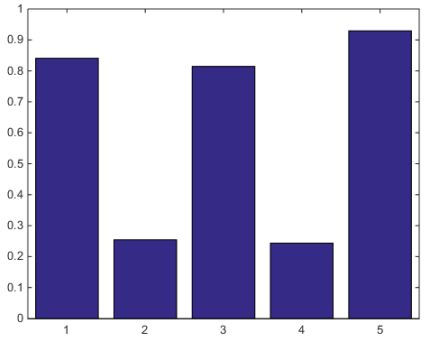
A learning puzzle?

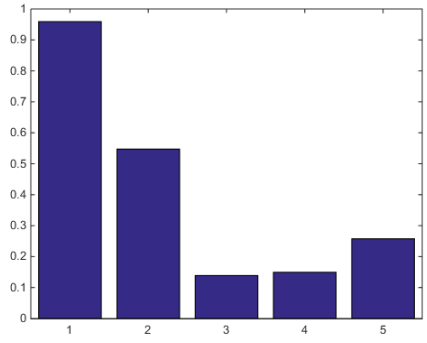


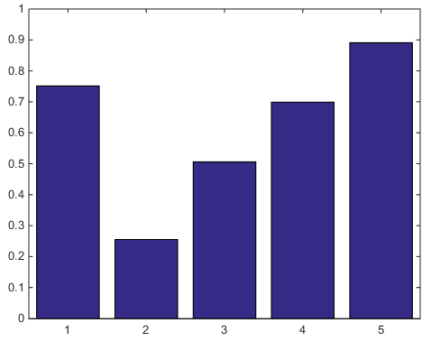




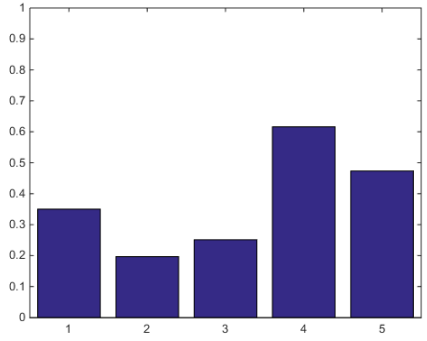
$y = +1$







$y = -1$



$y = ?$

Is learning even possible?

or: How I learned to stop worrying and love statistics

Supervised learning

Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, we would like to learn an (unknown) function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(\mathbf{x}) = y$ for $\mathbf{x}$ other than $\mathbf{x}_1, \ldots, \mathbf{x}_n$

but...

as we have just seen, this is impossible. Without any additional assumptions, we conclude ***nothing*** about f except (maybe) for its value on $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Probability to the rescue!

Any f agreeing with the training data may be ***possible***

but that does not mean that any f is equally ***probable***

A short digression

- Suppose that I have a biased coin, which lands on heads with some unknown probability p
  - $\mathbb{P}[\text{heads}] = p$
  - $\mathbb{P}[\text{tails}] = 1 - p$
- I toss the coin n times (independently)
  - $\widehat{p} = \frac{\#\text{ of heads}}{n}$

Does $\widehat{p}$ tell us anything about p?

What can we learn from $\widehat{p}$?

Given enough tosses (large n), we expect that $\widehat{p} \approx p$

Law of large numbers

$$\widehat{p} \to p \text{ as } n \to \infty$$

Clearly, at least in a very limited sense, we can learn something about p from observations

There is always the ***possibility*** that we are totally wrong, but given enough data, the ***probability*** should be very small

Connection to learning

Coin tosses: We want to estimate p

Learning: We want to estimate a function $f : \mathcal{X} \to \mathcal{Y}$

Suppose we have a ***hypothesis*** h and that $\mathcal{Y}$ is discrete

Think of the $(\mathbf{x}_i, y_i)$ as a series of independent coin tosses, where the $(\mathbf{x}_i, y_i)$ are drawn from a probability distribution

- heads: our hypothesis is correct, i.e., $h(\mathbf{x}_i) = y_i$
- tails: our hypothesis is wrong, i.e., $h(\mathbf{x}_i) \neq y_i$

Define

Risk: $R(h) := \mathbb{P}[h(X) \neq Y]$

Empirical risk: $\widehat{R}_n(h) := \frac{1}{n} \sum_{i=1}^{n} 1_{\{h(\mathbf{x}_i) \neq y_i\}}(i)$

Trust, but verify

The law of large numbers guarantees that as long as we have enough data, we will have that $R(h) \approx \widehat{R}_n(h)$

This means that we can use $\widehat{R}_n(h)$ to verify whether h was a good hypothesis

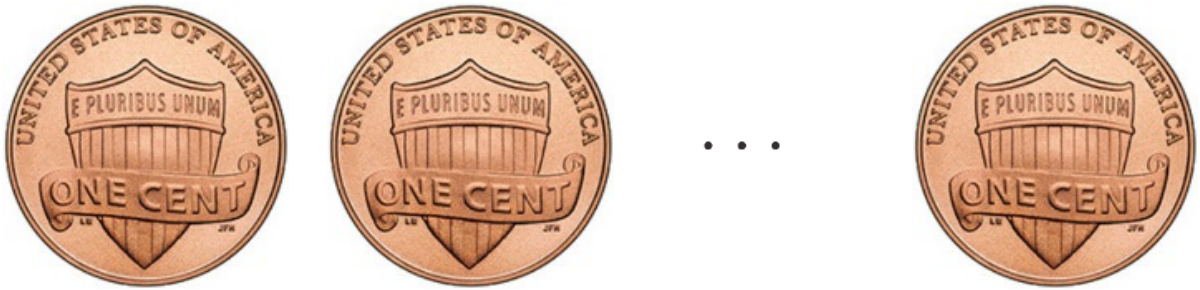
Unfortunately, ***verification is not learning***

- Where did h come from?
- What if $R(h)$ is large?
- How do we know if $h \approx f$, or at least, if $R(h) \approx R(f)$?
- Given many possible hypotheses, how can we pick a good one?

E pluribus unum

Consider an ensemble of many hypotheses

$$\mathcal{H} = \{h_1, \ldots, h_m\}$$



If we fix a hypotheses h_j before drawing our data, then the law of large numbers tells us that $\widehat{R}_n(h_j) \to R(h_j)$

However, it is also true that for a fixed n, if m is large it can still be very likely that there is some hypothesis h_k for which $\widehat{R}_n(h_k)$ is still very far from $R(h_k)$

Back to the coin analogy

Question 1: If I toss a fair coin 10 times, what is the probability that I get 10 heads?

Answer: … 0.001

Question 2: If I toss 1000 fair coins 10 times each, what is the probability that ***some*** coin will get 10 heads?

Answer: … 0.624

This phenomenon forms the fundamental challenge of ***multiple hypothesis testing***

...and back to learning

If we have many hypotheses (large m), then

even though for any fixed hypothesis h_j it is likely that

$$\widehat{R}_n(h_j) \approx R(h_j)$$

it is also likely that there will be at least ***one*** hypothesis h_k where $\widehat{R}_n(h_k)$ is very different from $R(h_k)$

How to adapt our approach to handle many hypotheses?

Next time: We will be a bit more quantitative and take a first crack at solving this problem