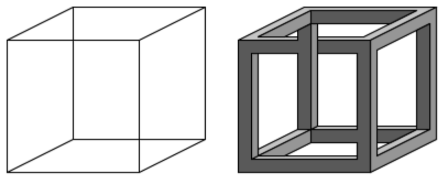
mlentary (optional 6.86x notes)

What tools can we use to automatically extract, extrapolate, and explain patterns in data? These notes review the main tools we develop in 6.86x. As binocular vision deepens and disambiguates, the differences between how our lectures and how these notes organize concepts may also enrich understanding.

You do not need to read these notes at all to get an A in this course; conversely, you may not cite these notes when justifying work in homework or exams.



prologue

TODO: five labeled pictures of mushrooms and poison levels (say, "safe", "stomach upsetting", "very dangerous"). A sixth, un-labeled picture with a question mark for its label. Make explicit that we view mushrooms are prompts/inputs and poison levels as answers/outputs.

TYPES OF LEARNING — How do we communicate patterns of desired behavior? We can teach:
by instruction: "to tell whether a mushroom is poisonous, first look at its gills..."
by example: "here are six poisonous fungi; here, six safe ones. see a pattern?"
by reinforcement: "eat foraged mushrooms for a month; learn from getting sick."
Machine learning is the art of programming computers to learn from such sources. We'll focus on the most important case: learning from examples. Given a list of N examples of properly answered prompts, we seek a pattern: a map from prompts in $\mathcal{X}$ to answers in $\mathcal{Y}$. So a program that learns from examples has type

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^N \to (\mathcal{X} \to \mathcal{Y})$$

← We'll see in §E that learning by example is key to the other modes of learning.

← Actually, we'll soon account for uncertainty by letting patterns map to *probability distributions over* answers; then a program that learns from examples has type

$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^N \to (\mathcal{X} \to \text{DistributionsOn}(\mathcal{Y}))$$

Sometimes, the prompt is always the same — say, "produce a beautiful melody" — and we seek (e.g.) to generate many answers. When we thusly focus on the output's structure, we're doing **unsupervised learning**; when our focus is on input-output relation, we're doing **supervised learning**.

Six labeled MNIST (x, y) pairs.

LEARNING ERROR — For $\mathcal{S} : (\mathcal{X} \times \mathcal{Y})^N$ a list of examples drawn from nature's distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, a pattern $f : \mathcal{X} \to \mathcal{Y}$ has a **training error** $\text{trn}_{\mathcal{S}}(f) = \mathbb{P}_{(x,y)\sim\mathcal{S}}[f(x) \neq y]$, an average over examples; and a **testing error** $\text{tst}(f) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$, an average over nature. We want $\mathcal{L}$ to map $\mathcal{S}$ to an f with low $\text{tst}(f)$. We often define $\mathcal{L}$ to roughly minimize $\text{trn}_{\mathcal{S}}$ over a set $\mathcal{H} \subseteq (\mathcal{X} \to \mathcal{Y})$ of candidates. Then tst decomposes into the failures of $\text{trn}_{\mathcal{S}}$ to estimate tst, of $\mathcal{L}$ to minimize $\text{trn}_{\mathcal{S}}$, and of $\mathcal{H}$ to contain nature's truth:

$$\begin{aligned}
\text{tst}(\mathcal{L}(\mathcal{S})) = \;& \text{tst}(\mathcal{L}(\mathcal{S})) && - \text{trn}_{\mathcal{S}}(\mathcal{L}(\mathcal{S})) && \}\ \text{generalization error} \\
& + \text{trn}_{\mathcal{S}}(\mathcal{L}(\mathcal{S})) && - \inf_{\mathcal{H}}(\text{trn}_{\mathcal{S}}(f)) && \}\ \text{optimization error} \\
& + \inf_{\mathcal{H}}(\text{trn}_{\mathcal{S}}(f)) && && \}\ \text{approximation error}
\end{aligned}$$

These terms are in tension. For example, as $\mathcal{H}$ grows, the approximation error may decrease while the generalization error may increase ("bias-variance").

MENTION $N = 24$! Left: scatterplot of $N = 24$ TRAINING set in width-darkness plane, with six above labeled pairs marked and with decision line for specific (suboptimal) (a, b) pair drawn and with decision line for optimal (a, b) pair drawn. Right: grid of all (a, b) pairs, colored by TRAINING error, with same two (a, b) pairs marked.

TINY EXAMPLE — $\mathcal{X} = \{\text{grayscale } 28\times28\text{-pixel images}\}$; $\mathcal{Y} = \{0, 1\}$. $\mathcal{D}$'s samples are photos x of handwriting of digit y. Each x has a width (number of inked columns) and darkness (average ink per column). $\mathcal{H}$ contains all $f_{a,b}$ defined by:

$$f_{a,b}(x) = \;0\; \text{ if } \; a \cdot \text{width}(x) + b \cdot \text{darkness}(x) < 0 \;\text{ else }\; 1$$

Our program $\mathcal{L}$ loops over all integer pairs (a, b) in $[-99, +99]$ to minimize $\text{trn}_{\mathcal{S}}$. $\mathcal{L}$ gives $(a, b) = (??, ??)$; $\text{trn}_{\mathcal{S}} = ??\%$ but $\text{tst} = ??\%$, so we see gen. error. Our brute-force loop sets opt. error to ≈ 0. Width and darkness lose important information; and straight lines badly approximate the boundary between digits; hence our large approx. error ($\approx ??\%$). In 6.86x we'll address all three issues.

Left: scatterplot of size-96 TESTING set in width-darkness plane, with decision line for training-optimal (a, b) pair drawn and with decision line for testing-optimal (a, b) pair drawn. Right: grid of all (a, b) pairs, colored by TESTING error, with same two (a, b) pairs marked.

linear models

nonlinearities

structured inference

graphical generative models

inferring conditional marginals

learning parameters

hierarchy and mixtures

hierarchy and transfer

variational and sampling methods

amortized inference

reductions to supervision

three brief example projects

Now comes a hurried tour of the sorts of analysis these notes discuss. I recommend that you skim these three example projects without trying to understand each step.

flooding plains

VISUALIZATION —

FITTING —

MODEL SELECTION —

PREDICTION —

OVERFITTING —

A WHIMSICAL STORY ABOUT AUTOMATION — We automate when we re-cast the previous generation's *methods* as mere *data* to produce and manipulate.

We once sought safety from floods. Long ago, if we lived near safe riverbanks it was by luck; we then found we could make unsafe rivers safe by erecting floodwalls. This was dangerous work, so we built brick laying robots to do the hard part. The machines needed as input some flood-risk estimates: the location and mud-softnesses of the river banks to work on.

So we sought flood-risk estimates. At first, if our city's flood-risk estimates were correct it was by luck; we then found we could translate topographic maps to flood-risk estimates by applying geology ideas. This was tedious work, so we built computers to do the hard part. The computers needed as input a risk-estimation *algorithm*: a recipe of the geology rules to calculate through.

We thus sought risk-estimation algorithms. At first, if our research center's risk-estimation algorithms were robust to new landscapes it was by luck; we then found we could calibrate our algorithms to historical flood data from many cities. To find the better of two candidate algorithms, we had to aggregate all their many small success and errors. This was subtle work, so we developed statistical theories to do the hard part. The theories needed as input a machine learning model: a computationally tractable class of candidate algorithms.

We thus sought machine learning models. At first, if our machine learning models were highly predictive it was by luck; we then found general principles to guide model design and selection from domain knowledge.

It is these principles that we study in 6.86x.

Looking back, these are principles that produce machine learning models that produce risk-estimation algorithms that produce flood-risk estimates that guide the building of floodwalls.

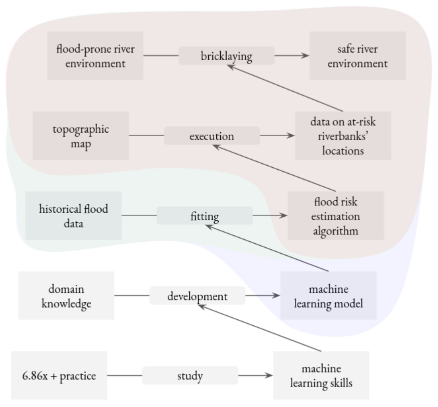


Figure 1: Machine learning continues our human tradition toward the systematic and automatic. What for one generation is a luckily discovered idea or recipe (bottom tip of red), the next generation refines by careful human thought (green). What to one generation is a task (green) requiring human thought is to a future generation merely a routine parameterized by some newly discovered recipe (bottom tip of blue). And the cycle repeats.

ancient tablets

VISUALIZATION —

TRANSCRIPTION —

CLEANING AND IMPUTATION —

MODELING AND FITTING —

GENERATION —

A WHIMSICAL STORY ABOUT MODELING —

pairing flavors

VISUALIZATION —

MODELING AND FITTING —

REGULARIZATION —

EXPLANATION —

DOMAIN ADAPTATION —

A WHIMSICAL STORY ABOUT INDUCTION —

appendices

Here are "refreshers" on programming and on math, especially on the math of high dimensions and of bayes' law. They can help jog your memory of the math and programming preparation you should have done in preparation for 6.86x. They also help establish our naming conventions. The refreshers are meant neither to explain these topics nor to provide a complete outline of prerequisites.

Depending on your background and personality, you might recognize and care that throughout these notes we are a bit sloppy with our probability and programming. No standard compiler can compile our pseudocode; likewise, our theorems are actually theorem sketches in need of additional analytic hypotheses. For readers so concerned, we assume an additional prerequisite: enough technical background that we may leave implicit both the articulation of measure-theory technicalities and the implementation of pseudocode.

```
def get_random_number():
    return 4 # chosen by a fair dice roll
             # guaranteed to be random
```

— xkcd, Randall Munroe, translated to Python

He had bought a large map representing the sea,
Without the least vestige of land:
And the crew were much pleased when they found it to be
A map they could all understand.
— The Hunting of the Snark, Charles Dodgson

python programming refresher

SETUP —

STATE AND CONTROL FLOW —

INPUT/OUTPUT —

NUMPY —

math notation and refresher

PROBABILITY — We've tried to use

$\mathsf{sans\ serif}$ for the names of random variables,

italics for the values they may take, and

$\mathcal{CURLY\ CAPS}$ for sets of such values.

For example, we write $p_{\mathsf{y}|\mathsf{h}}(y|h)$ for the probability that the random variable y takes the value y conditioned on the event that the random variable h takes the value h. Likewise, our notation $p_{\hat{\mathsf{h}}|\mathsf{h}}(h|h)$ indicates the probability that the random variables $\hat{\mathsf{h}}$ and h agree in value given that h takes a value $h \in \mathcal{H}$.

LINEAR ALGEBRA —

CALCULUS —

HIGHER DIMENSIONS —

high dimensional geometry

Can I just say Chris for one moment that I have a new theory about the brontosaurus. ... This theory goes as follows and begins now. All brontosauruses are thin at one end, much, much thicker in the middle and then thin again at the far end. That is my theory, it is mine, and belongs to me and I own it, and what it is too.
— Flying Circus, Monty Python

CONCENTRATION OF MEASURE —

Lemma (Chernoff). *The fraction of heads among N i.i.d. flips of a biased coin exceeds its mean p by g with chance at most $\exp(-Ng^2)$, for $0 \le p < p + g \le 1$.*

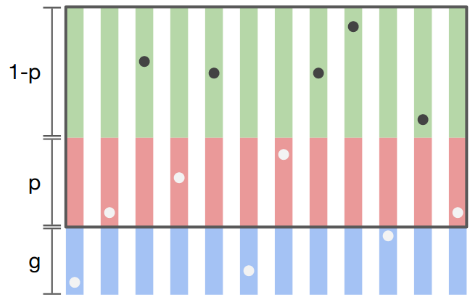


Figure 2: We sample points uniformly at random on N sticks, each with three parts: **green** with length $1-p$, **red** with length p, and **blue** with length g. We call non-blue points **boxed** and non-green points **hollow**.

Proof. Let our coin flips arise from sampling points on sticks (Figure 2), where green means tails, red means heads, and we condition on the event that blues do not occur. To show that less than $(p+g)N = p'N$ flips are heads is to show — given that all points are **boxed** — that less than $p'N$ points are red. For any M:

$$\begin{aligned}&\mathbb{P}[M \text{ are red} \mid \text{all are boxed}]\\ =\;&\frac{\mathbb{P}[\text{all hollows are red} \mid M \text{ hollow}] \cdot \mathbb{P}[M \text{ are hollow}]}{\mathbb{P}[\text{all are boxed}]}\\ =\;&(1-g/p')^M \cdot (1+g)^N \cdot \mathbb{P}[M \text{ are hollow}]\end{aligned}$$

We sum over $M \geq p'N$, bound $\mathbb{P}[\cdots p'N \cdots] \leq 1$, then invoke $(x \mapsto x^{p'})$'s concavity and exp's convexity:

$$\begin{aligned}&\mathbb{P}[\text{at least } p'N \text{ are red} \mid \text{all are boxed}]\\ \leq\;&(1-g/p')^{p'N} \cdot (1+g)^N \cdot \mathbb{P}[\text{at least } p'N \text{ are hollow}]\\ \leq\;&(1-g)^N \cdot (1+g)^N = (1-g^2)^N \leq \exp(-Ng^2) \qquad \square\end{aligned}$$

The Chernoff bound gives us the control over tails we would expect from the Central Limit Theorem, but for finite instead of asymptotically large N. In particular, when we learn from much but finite data, the training error will **concentrate** near the testing error.

Indeed, for any $f \in \mathbb{H}$, $\text{trn}_{\mathcal{S}}(f)$ is the average of N independent Bernoullis of mean $\text{tst}(f)$. So for finite $\mathbb{H}$, the gap is probably small:

$$\begin{aligned}&\mathbb{P}_{S\sim\mathbb{D}^N}[\text{gap}_{\mathcal{S}}(\mathbb{L}) \geq g]\\ \leq\;&\sum_{f\in\mathbb{H}} \mathbb{P}_{S\sim\mathbb{D}^N}[\text{tst}(f) \geq \text{trn}_{\mathcal{S}}(f) + g]\\ \leq\;&|\mathbb{H}| \cdot \exp(-Ng^2)\end{aligned}$$

For example, if $\mathbb{H}$ is parameterized by P numbers, each represented on a computer by 32 bits, then $|\mathbb{H}| \leq 2^{32P}$ and, with probability $1-\delta$, the gap is less than

$$\sqrt{(\log(1/\delta) + 32P)/N}$$

This bound's sensitivity to the description length $32P$ may seem artificial. Indeed, the various $\mathbb{H}$ used in practice — e.g. linear models or neural networks — depend smoothly on their parameters, so the parameters' least significant bits barely affect the classifier. In other words, $\mathbb{H}$'s cardinality is not an apt measure of its size. The VC-dimension measures $\mathbb{H}$ more subtly.

QUADRATIC FORMS —

COVARIANCE, CORRELATION, LEAST SQUARES REGRESSION —

bayesian inference

So little of what could happen does happen.
— salvador dali

CONCEPTUAL FRAMEWORK — We're confronted with an observation or dataset o that comes from some unknown underlying pattern h. We know how each possible value h for h induces a distribution on o and we have a prior sense of which hs are probable. Bayes' law helps us update this sense to account for the dataset by relating two functions of h:

$$\underbrace{p_{h|o}(h|o)}_{\text{posterior}} \propto \underbrace{p_{o|h}(o|h)}_{\text{likelihood}} \cdot \underbrace{p_h(h)}_{\text{prior}}$$

Bayes' law underlies most of our analyses throughout these notes.

Like Newton's $F = ma$, Bayes is by itself inert: to make predictions we'd have to specify our situation's forces or likelihoods. Continuing the metaphor, we will rarely solve our equations exactly; we'll instead make approximations good enough to build bridges and swingsets. Still, no one denies that $F = ma$ orients us usefully in the world of physics. So it is with the law of Bayes.

Formally, we posit a set $\mathcal{H}$ of *hypotheses*, a set $\mathcal{O}$ of possible *observations*, and a set $\mathcal{A}$ of permitted *actions*. We assume as given a joint probability measure $p_{o,h}$ on $\mathcal{O} \times \mathcal{H}$ and a *cost function* $c : \mathcal{A} \times \mathcal{H} \to \mathbb{R}$. That cost function says how much it hurts to take the action $a \in \mathcal{A}$ when the truth is $h \in \mathcal{H}$. Our primary aim is to construct a map $\pi : \mathcal{O} \to \mathcal{A}$ that makes the expected cost $\mathbb{E}_{h,o}\, c(\pi(a); h)$ small.

Below are three examples. In each case, we're designing a robotic vacuum cleaner: $\mathcal{H}$ contains possible floor plans; $\mathcal{O}$, possible readings from the robot's sensors. The examples differ in how they define and interpret $\mathcal{A}$ and c.

A. $\mathcal{A}$ consists of probability distributions over $\mathcal{H}$. We regard $\pi(o)$ as giving a posterior distribution on $\mathcal{H}$ upon observation o. Our cost $c(a; h)$ measures the surprise of someone who believes a upon learning that h is true. Such *inference problems*, being in a precise sense universal, pose huge computational difficulties; we thus often collapse distributions to points, giving rise to the distinctive challenge of balancing estimation error with structural error.

B. $\mathcal{A}$ consists of latitude-longitude pairs, interpreted as a guessed location of the robot's charging station. The cost $c(a; h)$ measures how distant our guess is from the truth. Such *estimation problems* abound in science and engineering; they pose the distinctive challenge of balancing sensitivity-to-misleading-outliers against sensitivity-to-informative-datapoints.

C. $\mathcal{A}$ consists of instructions we may send to the motors, instructions that induce motion through our partially-known room. The cost $c(a; h)$ incentivizes motion into dusty spaces and penalizes bumping into walls. We often compose such *decision problems* sequentially; this gives rise to the distinctive challenge of balancing exploration with exploitation.

EFFECT OF PRIOR CHOICE —

MIXTURE PRIORS AND HIERARCHY —

FREQUENTISM AND CHOICE OF PRIOR — Our engineering culture prizes not just *utility* but also *confidence*, since strong guarantees on our designs allow composition of our work into larger systems: equality, unlike similarity, is transitive. For example, we'd often prefer a 99% guarantee of adequate performance over a 90% guarantee of ideal performance. This asymmetry explains our pessimistic obsession with worst-case bounds over best-case bounds, cost functions over fitness functions, and simple models with moderate-but-estimatable errors over rich models with unknownable-but-often-small errors.

The *frequentist* or *distribution-free* style of statistics continues this risk-averse tradition. In the fullest instance of this style, we do inference as if the true unknown prior on $\mathcal{H}$ is chosen adversarially. That is, we try to find π that makes the following error small:

$$\max_{p_h} \mathbb{E}_{h \sim p_h(\cdot)} \mathbb{E}_{o \sim p_o(\cdot|h)}\, c(\pi(o); h)$$

Intuitively,

p-hacking —

hidden assumptions —

(multiple) hypothesis testing —

Let's now consider the case where $\mathcal{H}$ is a small and finite. We

Table of Contents