mlentary (optional 6.86x notes)

As binocular vision deepens and disambiguates, these notes may aid review of the lectures by giving a complementary organization of concepts.

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 solving homework or exams.

A. Prologue

KINDS 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 (§E shows how this case unlocks the others). Given a list of N examples of properly answered prompts, we seek a pattern: a map from prompts to answers. Say $\mathcal{X}$ is the set of prompts; $\mathcal{Y}$, of answers. Then a program that learns from examples has type:

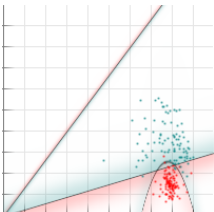$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^N \to (\mathcal{X} \to \mathcal{Y})$$

LEARNING ERROR — Draw examples $\mathcal{S} : (\mathcal{X} \times \mathcal{Y})^N$ from a distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. A pattern $f : \mathcal{X} \to \mathcal{Y}$ has **training error** $\mathrm{trn}_{\mathcal{S}}(f) = \mathbb{P}_{(x,y)\sim\mathcal{S}}[f(x) \neq y]$ and **testing error** $\mathrm{tst}(f) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$. We want low $\mathrm{tst}(\mathcal{L}(\mathcal{S}))$. We often set $\mathcal{L}$ to minimize $\mathrm{trn}_{\mathcal{S}}$ in a candidates set $\mathcal{H} \subseteq (\mathcal{X} \to \mathcal{Y})$. Then tst decomposes into the failures of $\mathrm{trn}_{\mathcal{S}}$ to estimate tst, $\mathcal{L}$ to minimize $\mathrm{trn}_{\mathcal{S}}$, and $\mathcal{H}$ to contain "the" truth:
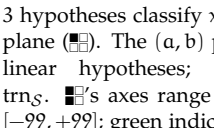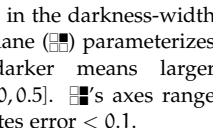
$$
\begin{aligned}
\mathrm{tst}(\mathcal{L}(\mathcal{S})) = \;& \mathrm{tst}(\mathcal{L}(\mathcal{S})) && - \; \mathrm{trn}_{\mathcal{S}}(\mathcal{L}(\mathcal{S})) && \text{\} \textbf{generalization} error} \\
& + \; \mathrm{trn}_{\mathcal{S}}(\mathcal{L}(\mathcal{S})) && - \; \inf_{\mathcal{H}}(\mathrm{trn}_{\mathcal{S}}(f)) && \text{\} \textbf{optimization} error} \\
& + \; \inf_{\mathcal{H}}(\mathrm{trn}_{\mathcal{S}}(f)) && && \text{\} \textbf{approximation} error}
\end{aligned}
$$

These terms are in tension. For example, as $\mathcal{H}$ grows, the approx. error may decrease while the gen. error may increase — this is the "**bias-variance** tradeoff".

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 the digit y. $\mathcal{H}$ contains all "linear hypotheses" $f_{a,b}$ defined by:

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

Our program $\mathcal{L}$ loops over all integer pairs (a, b) in $[-99, +99]$ to minimize $\mathrm{trn}_{\mathcal{S}}$, giving $(a, b) = (-20, 6)$ with $\mathrm{trn}_{\mathcal{S}} \approx 5\%$. We got optimization error ≈ 0 (albeit by *unscalable brute-force*). So approximation error $\approx \mathrm{trn}_{\mathcal{S}} \approx 5\%$. Indeed, our straight lines are *too simple*: width and darkness lose useful information and the "true" boundary looks curved (see ▦'s curve). The testing error $\mathrm{tst} \approx 8\%$ exceeds $\mathrm{trn}_{\mathcal{S}}$: we suffer *generalization error*. In 6.86x we'll address all three italicized issues.

Exercise: how do ▦'s two straight lines and ▦'s two marked points correspond?
Exercise: visualize $\mathrm{trn}_{\mathcal{S}}(f_{a,b})$ in the (a, b) plane for $N = 1$ training example.
Exercise: why is generalization error usually positive?
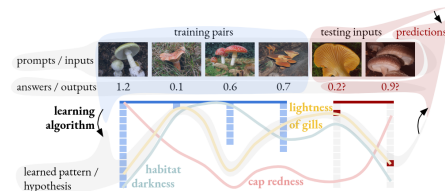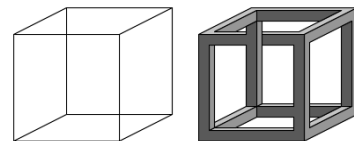
Learning from examples means taking a table of "training" examples as input and producing a "pattern" as output. We evaluate the pattern on "testing" prompts to make predictions.
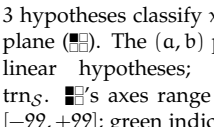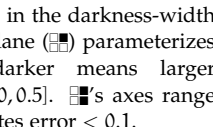
← We'll soon allow uncertainty by letting patterns map to *probability distributions* answers (§B.1); then an example-based learner has type:
$$\mathcal{L} : (\mathcal{X} \times \mathcal{Y})^N \to (\mathcal{X} \to \mathrm{DistributionsOn}(\mathcal{Y}))$$
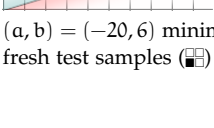Sometimes, the prompt is always the same — say, "produce a beautiful melody" — and we seek (e.g.) to generate many answers. So-called **unsupervised learning** thusly focuses on output structure. **Supervised learning** focuses on the input-output relation.

Six example (x, y)s. An x's *darkness* is its average pixel darkness; its *width* is the stddev column index weighted by column darkness. We normalize both to have max value 1.0:
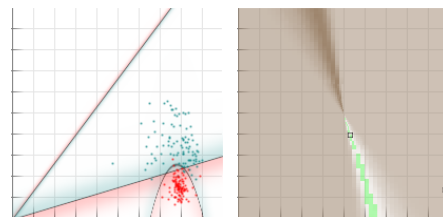
3 hypotheses classify xs in the darkness-width plane (▦). The (a, b) plane (▦) parameterizes linear hypotheses; darker means larger $\mathrm{trn}_{\mathcal{S}}$. ▦'s axes range $[0, 0.5]$. ▦'s axes range $[-99, +99]$; green indicates error < 0.1.
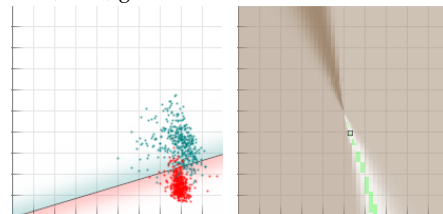
$(a, b) = (-20, 6)$ minimizes $\mathrm{trn}_{\mathcal{S}}$ to 0.05 but on fresh test samples (▦) suffers $\mathrm{tst} = 0.08$.

B. Linear models

0. linear approximations

FEATURIZATION — As in the prologue, we represent our input x as a fixed-length list of numbers so that we can treat x with math. There, we represented each photograph by 2 numbers: width and darkness. We could instead have represented each photograph by 784 numbers, one for the brightness at each of the $28 \cdot 28 = 784$ many pixels. Or by 10 numbers, each measuring the overlap of x's ink with a "standard" photo of the digits 0 through 9.

When we choose how to represent x by a list of numbers, we're choosing a **featurization**. We call each number a "feature". For example, width and darkness are two features.

TODO: mention one-hot, etc TODO: mention LOWRANK for multiregression There are lots of interesting featurizations, each making different patterns easier to learn. So we judge a featurization with respect to the kinds of patterns we use it to learn. TODO: graphic of separability; and how projection can reduce it Learning usually happens more accurately, robustly, and interpretably when our featurization is abstract (no irrelevant details) but complete (all relevant details), compressed (hard to predict one feature from the others) but accessible (easy to compute interesting properties).

Exercise: Beyond width and darkness, what features do you think might help us to separate digits 0 from 1 by a line through the origin? How about 3 from 8?

GEOMETRY OF FEATURE-SPACE —

Caution: a feature $A(x)$ that is statistically independent from y may still be relevant for predicting y. For example, if A, B are two features, it is possible that $A(x), y$ are independent and that $B(x), y$ are independent and yet $A(x), B(x), y$ are *dependent*!

TODO: example featurization (e.g. MNIST again?)

TODO: projectivization (say this foreshadows kernel discussion?)

Now say we've decided on a **featurization** of our input data x.

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

Illustrate 'averaging' of good features vs 'correction' of one feature by another (how much a feature correlates with error)

Example. Consider the uniform distribution on the four corners of a tetrahedron embedded within the corners of a cube TODO: graphic. The three spatial coordinates give three bit-valued random variables. Any two of these variables are independent. But the three together are dependent. TODO: also do a decision boundary (simpsons style) graph illustrating this phenomenon

LINEAR ALGEBRA — Linear algebra is the part of geometry that focuses on when a point is the origin, when a 'line' is a straight, and when two straight lines are parallel. Linear algebra thus helps us deal with the preceding pictures mathematically. The concept of 'straight lines' gives a simple, flexible model for extrapolation from known points to unknown points. That is intuitively why linear algebra will be crucial at every stage of 6.86x.

The elements of linear algebra are **column vectors** and **row vectors**. FILL IN Though we represent the two similarly in a computer's memory, they have different geometric meanings. We save much anguish by remembering the difference.

← It is important to thoroughly understand these basics of linear algebra. Please see §G.2 for further discussion of these basics.

We may **evaluate** a row vector on a column vector. FILL IN A **dot product** is a way of translating between row and column vectors. FILL IN: DISCUSS GENERALIZATION; (DISCUSS ANGLE, TOO)

RICHER OUTPUTS — We've learned how to construct a set $\mathcal{H}$ of candidate patterns

$$f_{\vec{w}}(\vec{x}) = \text{threshold}(\vec{w} \cdot \vec{x})$$

that map (a featurization of) a prompt $\vec{x}$ to a binary answer $y = 0$ or $y = 1$.

What if we're interested in predicting a richer kind of y? For example, maybe there are k many possible values for y instead of just 2. Or maybe there are infinitely many possible values — say, if y is a real number or a length-l list of real numbers. Or maybe we want the added nuance of predicting probabilities, so that f might output "20% chance of label $y = 0$ and 80% chance of label $y = 1$" instead of just "$y = 1$".

I'll write formulas and then explain.

$$f_{\vec{w}_i : 0 \le i < k}(\vec{x}) = \text{argmax}_i(\vec{w}_i \cdot \vec{x})$$

$$f_{\vec{w}}(\vec{x}) = \vec{w} \cdot \vec{x}$$

$$f_{\vec{w}_i : 0 \le i < k}(\vec{x}) = \text{normalize}(\exp(\vec{w}_i \cdot \vec{x}) : 0 \le i < k)$$

1. iterative optimization

(STOCHASTIC) GRADIENT DESCENT — We have a collection $\mathcal{H}$ of candidate patterns together with a function $1 - \text{trn}_\mathcal{S}$ that tells us how good a candidate is. In §A we found a best candidate by brute-force search over all of $\mathcal{H}$; this doesn't scale to our linear models, since now $\mathcal{H}$ is intractably large. So: *what's a faster algorithm to find (or approximate) a best candidate?*

$\leftarrow$ We view $1 - \text{trn}_\mathcal{S}$ as an estimate of our actual notion of "good": $1 - \text{tst}$.

A common idea is to start arbitrarily with some $h_0 \in \mathcal{H}$ and repeatedly improve to get $h_1, h_2, \cdots$. Two questions are: *how do we select h_{t+1} in terms of h_t?* And *how do we know when to stop?* We'll discuss termination conditions later — for now, let's agree to stop at h_{1000}.

As for selecting a next candidate, we'd like to use more detailed information on h_t's inadequacies to inform our proposal h_{t+1}. Intuitively, if h_t misclassifies a particular $(x_n, y_n) \in \mathcal{S}$, then we'd like h_{t+1} to be like h_t but nudged a bit in the direction of accurately classifying (x_n, y_n).

FILL IN FOR PROBABILITIES (LOGISTIC) MODEL

This is the idea of **gradient descent**. MOTIVATE AND GIVE PSEUDOCODE FOR STOCHASTIC GD

LOGISTIC MODELS —

PERCEPTRONS — We get a historically crucial and intuition-pumping algorithm when we do logistic regression under the constraint that the weight vectors be *very long*.

HINGE LOSS AND SVMS — Logistic ain't the only way to go in classification. Here we discuss a generalization and the merits of various special cases.

2. priors and generalization

ON OVERFITTING ——

LOG PRIORS AND BAYES ——

ℓ^p REGULARIZATION; SPARSITY ——

ESTIMATING GENERALIZATION ——

3. model selection

All human beings have three lives: public, private, and secret.
— gabriel garcìa marquez

4. generalization bounds

8

5. ideas in optimization

premature optimization is the root of all evil
— donald knuth

6. kernels enrich approximations

FEATURES AS PRE-PROCESSING —

ABSTRACTING TO DOT PRODUCTS —

KERNELIZED PERCEPTRON AND SVM —

KERNELIZED LOGISTIC REGRESSION —

... animals are divided into (a) those belonging to the emperor; (b) embalmed ones; (c) trained ones; (d) suckling pigs; (e) mermaids; (f) fabled ones; (g) stray dogs; (h) those included in this classification; (i) those that tremble as if they were mad; (j) innumerable ones; (k) those drawn with a very fine camel hair brush; (l) et cetera; (m) those that have just broken the vase; and (n) those that from afar look like flies.
— jorge luis borges

An Intermezzo

Congrats! If you've digested all the above, then you've digested by far important part of these notes. I invite you to take a break, make a sandwich, drink some coffee, enjoy a walk, before proceeding.

To accompany your break, below is some poetry generated by a rudimentary language model. You'll be able to completely understand how it works; that said, it doesn't work well. The tools of the next section can drastically improve this model.

METHOD —

OUTPUT —

C. Nonlinearities

0. fixed featurization

1. learned featurizations

2. differentiation

3. architecture and symmetry

4. feature hierarchies

5. stochastic gradient descent

6. loss landscape shape

The virtue of maps, they show what can be done with limited space, they foresee that everything can happen therein.
— josé saramago

An Intermission

D. Structured inference

0. graphical generative models

1. inferring conditional marginals

2. learning the parameters

3. hierarchy and mixtures

I like crossing the imaginary boundaries people set up
between different fields — it's very refreshing.
— maryam mirzakhani

4. hierarchy and transfer

5. variational and sampling methods

Take a chance, won't you? Knock down the fences that divide.
— thurgood marshall

6. amortized inference

A Divertisment

E. Reductions to supervision

0. to build a tool, use it

1. distributions as maps

2. self-supervised: downstream tasks

3. self-supervised: autoregression

4. reinforcement: exploration-exploitation

5. reinforcement: states and q-values

There's a good reason why nobody studies history: it just teaches you too much.
— noam chomsky

6. beyond the i.i.d. hypothesis

OUT-OF-DISTRIBUTION TESTS —

DEPENDENT SAMPLES —
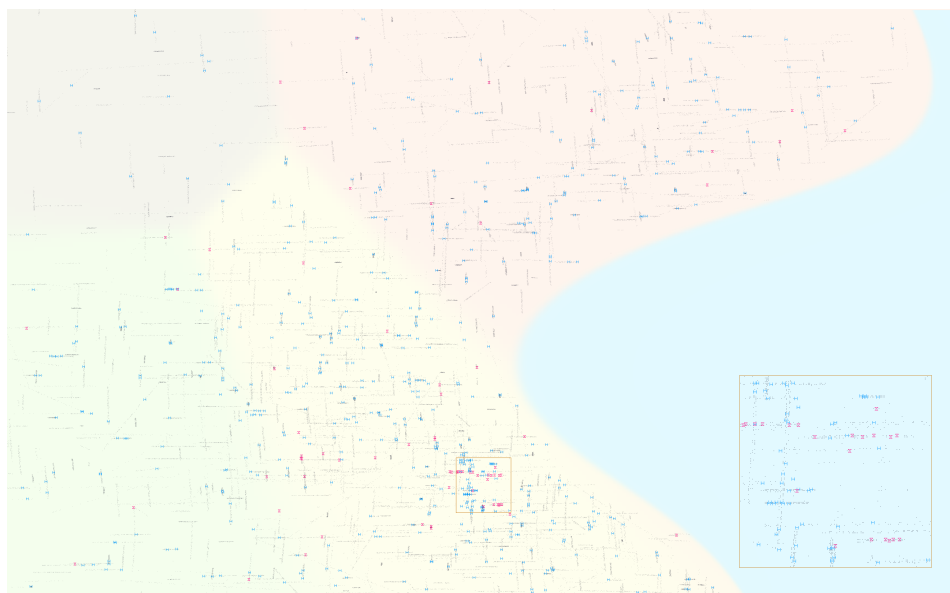
CAUSALITY —

A Farewell

Congrats!

FURTHER READING —

F. Three brief example projects

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

1. flooding plains

VISUALIZATION —

Figure 1: In light yellow and blue are land and sea: 6.4km × 10.4km around the city of Zembla. Gray dots: untested faucets. Red bowties: faucets tested positive. Blue I-beams: faucets tested negative. The small yellow box surrounds a neighborhood that enjoyed more thorough testing; the large yellow box in the lower right magnifies that extra-tested region. TODO: number of gray dots, among labels: training set vs hidden

FITTING —

MODEL SELECTION —

PREDICTION —

OVERFITTING —

A WHIMSICAL STORY ABOUT AUTOMATION — We automate when we re-cast the previous generation's *methods* as mere *data* — 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
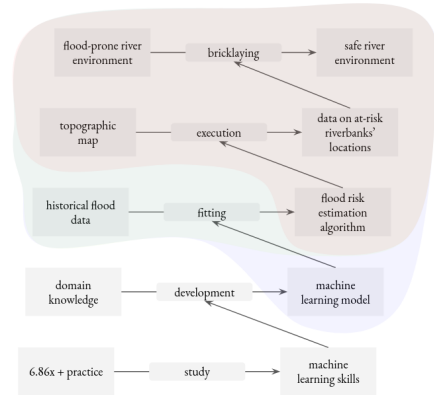
Figure 2: Machine learning continues our human tradition toward the systematic and automatic. What for one generation is a luckily discovered idea or recipe (bottom tip of red), the next generation refines by careful human thought (green). What to one generation is a task (green) requiring human thought is to a future generation merely a routine parameterized by some newly discovered recipe (bottom tip of blue). And the cycle repeats.

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.

1. ancient tablets

2. pairing flavors

Appendices

Here are "refreshers" on programming and on math, especially on the math of high dimensions and of bayes' law. Beyond establish our naming conventions, they can help jog your memory of the math and programming you might have encountered prior to taking 6.86x. The refreshers are meant neither to explain their content well nor to provide a complete outline of skills needed for 6.84x. For further support, please ask questions on the forum and make frequent use of office hours!

python programming refresher

If I have not seen as far as others, it is because giants were standing on my shoulders.
— hal abelson

SETUP — Python is a popular programming language. Beyond the Python interpreter, there is an ecosystem of useful tools: machine learning modules that help us avoid reinventing the wheel; a rainbow of feature-rich IDEs to choose from; and a package manager called `conda` that eases the dependency-spaghetti hassle often present in installing the right version of stuff.

Let's set up Python3.

I'll assume we're working on Linux (specifically, Ubuntu 20 or related). If you're on a different Linux or on MacOS, then similar steps should work — google search to figure out how. If you're on Windows 10 or higher, then similar steps should work after you set up the "Linux Subsystem for Windows."

FILL IN: how to run, comments, basic stdIO line by line HELLO WORLD; string type line by line FAHRENHEIT KELVIN CELSIUS; float and integer types

STATE AND CONTROL FLOW —

finite state automata; skip, abort variable cube; assignment sequencing conditionals iteratives nesting, (scope?,) indentation

ROUTINES — lambdas def'd functions (ordinary args, return) interfaces: (kwargs; None as default return value; sentinels; more) code architecture and hygiene

Once you learn how to program, I highly recommend reading Edsger Dijkstra's *A Discipline of Programming*. That book contains many beautiful viewpoints that made me a better programmer.

DATA IN AGGREGATE — lists and numpy arrays dictionaries comprehensions functional idioms

INPUT/OUTPUT — print formatting and flushing basic string manipulations (strip, join, etc) file io command line arguments example: read csv random generation and seeds??

HOW NOT TO INVENT THE WHEEL — matplotlib.plot numpy os package managers etc

ARRAYS AKA TENSORS —

probability refresher

We've tried to use

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 $p_{y|h}(y|h)$ 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 $p_{\hat{h}|h}(h|h)$ indicates the probability that the random variables $\hat{h}$ and h agree in value given that h takes a value $h \in \mathcal{H}$.

linear algebra refresher

Linear algebra is the part of geometry that focuses on when a point is the origin, when a 'line' is a straight, and when two straight lines are parallel. Linear algebra thus helps us deal with the preceding pictures mathematically. The concept of 'straight lines' gives a simple, flexible model for extrapolation from known points to unknown points. That is intuitively why linear algebra will be crucial at every stage of 6.86x.

← It is important to thoroughly understand these basics of linear algebra. Please see §G.2 for further discussion of these basics.

The elements of linear algebra are **column vectors** and **row vectors**. FILL IN Though we represent the two similarly in a computer's memory, they have different geometric meanings. We save much anguish by remembering the difference. FILL IN

FILL IN LINEAR DECISION BOUNDARY! (remark on featurization and argmax nonlinearities)

We may **evaluate** a row vector on a column vector. FILL IN A **dot product** is a way of translating between row and column vectors. FILL IN: DISCUSS GENERALIZATION; (DISCUSS ANGLE, TOO)

FILL IN COMPUTATION AND BASES

VISUAL ILLUSTRATION OF HOW CHOICE OF DOT PRODUCT MATTERS

mathematical language refresher

SPACE — René Descartes had a good idea: we can study the invisible and intangible by hacking our visual cortex and our proprioceptive senses. We do this through the profound metaphor of **SPACES of BLAH**, where BLAH is some kind of thing we're analyzing

In the next two passages, we discuss two mental moves we like to do with this spatial metaphor. The first simplifies our thinking by forgetting irrelevant truths. The second simplifies our thinking by positing useful falsehoods.

SAMENESS — The key challenges here are of *uniqueness* and *lifting*

SPECIALNESS — The key challenges here are of *existence* and *extension*

TYPES — In technical subjects, it is very useful to keep track of what "types" our objects of study are. This reduces cognitive load by reminding us what we're allowed to do with the objects.

We write "$x : S$" (or, synonymously, "$x \in S$") as shorthand for the sentence: *the thing we're calling x has the type we're calling S.*

← In many math books the two are not synonymous. But this ain't a math book.

We can build new types from old through: if X, Y are two types, then $X \sqcup Y$ names the type of things that are either of type X or of type Y. Meanwhile, $X \times Y$ names the type of pairs (x, y) where $x : X$ and $y : Y$. Finally, $X \to Y$ names the type of functions from X to Y. So $f : X \to Y$ means that f has type $X \to Y$.

We have functions left $: X \to (X \sqcup Y)$ and right $: Y \to (X \sqcup Y)$ and first $: (X \times Y) \to X$ and second $: (X \times Y) \to Y$.

We have $\text{left}(x) : X \sqcup Y$ if and only if $x : X$ and $\text{right}(y) : X \sqcup Y$ if and only if $y : Y$. We have $z : X \times Y$ if and only if $\text{first}(z) : X$ and $\text{second}(z) : Y$.

We often consider natural numbers as types; the natural number n, when viewed as a type, has exactly n many elements. Thus, $X \sqcup 1$ is a type of things that are either members of X or the unique member of 1.

TRANSFORMS — When a mathematician says "**function**", they are talking about a gigantic (often infinitely large) table of input-output pairs. The table assigns to each input exactly one output.

There are infinitely many functions. One example is

$$f : \mathbb{R} \to \mathbb{R} \text{ defined by } x \mapsto x^2$$

The above line of math is standard short-hand for the following paragraph:

Consider a big table of input-output pairs. Each input is a real number. Each output is a real number. If x is a real number, then there is exactly one input-output pair whose input is x. Moreover, the output of that pair is the real number x^2. We'll call this big table "f". We'll write "$f(x)$" to refer to the output corresponding to an input x; in this case, $f(x)$ is another name for x^2. For example, $f(5)$ is 25.

TUPLES AND TAGS —

calculus refresher

DERIVATIVES — When we find the *derivative* of a function, we're finding a best linear approximation to that function near a point on that function's graph. Said another way, a derivative $D(f)(x)$ of a function f at an input x is a linear function $h \mapsto D(f)(x)(h)$ such that

$$f(x + h) \approx f(x) + D(f)(x)(h) \text{ for all small } h, \text{ for all } x$$

For example, if $f(x) = 1 + 3x$ then $D(f)(x)$ is the map $h \mapsto 3h$. If $f(x) = x^2$ then $D(f)(x)$ is the map $h \mapsto 2x \cdot h$.

TODO: perimeter graphic for x^2's derivative

DISCUSS Linearity in f
DISCUSS Chain Rule
DISCUSS Product Rule

MINIMA —

EXPONENTIAL FUNCTION — Much as we might try to solve "algebraic equations" (if $y^3 + y = t^2 + 1$, what is y in terms of t?), we sometimes also want to solve "differential equations" (if $Dy = -t \cdot y$, what is y in terms of t?).

From the world of algebraic equations, we might expect two subtleties:

(a) sometimes, we won't be able to write the answer in terms of the familiar arithmetic operations of addition, subtraction, multiplication, and division; what we do instead is to express the answers to a diversity of harder problems in terms of answers to a few easier problems. For example, $y^2 = t$ has no "arithmetic" solution; instead, we just give a name ("sqrt") to that function. We can solve more complicated equations, e.g. $y^2 = t + y$, using the arithmetic operations together with sqrt. This re-expression may seem like cheating; its upshot is that by building intuition for the one new function sqrt, we end up getting insight into lots of more complicated equations. In the world of differential equations, we likewise introduce a few new functions. The most important is called **the exponential function**, or exp for short. Just as sqrt is defined to solve $y^2 = t$, our function exp is defined to solve $Dy = y$.

(b) sometimes, there are multiple solutions. For example, both sqrt(5) and $-$sqrt(5) solve $y^2 = 5$. What we do in algebra is to choose one kind of solution as the thing we give a name to, and we remember a simple rule for how all the other solutions relate to this standard one. Thus, we define sqrt as the continuous solution to $y^2 = t$ *that at input one has output one*: $y(1) = 1$. We get the other solution if we scale the output by -1. In the world of differential equations, there is a similar ambiguity. If $y = \exp(t)$ solves $Dy = y$, then so does $y = 2 \cdot \exp(t)$: we can scale the output by any number to get another solution. We address this scaling ambiguity by defining exp as the solution to $Dy = y$ *that at input zero has output one*: $y(0) = 1$. We get the other solutions if we scale the output.

This definition works just as well: sqrt is the solution to $y^2 = t$ *that's always at least zero.*

Okay, so that's our golden definition:

$$D \exp(t) = \exp(t) \qquad \exp(0) = 1$$

INSERT graph of exp Give a sense of scale: what is exp at $-10, -2.3, -0.69, -1, 0, +1, +0.69, +2.3, +10$?

All solutions of $Dy = t$ are multiples of this solution.

From the previous paragraph flow three facts about exp that'll help us in 6.86x. Here they are, with non-standard and suggestive names:

(a) Entropy Law

$$\exp(a + b) = \exp(a) \cdot \exp(b)$$

To see this fact, just note that we can horizontally shift exp's graph by, says, 5 to get the graph of a function defined by $t \mapsto \exp(5 + t)$. Since derivatives are just best linear approximations, and since a graph's best linear approximations simply shift along with the graph, we have $D \exp(5 + t) = \exp(5 + t)$. So this shifted function *also* solves $Dy = y$. Therefore, it must be a multiple of our standard solution exp. Which multiple? Well, at $t = 0$ we have $\exp(5 + t) = \exp(5)$ while $\exp(t) = 1$. So we want to scale $\exp(t)$ by the number $\exp(5)$ to get $\exp(5 + t)$. We conclude:

$$\exp(5 + t) = \exp(5) \cdot \exp(t)$$

(b) Logarithms

$$\exp()$$

(c) Tails

$$\exp()$$

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.
— john cleese

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 3: 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 3), 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:

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

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

$$\mathbb{P}[\text{at least } p'N \text{ are red} \mid \text{all are boxed}]$$
$$\le (1 - g/p')^{p'N} \cdot (1 + g)^N \cdot \mathbb{P}[\text{at least } p'N \text{ are hollow}]$$
$$\le (1 - g)^N \cdot (1 + g)^N = (1 - g^2)^N \le \exp(-Ng^2) \quad \square$$

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 $f \in \mathbb{H}$, $\mathrm{trn}_{\mathcal{S}}(f)$ is the average of N independent Bernoullis of

mean $\mathrm{tst}(f)$. So for finite $\mathbb{H}$, the gap is probably small:

$$\mathbb{P}_{S\sim\mathbb{D}^N}\left[\mathrm{gap}_S(\mathbb{L}) \geq g\right]$$
$$\leq \sum_{f\in\mathbb{H}} \mathbb{P}_{S\sim\mathbb{D}^N}\left[\mathrm{tst}(f) \geq \mathrm{trn}_S(f) + g\right]$$
$$\leq |\mathbb{H}|\cdot\exp(-Ng^2)$$

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 dalí

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.

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.

← 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.

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 unknowable-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