STOCHASTIC OPTIMIZATION (AKA MACHINE LEARNING)

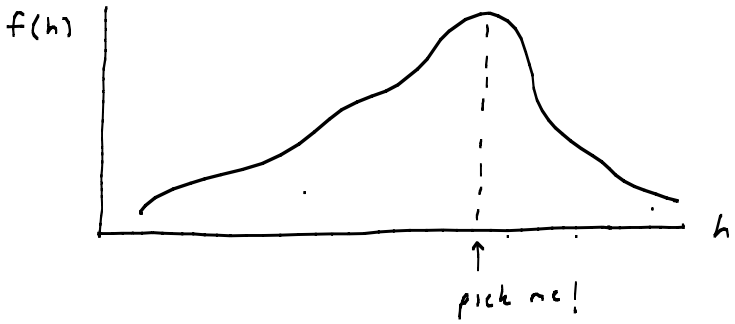
15-359 S12
SHIVA KAUL
SKIKAUL@CS.CMU.EDU

You have probably encountered a deterministic optimization problem...

$$\text{maximize } f(h) \quad \text{s.t. } h \in H$$

where f is deterministic and H is some set. This task is computational.

...if not, now you have.



In stochastic optimization:

- f is a RANDOM function drawn from some distribution $\mathcal{D}$.

The distribution to the right places more probability on the thicker functions.

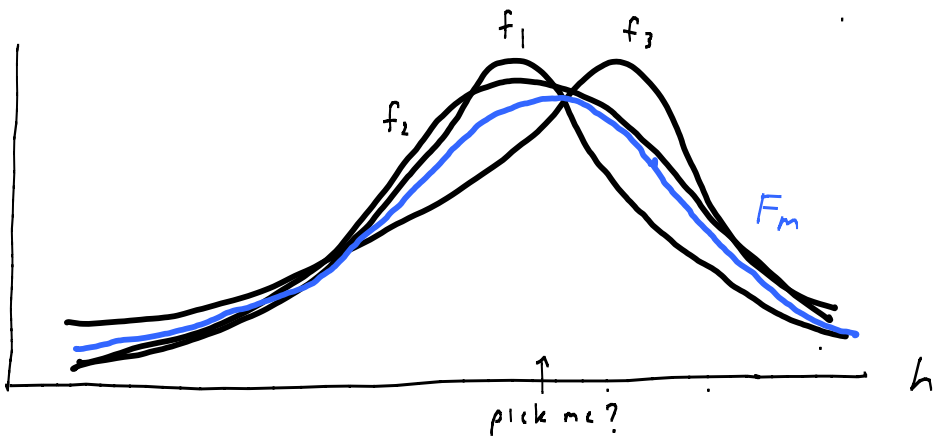
Now the computational task is a bit more complicated:

$$\text{maximize } \mathop{E}_{f \sim \mathcal{D}}[f(h)] \quad \text{s.t. } h \in H$$

- we don't know the distribution D. All we have is an iid sample

$f_1, \ldots, f_m \sim D$

now the problem is both computational and <u>statistical</u>; we are now estimating the true optimum from limited data.



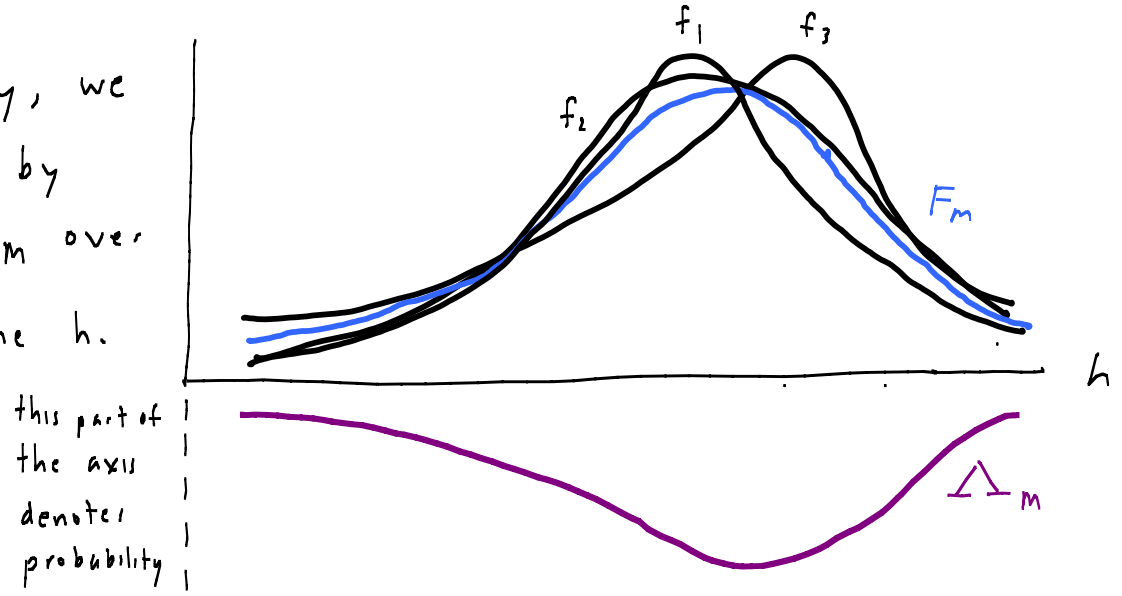
Perhaps a reasonable estimator is the solution to the deterministic problem:

maximize $\hat{F}_m(h) = \frac{1}{m} \sum_{i=1}^{m} f_i(h)$ s.t. $h \in \mathcal{H}$

the 'empirical objective'

We'll study that later. One more change to the optimization setup:

- Due to this uncertainty, we may hedge our estimate by returning a distribution Δ_m over h's rather than just one h.



Let's overload the notation to accommodate such distributions:

$$f(\Lambda_m) = \mathop{\mathbb{E}}_{h \sim \Lambda_m}\left[f(h) \right]$$

Now let's state the stochastic optimization problem:

> given $f_1, .., f_m \sim D$
>
> maximize $F(\Lambda_m) = \mathop{\mathbb{E}}_{f \sim D}\left[f(\Lambda_m) \right]$ s.t.

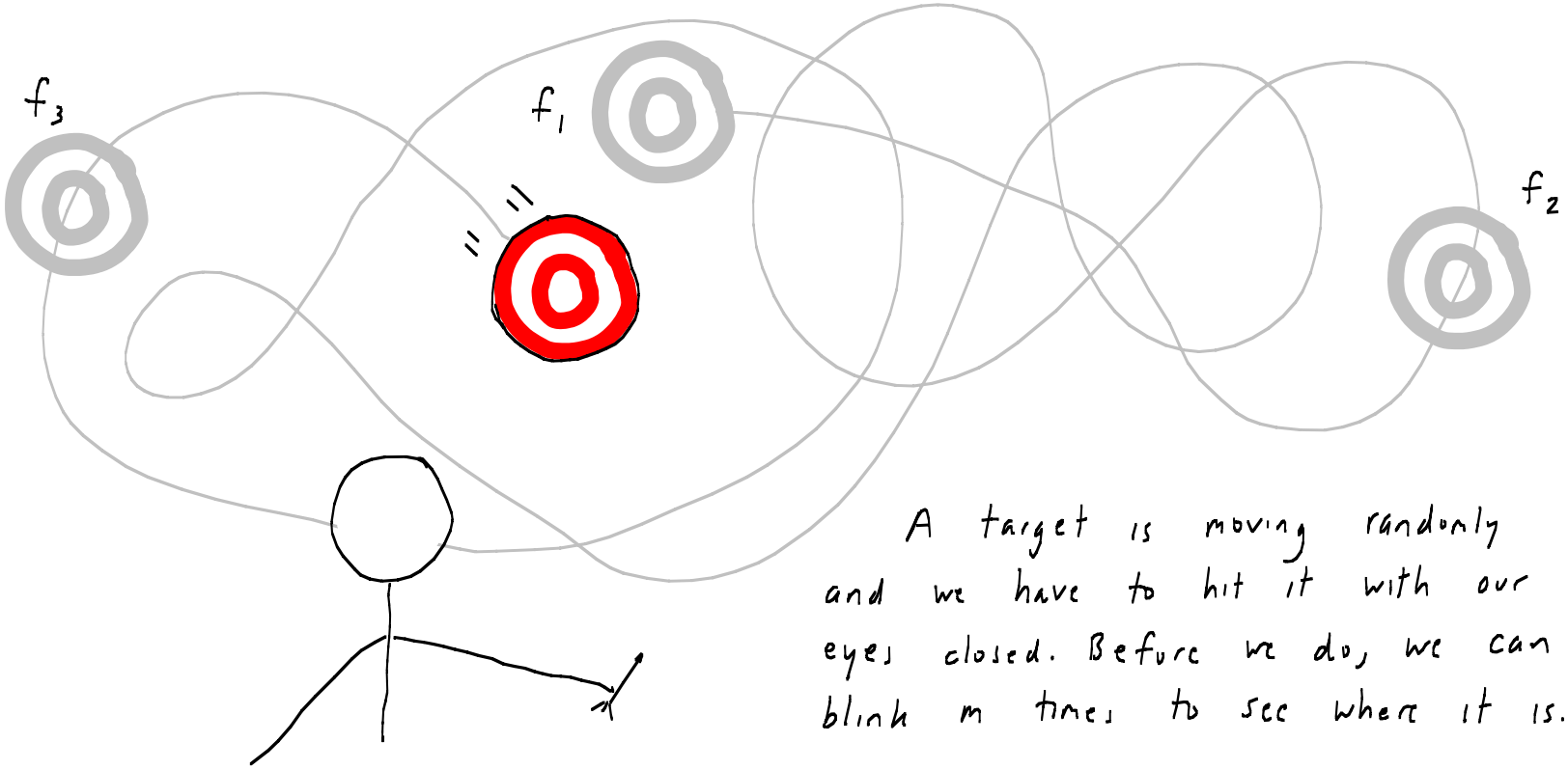
In machine learning terminology,

- D is fixed by nature.
- Each h is a hypothesis about how nature works.

It turns out most of machine learning is stochastic optimization.
But stochastic optimization also includes such important tasks as.....

... Playing blind carnival games!

Nifty example due to Geoff Gordon.



A target is moving randomly and we have to hit it with our eyes closed. Before we do, we can blink m times to see where it is.

$\mathcal{H}$ = places to throw dart

$$f^{(x)}(h) = \begin{cases} 1 & \text{if a dart at } h \text{ hits the target at } x \\ 0 & \text{otherwise} \end{cases}$$

D = distribution over target positions x

---- BEGIN RANT -----

Humans are equipped with many instincts and heuristics which help us model and understand the world. Or, cynically, humans are blinded by vicious biases and prejudices. In pop culture (and unfortunately some pop science), these mental features have corrupted our feeble attempts at rational scientific inquiry, and machine learning is our salvation. Big data, some say, will sweep away our pathetic 'models' and usher in Pax Algorithmica.

Tons of data, fast computers, and clever algorithms are great. Nonetheless, good old-fashioned hunches / intuitions / models are an important and enduring aspect of machine learning.

(Or, cynically, we're still doomed, even with data and computers.)

--- END RANT ----

Our choice of Λ will be good (by the triangle inequality) if:

#1. $F_m(\Lambda_m)$ is small (the aforementioned estimator ensures this)

#2. The distance between $F_m(\Lambda_m)$ and $F(\Lambda_m)$ is small

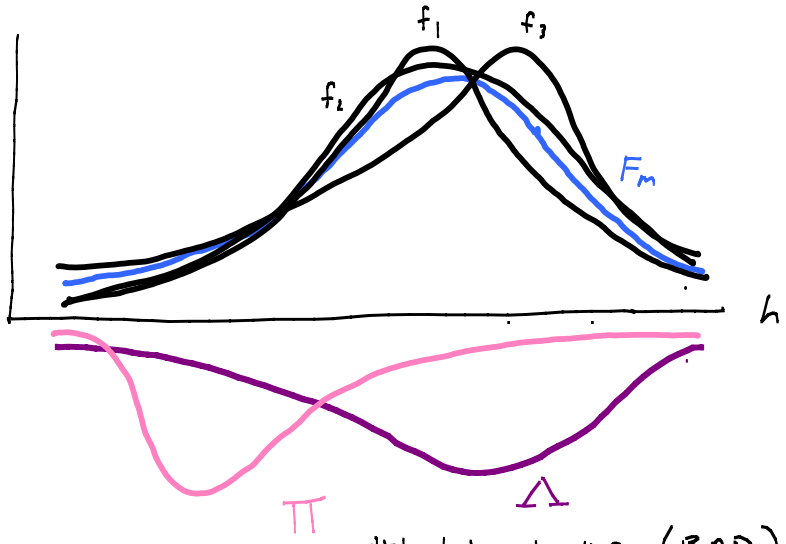
Intuitively, the bigger m is, the better.

The following methodology (too abstract to be called an algorithm) highlights the role of hunches.
(We'll describe it conceptually, then quantitatively. Then we'll prove it works)

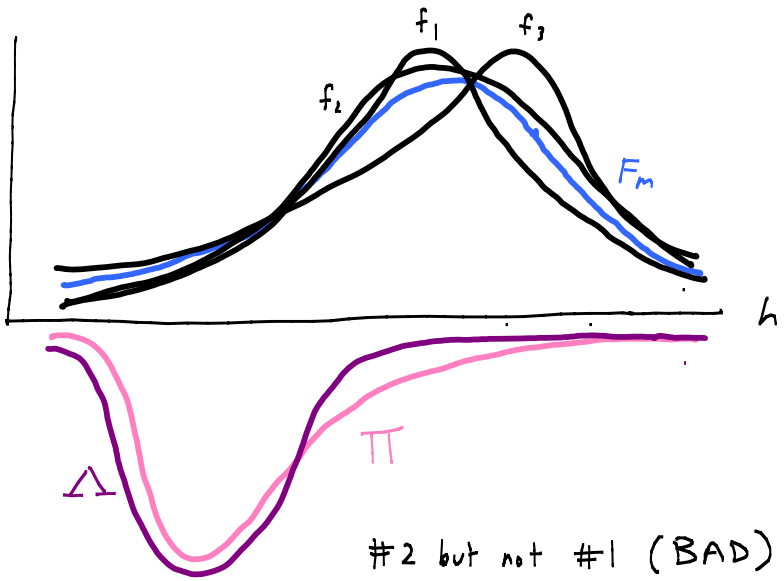
- start off with a hunch in the form of a distribution Π over H. 'Reasonable' h's get more probability; 'silly' h's get less.
- receive the data $f_1, \ldots, f_m$. With probability δ, the following steps don't work.
- for any Λ we choose, these relationships hold (we'll prove this!)

#2 distance of $F_m(\Lambda_n)$ from $F(\Lambda_n)$ $\searrow$ if • distance of Λ_m from Π $\searrow$ • m $\nearrow$

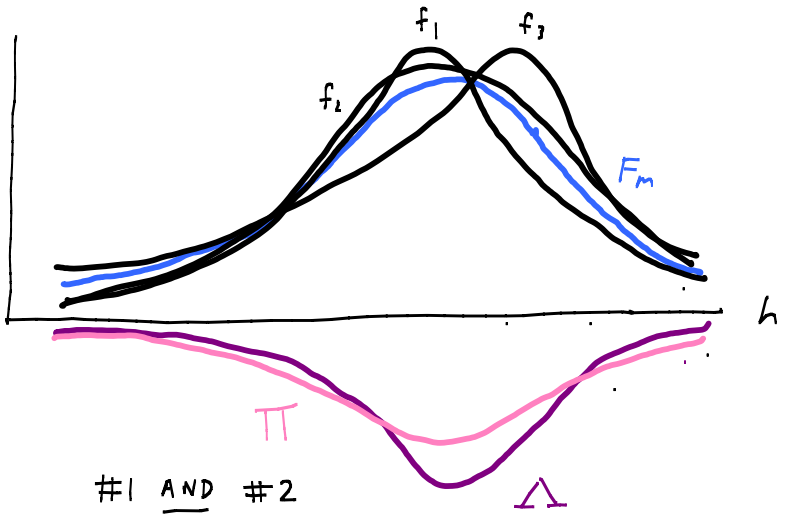
- strike a balance between #1 and distance of Λ_m from Π. (a purely computational task)



#1 but not #2 (BAD)



#2 but not #1 (BAD)



#1 AND #2
(GOOD)

← a good hunch Π is rewarded.

We will quantify the notions of 'distance' or 'closeness' with KL divergence.

$F_m(\Lambda)$ vs. $F(\Lambda)$:

Remember binary KL-divergence for $q, p \in \{0,1\}$:

$$kl[q,p] = q \ln\left(\frac{q}{p}\right) + (1-q)\ln\left(\frac{1-q}{1-p}\right)$$

If $\forall f,h$, $f(h) \in \{0,1\}$ (like in the dartboard example),
then $F_m(h), F(h) \in \{0,1\}$, so $kl[F_m(h), F(h)]$ makes sense.

Λ vs. Π:

Remember how $kl[q,p]$ is a special case of a 'distance' (sort of) between distributions? We'll use that.

$$KL[\Lambda, \Pi] = \mathop{E}_{h\sim\Lambda}\left[\ln\left(\frac{\Lambda(h)}{\Pi(h)}\right)\right]$$

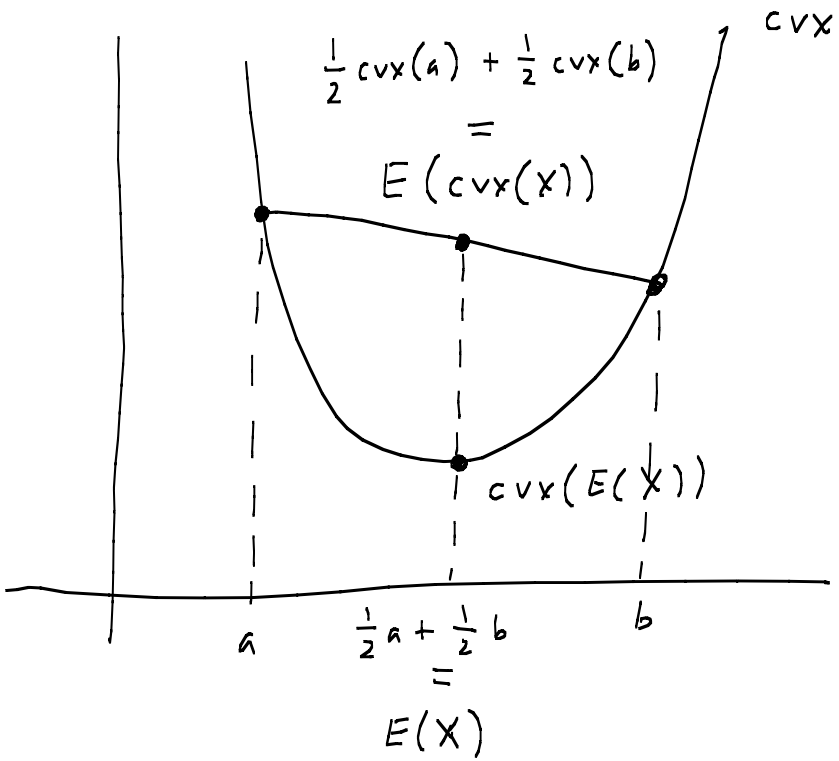
Here's our final methodology, called PAC-Bayes for reasons I won't go into. David McAllester came up with it. I believe this simplified/improved form is by John Langford. ← CMU Grad

- Assumption: $f(h) \in \{0, 1\}$
- start off with a hunch in the form of a distribution π ("prior") over H. 'Reasonable' h's get more probability; 'silly' h's get less.
- receive the data $f_1, \ldots, f_m$. With probability δ, the following steps don't work.
- for any Δ ("posterior") we choose,

$$kl\left[F_M(\Delta_m), F(\pi)\right] \leq \frac{KL\left[\Delta_m, \pi\right] + \ln\left(\frac{m+1}{\delta}\right)}{m}$$

- strike a balance between $F_m(\Delta_m)$ and $KL\left[\Delta_m, \pi\right]$ (a purely computational task)

To prove it, we need Jensen's inequality.



If cvx is a convex function and X is a random variable,

$$cvx(E(X)) \leq E(cvx(X))$$

To remember the direction of the inequality, keep the illustrated special case in mind.

↙

$$X = \begin{cases} a & \text{with probability } 1/2 \\ b & \text{otherwise} \end{cases}$$

Our proof follows the presentation of Ofer Dekel. Define $Z = \mathrm{E}_{h\sim\Pi}(\exp\left(m \cdot \mathrm{KL}\left(F_m(h), F(h)\right)\right))$, a random variable with respect to $S \sim D^m$. We'll prove this theorem in two steps.

1. With probability at least $1-\delta$, $\mathrm{KL}(F_m(\Lambda), F(\Lambda)) \leq \frac{\mathrm{KL}(\Lambda,\Pi)+\ln\left(\frac{\mathrm{E}_{S\sim D^m}(Z)}{\delta}\right)}{m}$
2. $\mathrm{E}_{S\sim D^m}(Z) \leq m+1$

1. By Markov's inequality,

$$\frac{\mathrm{E}_{S\sim D^m}(Z)}{a} \geq \mathrm{P}(a < Z)$$

Plug in $a = \frac{\mathrm{E}_{S\sim D^m}(Z)}{\delta}$ to get

$$\delta \geq \mathrm{P}\left(\frac{\mathrm{E}_{S\sim D^m}(Z)}{\delta} < Z\right)$$

Inverting this, we have with probability at least $1-\delta$,

$$\begin{aligned}
\frac{\mathrm{E}_{S\sim D^m}(Z)}{\delta} &\geq Z && \text{(above)}\\
\ln\left(\frac{\mathrm{E}_{S\sim D^m}(Z)}{\delta}\right) &\geq \ln(Z) && \text{(monotonicity of log)}\\
&= \ln\left(\mathrm{E}_{h\sim\Pi}(\exp(m \cdot \mathrm{KL}(F_m(h), F(h))))\right) && \text{(definition)}\\
&= \ln\left(\mathop{\mathrm{E}}_{h\sim\Lambda}\left(\frac{\Pi(h)}{\Lambda(h)}\exp(m \cdot \mathrm{KL}(F_m(h), F(h)))\right)\right) && \text{(change of measure)}\\
&\geq \mathop{\mathrm{E}}_{h\sim\Lambda}\left(\ln\left(\frac{\Pi(h)}{\Lambda(h)}\right) + m \cdot \mathrm{KL}(F_m(h), F(h))\right) && \text{(concavity of log, Jensen's, log of product)}\\
&= -\mathrm{KL}(\Lambda,\Pi) + m \cdot \mathrm{E}_{h\sim\Lambda}(\mathrm{KL}(F_m(h), F(h))) && \text{(log of quotient, linearity, KL definition)}\\
&\geq -\mathrm{KL}(\Lambda,\Pi) + m \cdot \mathrm{KL}(F_m(\Lambda), F(\Lambda)) && \text{(convexity of KL, Jensen's)}
\end{aligned}$$

Rearrange the terms to get part 1. Part 2 is on the homework. $\square$

Now we have a methodology for stochastic optimization. What do we use it for, and how do we derive algorithms?

An important application is *stochastic supervised learning*, which solves an abstract prediction task: given some observation x (from a 'feature' space $\mathcal{X}$), choose an outcome y (from the outcome space $\mathcal{Y}$). A loss function $\ell(y, \hat{y}) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ quantifies how bad it is to choose $\hat{y}$ when the actual outcome is y. (Higher values of ℓ are worse). Some examples:

- Passing/failing P&C: given a student's homework and quiz grades, predict if they will pass the final exam. Here $\mathcal{X} = \mathbb{R}^d$ where d is the number of assignments/quizzes. $\mathcal{Y} = \{0, 1\}$ encodes 'fail' or 'pass'. $\ell(y, \hat{y}) = 1$ if $y \neq \hat{y}$ and 0 otherwise. We call this setting of $\mathcal{Y}$ and ℓ a classification task.
- Counting faces: given an image, count how many human faces there are. $\mathcal{X}$ is the set of all images, $\mathcal{Y}$ is the set of nonnegative integers, and $\ell(y, \hat{y}) = |y - \hat{y}|$.

In stochastic supervised learning, we assume there is a joint, unknown probability distribution D over the space $\mathcal{X} \times \mathcal{Y}$ (i.e. over random pairs (x, y).) We seek an $h : \mathcal{X} \to \mathcal{Y}$ (called a hypothesis) with low risk

$$\ell(h; D) := E_{(x,y)\sim D}(\ell(h; (x, y)))$$

where

$$\ell(h; (x, y)) = \ell(y, h(x))$$

To guide our choice of h, we are given training data:

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$$

where each (x_i, y_i) is drawn independently from D. (Note that S is a set, so the order of elements doesn't matter.) A learning algorithm A takes this training data and returns a hypothesis. This usually involves fitting a hypothesis to the training data - in machine learning parlance, minimizing the empirical risk

$$\ell(h; S) = \mathrm{E}_{(x,y)\sim S}\ell(h; (x, y)) = \frac{1}{m}\sum_{i=1}^{m} \ell(h; (x_i, y_i))$$

As I said, this is a special case of stochastic optimization: take $f(h) = \ell(y, h(x))$.

How do we derive algorithms? The bound we proved has two especially helpful properties:

- It applies simultaneously to all Λ. A learning algorithm can consider many different Λ and pick one with a good bound.
- If you don't really have a hunch, that's (sometimes) still OK. Choose a Π which isn't too concentrated. When m is small, even if there is an h with small $F_m(h)$, we won't want Λ to put a lot of probability on it, because then the stability term $\mathrm{KL}(\Lambda;\Pi)$ will be large. (Π is shielding us from overfitting.) However, as m increases, the stability term becomes less important, so we can afford to make Λ more concentrated in order to achieve low $F_m(\Lambda)$.