ECE 0402 - Pattern Recognition

LECTURE 2

Today: First look at the theory of generalizations for the binary supervised classification problem.

Supervised Learning: Given training data $(x_1, y_1), ..., (x_n, y_n)$, we would like to learn a function $f: X \mapsto Y$ such that $f(x) = y$ for x other than $x_1, ..., x_n$

Without any additional assumptions, we conclude nothing about f except for its value on this finite set inputs $x_1, ..., x_n$ – NOT so correct!

Probabilistic perspective:

- Any f agreeing with the training data may be **possible**.
- But that doesn't mean that any f is equally **probable**

  Example: $P[heads] = p,\ P[tails] = 1 - p$

  - toss the coin n times (independently)
  - $\hat{p} = \frac{\#of\ \ heads}{n}$
  - does $\hat{p}$ tell us anything about p?
  - for large n we expect $\hat{p} \approx p$
  - Law of large of numbers:

$$\hat{p} \to p \quad as \quad n \to \infty$$

  - we can learn something about p from observations– at least in a very limited sense
  - there is always the **possibility** that we are totally wrong ($\hat{p} \neq p$), but given enough data, the **probability** should be very small.

  coin tosses: we want to estimate p

  learning: we want to estimate a function $f: X \mapsto Y$

Suppose we have hypothesis h and think of the (x_i, y_i) as series of independent coin tosses where (x_i, y_i) are drawn from a probability distribution

- heads: our hypothesis is correct, i.e., $h(x_i) = y_i$
- tails: our hypothesis is wrong, i.e., $h(x_i) \neq y_i$

Definition:

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

$$\text{Empirical Risk} : \hat{R}_n(h) := 1/n \sum_{i=1}^{n} 1_{\{h(x_i) \neq y_i\}}$$

Risk is something we would like to know but to know this, we would have to know the distribution generates all of our data–lots of stuff that we probably don't know in a typical problem...So have this other thing, called Empirical Risk, it depends on the number of observations we get.

The law of large numbers guarantees that as long as we have enough data, we will have that $R(h) \approx \hat{R}_n(h)$. This means that we can use $\hat{R}_n(h)$ to verify whether h was a good hypothesis

- where did h come from?
- what if $R(h)$ is large?

Now consider, ensemble of many hypothesis $\mathcal{H} = h_1, ..., h_m$. If we fix h_j before drawing our data, then the LLN tells us that $\hat{R}_n(h_j) \to R(h_j)$. 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 $\hat{R}_n(h_k)$ is still very far from $R(h_k)$.

Example:

- Toss a fair coin 10 times, the probability of 10 heads: 0.001
- Toss 1000 fair coins 10 times, the probability that **some** coin will get 10 heads: 0.624

This phenomenon forms the fundamental challenge of multiple hypothesis testing. So the question is: how to adapt our approach to handle many hypothesis?

Assumption: There is some underlying function $f: X \mapsto Y$ that captures some input-output relationship that we would like to estimate. We do not know f, but we get to observe examples input-output pairs which are **generated independently at random**.

There is a truly unknown function $f: X \mapsto Y$ that we would like to estimate, and we get to observe input - output pairs. Assumption is that (x_i, y_i) which are generated independently at random. If we don't know anything about f, the only way we can talk about estimating f by looking at examples is to assume that input-output pairs are generated at random. So randomness is our assumption.

- we draw x_i according to some unknown distribution and get to observe $(x_i, f(x_i))$

- x_i from some unknown distribution and we have $(x_i, f(x_i)+n_i)$ where n_i models "noise" with an unknown distribution – interpretation: the labels are not always going to be perfect
- we draw pairs (x_i, y_i) according to some unknown joint distribution

Example: Binary Classification Problem
As a start, let's focus on one example: Binary classification where you have two classes.

Ingredients:

- output: $Y = \{0, 1\}$ or $Y = \{-1, +1\}$ are the class labels.
- seen: the training data $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ where each $x_i \in \mathbb{R}^d$ are "feature vectors".
- The learning model consist of an algorithm and $\mathcal{H} = \{h_1, ..., h_m\}$ an ensemble of possible hypothesis–potential candidates rules for the **unknown** mapping of $X \mapsto Y$.
- algorithm select the "best" possible hypothesis from $\mathcal{H}$ set.

The data are assumed to be random – are independent samples generated from some joint distribution on $\mathbb{R}^d \times \{0, 1\}$, but we don't know anything about this distribution a priori.

Tools:

- Risk $R(h_j) := \mathbb{P}[h_j(X) \neq Y]$, in other words probability of error.

  This is an utopia in learning setting, unless you know the distribution of X.....Here the risk is defined as the probability of error. This kind of definition makes sense for binary classification, it starts to make less sense if we are talking about regression problem where y's are real numbers. We will need of define more meaningful relations for regression problems...For now this is good.

- Empirical Risk $\hat{R}_n(h_j) := 1/n \sum_{i=1}^{n} 1_{h_j(x_i) \neq y_i}(i)$

  We can compute this one, it is just the estimate of the risk based on the training data–apply h_j on the training data x_i, count how many times it disagrees with the y_is we observed. This is okay because law of large numbers (LLN) says that as $n \to \infty$, $\hat{R}_n(h_j) \to R(h_j)$. In English, as the sample size grows to infinity, the empirical estimate of the risk has to converge to true risk.

Notation: Indicator function

$$1_{\{A\}}(t) = \begin{cases} 1 & \text{if } t \in A \\ 0 & \text{if } t \notin A \end{cases}$$

How to:

- Repeat: The empirical risk $\hat{R}_n(h_j)$ gives us an estimate of the true risk $\hat{R}(h_j)$, and from the LLN we know that $\hat{R}_n(h_j) \to R(h_j)$ as $n \to \infty$. Hence, this simple tool suggest the most natural way of learning in this framework.
- We have a set of hypothesis $\mathcal{H}$ and we want to choose one from that set to achieve a small risk, i.e., $h^* = \arg\min_{h_j \in \mathcal{H}} \hat{R}_n(h_j)$.

  Not a bad strategy, known also as **Empirical Risk Minimization (ERM)**

  But what if $\mathcal{H}$ is a big set, twenty trillion hypothesis, or even infinite? You may not wanna actually compute the empirical risk. We will have to do something else to search over this...

Side effects/danger:

- Is it a good idea to go after ERM?
  - If we have enough data, large n; LLN tells us empirical risk is a good estimate of true risk, $\hat{R}_n(h_j) \approx R(h_j)$.
  - However, we also have this other problem that says: if the number of hypothesis m is very large , then maybe one of these h_k's in this set will give $\hat{R}_n(h_k) \ll R(h_k)$ or $\hat{R}_n(h_k) \gg R(h_k)$.
    In other words, the empirical risk is very very different than true risk.
- What can we say about $R(h^*)$? How well we will do if we follow ERM strategy when we try h^* on unseen data? We know the empirical risk is small because we pick the one so minimizes that. What can we then say about the True performance, if we take h^* and go out and use it on new data we haven't seen yet..
  - we know $\hat{R}_n(h^*)$ is small, this could be because true risk $R(h^*)$ was small. OR...
  - it was one those examples where the empirical risk was much smaller than true risk $\hat{R}_n(h_k) \ll R(h_k)$ for some h_k.

  How do we know which one of these would have generated it? Which explanation is these two we think is more likely, vote?

  My favorite answer to this (and pretty much every question): "well it depends..." it depends on just how large is m? Second becomes more likely (just by chance) if you

have twenty gazillion hypothesis ... if you have many hypothesis, just by chance, they make look a lot better than how they actually were...This probably is an unsatisfying answer for an engineer, I am telling large-m but how big is really big.... We would like to be able to give a quantitative answer. What should m be if we have finite amount of training data n? Hint: Confidence bounds.

- If we are deciding between m hypothesis, how much data we need to ensure that

$$| \hat{R}_n(h^*) - R(h^*) | \leq \epsilon \quad (*)$$

  for some $\epsilon \in (0, 1)$. This is saying: the empirical risk for the hypothesis we picked is within ϵ of the true risk. We wanna be able to look at the training error and really conclude something about how well it is going to perform in the real world.
- If we want a result like this, we can't use asymptotic results like LLN and CLT do not give us answers to these questions.
- We want a non-asymptotic result, we wanna be able to quantify if m is finite and n is finite, what is the probability that (*) holds.

$$\mathbb{P}\left[| \hat{R}_n(h^*) - R(h^*) | \leq \epsilon \right]$$

Our goal boils down to make

$$\mathbb{P}\left[| \hat{R}_n(h^*) - R(h^*) | \leq \epsilon \right] \approx 1 \quad (**)$$

by setting n appropriately. Sanity check before we move along forward: What is random here? We are talking about the probability of empirical risk and true risk being close to each other. If I am talking about a probability, what is generating it? What is random in this statement?

- $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
- $\hat{R}_n(h_1), ..., \hat{R}_n(h_m)$ [because it depends on training data]
- h^* [because it depends on training data, in a complicated way.]

Too much randomness in (**) and with all that, it is not obvious how would you go about calculating this.

Let's just analyze a single hypothesis to make the problem easier. Here, we are not trying to do learning, we just have a single hypothesis and we wanna understand the probability that the empirical risk and the true risk close to each other in a quantitative sense. The direct analysis is a little tricky since h^* depends on the (random) data in a rather complicated way, but it turns out that we can analyze this in a fairly straightforward way by first considering

the case of a single fixed h. But keep in mind, as we talk about it in the first lecture, this is not learning, more like verification.

How close is the empirical risk to the true risk?:

$$\mathbb{P}[\ \mid \hat{R}_n(h_j) - R(h_j) \mid \leq \epsilon\ ] \approx 1$$

At this point, it is critical to realize that...Now that h_j is just fixed, true risk for it is a number, and $\hat{R}_n(h_j)$ is the only random entity in here.

To get the bound,

- we will show that $\hat{R}_n(h_j)$ is a sum of independent random variables (this is easy)
- then show that $\mathrm{E}[\hat{R}_n(h_j)] = \mathrm{R(h)}$ (also easy),
- and then develop a general-purpose probabilistic tail bound that quantifies how such a sum concentrates around its mean (this is hard).

We can write empirical risk of hypothesis h_j as a sum of Bernoulli random variables (RVs):

$$\hat{R}_n(h_j) := 1/n \sum_{i=1}^{n} 1_{h_j(x_i) \neq y_i} = 1/n \sum_{i=1}^{n} S_i$$

So, if you have a sum of Bernoulli RVs, what do you get?

S_i is Bernoulli RVs, thus, $n\hat{R}_n(h_j)$ is a Binomial RV. Since $\mathbb{P}[S_i = 1] = \mathbb{P}[h_j(x_i) \neq y_i] = R(h_j)$

$$\begin{aligned} E\left[n\hat{R}_n(h_j)\right] = E\left[\sum_{i=1}^{n} S_i\right] &= \sum_{i=1}^{n} E[S_i] \\ &= n\ \mathbb{P}[h_j(x_i) \neq y_i] \\ &= n\ R(h_j) \end{aligned}$$

This give us an equivalent way of thinking about our problem,

$$\mathbb{P}\left[\mid n\hat{R}_n(h_j) - nR_n(h_j) \mid \ \leq\ n\epsilon\right]$$

This is the probability that a Binomial RV will deviate from its mean by more than $n\epsilon$.

$$\mathbb{P}\left[\ \mid n\hat{R}_n(h_j) - nR_n(h_j) \mid \ \leq\ n\epsilon\right] = F(nR(h_j) + n\epsilon) - F(nR(h_j) - n\epsilon)$$

If we remember the CDF of a binomial Rv:

$$F(a) = \sum_{i=0}^{\lfloor a \rfloor} \binom{n}{i} R(h_j)^i (1 - R(h_j))^{(n-i)}$$

immediately you realize this is super ugly, has no closed form.

Rather than calculating this probability exactly, it is good enough to get a good bound on it, i.e. looking for an inequality of the form:

$$\mathbb{P}\left[\mid \hat{R}_n(h_j) - R(h_j) \mid \ \leq \ \epsilon\right] \geq \ 1-?$$

or equivalently,

$$\mathbb{P}\left[\mid \hat{R}_n(h_j) - R(h_j) \mid \ \geq \ \epsilon\right] \leq \ ?$$

In other words, we are looking for a bound on how a empirical risk deviates from the true risk (i.e., probability that we are getting far away is not too much). In probability theory such inequalities are known as **concentration inequalities**. There are a number of different concentration inequalities that give us various bounds along these lines. We will start with a simple one, and then build up to a stronger result. Let's remember the ones that is important for our analysis of risk.

- **Markov's Concentration Inequality:** for $X \geq 0$ any nonnegative RV, and any $t \geq 0$:

  $$\mathbb{P}\left[X \geq t\right] \leq \frac{\mathbb{E}[X]}{t}$$

  From Markov's Inequality, for any strictly monotonically increasing (non-negative-valued) function ϕ:

  $$\mathbb{P}[X \geq t] = \mathbb{P}\left[\phi(X) \geq \phi(t)\right] \leq \frac{\mathbb{E}\left[\phi(X)\right]}{\phi(t)}$$

  The first result is Chebyshev's Inequality and this surprisingly can get you pretty good results.

- **Chebyshev's Inequality:**

  $$\mathbb{P}\left[\ \mid X - \mathbb{E}[X] \mid \ \geq \ \epsilon\right] \leq \frac{var[X]}{\epsilon^2}$$

  This is more looking like the kind of thing we are after. (You can prove this using Markov's. Markov's inequality's proof is also straight forward, They are super useful, worth to review if you forgot...)

- **Hoeffding's Inequality:** This is the most useful one for our case. It assumes a bit more about our RV beyond having a finite variance, but gets us a much tighter bound.

  Let $X_i, ..., X_n$ be independent bounded RVs (more assumption, we are not just talking about non-negative random variables, bounded in interval), $\mathbb{P}[X_i \in [a, b]] = 1$ for all i.

Let $S_n = \sum_{i=1}^{n} X_i$. Then for any $\epsilon > 0$, we have;

$$\mathbb{P}\left[\mid S_n - \mathbb{E}[S_n] \mid \ \geq \ \epsilon\right] \leq 2\, e^{-\frac{2\epsilon^2}{n(b-a)^2}}$$

If you are trying to bound $\mathbb{P}[\mid S_n - \mathbb{E}[S_n] \mid \geq \epsilon]$, there is two ways you could violate it. You could have S_n is too big or S_n is too small. To begin consider only the upper tail inequality:

$$\begin{aligned}
\mathbb{P}[S_n - \mathbb{E}[S_n] \geq \ \epsilon] &= \mathbb{P}[\lambda\ S_n - \mathbb{E}[S_n] \geq \ \lambda\epsilon] \quad (\lambda > 0) \\
&= \mathbb{P}[e^{\lambda\ (S_n - \mathbb{E}[S_n])} \geq \ e^{\lambda\epsilon}] \quad \text{, apply Markov Ineq. to this} \\
&\leq \frac{e^{\lambda\ (S_n - \mathbb{E}[S_n])}}{e^{\lambda\epsilon}} \\
&= e^{-\lambda\epsilon}\ \ \mathbb{E}\ [\ e^{\lambda(X_1 - \mathbb{E}[X_1] + \ldots + X_n - \mathbb{E}[X_n])}\ ] \\
&= e^{-\lambda\epsilon}\ \ \prod_{i=1}^{n}\ \mathbb{E}\ \left[\ e^{\lambda(X_i - \mathbb{E}[X_i])}\right] \quad \text{independence}
\end{aligned}$$

Using Hoeffding's Lemma, it is not obvious but also not too hard to show that (to prove use convexity and then get a bound using Taylor series expansion),

$$\mathbb{E}\ \left[\ e^{\lambda(X_i - \mathbb{E}[X_i])}\right] \leq e^{\lambda^2 (b-a)^2/8}$$

Plugging this in, we obtain that for any positive λ,

$$\mathbb{P}[S_n - \mathbb{E}[S_n] \geq \epsilon] \ \leq \ e^{-\lambda\epsilon} e^{\lambda^2 (b-a)^2/8}$$

By setting $\lambda = \frac{4\epsilon}{n(b-a)^2}$,

$$\begin{aligned}
\mathbb{P}[S_n - \mathbb{E}[S_n] \geq \epsilon] &\leq \ e^{-\frac{4\epsilon^2}{n(b-a)^2}} e^{\frac{2\epsilon^2}{n(b-a)^2}} \\
&= e^{-\frac{2\epsilon^2}{n(b-a)^2}}
\end{aligned}$$

Okay, this is for S_n too big, bounded! You can use the same argument and do the other version (lower tail probability):

$$\mathbb{P}[\mathbb{E}[S_n] - S_n \geq \epsilon] = e^{-\frac{2\epsilon^2}{n(b-a)^2}}$$

Finally, combined:

$$\mathbb{P}[\mid S_n - \mathbb{E}[S_n] \mid \geq \epsilon] \leq 2e^{-\frac{2\epsilon^2}{n(b-a)^2}}$$

Special case: X_i are Bernoulli RVs, then S_n is a Binomial RV, and Hoeffding's Inequality becomes:

$$\mathbb{P}[\mid S_n - \mathbb{E}[S_n] \mid \geq \epsilon] \ \leq \ 2e^{-\frac{2\epsilon^2}{n}}$$

Going back to our original problem, we are interested in:

$$\mathbb{P}[|\ \hat{R}_n(h_j) - R(h_j)\ |\ \ \geq\ \epsilon]$$

This is not exactly Binomial, we need to multiply with n, and use Hoeffding's Lemma:

$$\begin{aligned}
&\mathbb{P}[|\ \hat{R}_n(h_j) - R(h_j)\ |\ \ \geq\ \epsilon] \\
&\qquad = \mathbb{P}[|\ n\ \hat{R}_n(h_j) - n\ R(h_j)\ |\ \ \geq\ n\epsilon] \\
&\qquad \leq 2\ e^{-2\epsilon^2 n}
\end{aligned}$$

As n gets really big, bounds gets tighter exponentially fast! Strong statement. Thus after much effort, we have that for "a particular" hypothesis h_j,

$$\mathbb{P}[|\ \hat{R}_n(h_j) - R(h_j)\ |\ \ \geq\ \epsilon]\ \leq\ 2\ e^{-2\epsilon^2 n}$$

However, we are ultimately interested in h^*, not just a single hypothesis h_j. Ah :)

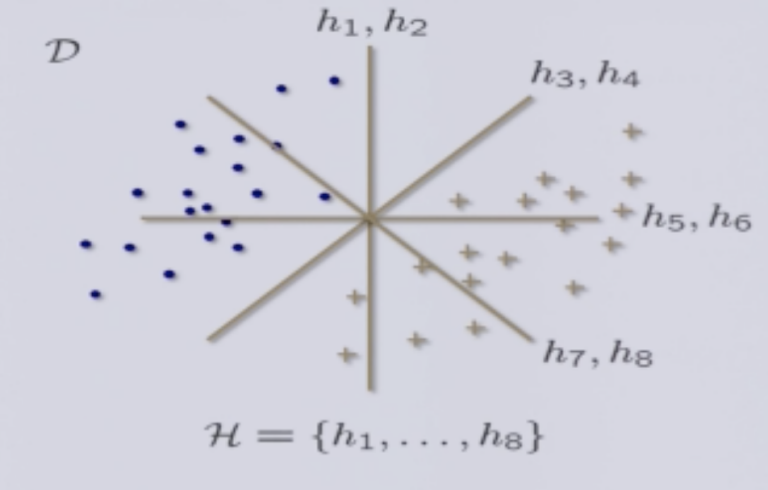
This gives us insight into how the performance of one single classification rule on the training set generalizes. What we want is some assurance that the one we judge to be the best, by performing ERM on the data, will be close to the best choice we could have made. We will get this assurance by developing a similar probability bound that holds uniformly over all classifiers in H.

One way to argue that $|\ \hat{R}_n(h^*) - R(h^*)\ |\ \ \leq\ \epsilon$ is to ensure that $|\ \hat{R}_n(h_j) - R(h_j)\ |\ \ \leq\ \epsilon$ **simultaneously** for all possible j. We can express this mathematically as

$$\begin{aligned}
\mathbb{P}[\ \ |\ \hat{R}_n(h^*) - R(h^*)\ | \geq \epsilon] \leq \mathbb{P}[\ &|\ \hat{R}_n(h_1) - R(h_1)\ | \geq \epsilon \\
&\text{OR}\ \ |\ \hat{R}_n(h_2) - R(h_2)\ | \geq \epsilon \\
&\qquad \vdots \\
&\text{OR}\ \ |\ \hat{R}_n(h_m) - R(h_m)\ | \geq \epsilon\ ]
\end{aligned}$$

By that, we can show:

$$\mathbb{P}[\ \ |\ \hat{R}_n(h^*) - R(h^*)\ | \geq \epsilon] \leq\ 2m\ e^{-2\epsilon^2 n}$$



Back to the definitions of risk/empirical risk

$$\text{true risk}: \; R(h_j) \;:=\; \mathbb{P}[h_j(X) \neq Y]$$

$$\text{empirical risk}: \; \hat{R}_n(h_j) \;:=\; \tfrac{1}{n}\textstyle\sum_{i=1}^{n} 1_{h_j(x_i)\neq y_i}(i)$$

Summary so far: a natural strategy is ERM: $h^* = \arg\min_{h_j \in \mathcal{H}} \hat{R}_n(h_j)$

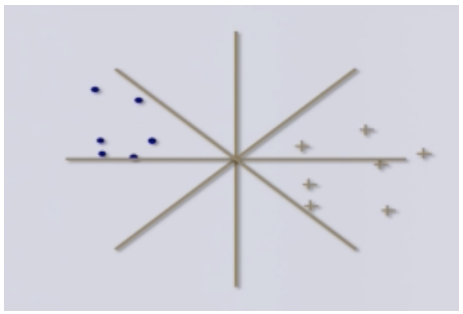
We are going use training data to select best classier [refer to picture]. We know that as long as we get enough data, for particular hypothesis $\hat{R}_n(h_j) \approx R(h_j)$. The problem with that strategy if, if m is large then we can also expect there are some h_k for which $\hat{R}_n(h_k) \ll R(h_k)$ or $\hat{R}_n(h_k) \gg R(h_k)$. In English, some hypothesis might look a lot better than they really are, just by chance!

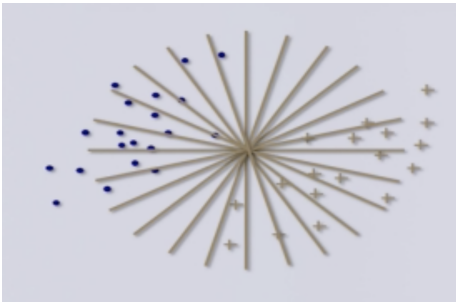
Looks like ERM will be extremely vulnerable - we should worry about this :)

Thus, what can we say about $R(h^*)$?

- We know that $\hat{R}_n(h^*)$ is as small as it can be
  - this **could** be because $R(h^*)$ is small
  - or, it could be because $\hat{R}_n(h_k) \ll R(h_k)$ for some h_k
- Which explanation is more likely?
  - **it depends...** just how large is m?

To get a quantitative resolution to the question how big n needs to be, we relied on concentration inequalities. For a particular hypothesis we appeal to Hoeffding's:





$$\mathbb{P}[\ \mid \hat{R}_n(h_j) - R(h_j) \mid \geq \epsilon] \leq e^{-2\epsilon^2 n}$$

However, we are ultimately interested in h^*, not just a single h_j. And union bound got us to this point:

$$\mathbb{P}[\ \mid \hat{R}_n(h^*) - R(h^*) \mid \geq \epsilon] \leq\ 2m\ e^{-2\epsilon^2 n}$$

- linearly increasing with m
- exponentially decreasing with n
- for fixed n how big m can actually be?

Sorry, same question again: When can we be confident that $\hat{R}_n(h^*) \approx R(h^*)$?

Note that $2m\ e^{-2\epsilon^2 n} = e^{\ log(2m)\ -\ 2\epsilon^2 n}$

Maybe now it is more obvious that we don't want $log(2m)$ to get much bigger than $2\epsilon^2 n$ As long as m isn't too big ($m < e^n$) then we can be confident that $\hat{R}_n(h^*) \approx R(h^*)$

All this is good, we came to a point that we can say something about for a particular case where we are considering finite number of classifiers, how much data do we need to be sure that our training error is a representative. In other words, how this classifier will perform in practice. So that's good. But not quite enough to have $\hat{R}_n(h^*) \approx R(h^*)$ (learning)

Ideally, we would like to have $R(h^*) \approx 0$, we are very demanding!
Note that if $\hat{R}_n(h^*) \approx R(h^*)$, then $\hat{R}_n(h^*) \approx 0$ implies $R(h^*) \approx 0$.

Learning Problem:

1. Can we ensure that $R(h^*)$ is close to $\hat{R}_n(h^*)$? ✓
2. Can we make $\hat{R}_n(h^*)$ small enough?

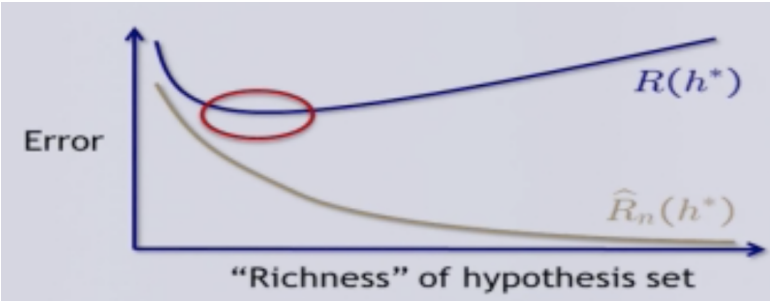
We want to make $\hat{R}_n(h^*)$ as small as possible and our initial intuition was to pick

$$h^* = \arg\min_{h_j \in \mathcal{H}} \hat{R}_n(h_j)$$

This will make $\hat{R}_n(h^*)$ small only if there is some $h_j \in \mathcal{H}$ such that actually does well on our training data set. We need a rich set of possible hypotheses...

And we already understood the fundamental **tradeoff** here: more hypothesis ultimately sacrifices our guarantee that $\hat{R}_n(h^*) \approx R(h^*)$.

- if we have rich set of hypothesis training error, $\hat{R}_n(h^*)$ goes down, no news.
- unfortunately at the same time $\hat{R}_n(h^*) - R(h^*)$ goes up.



Since all you can see is the error in the training data, it is very tempting to operate in the terrible regions of this graph, and really over-fit.

We would like to have a small number of hypothesis so that $\hat{R}_n(h^*) \approx R(h^*)$ while hoping to have $\hat{R}_n(h^*) \approx 0$. Together these imply $R(h^*) \approx 0$. In general this may not be possible, since there may not be any function f with $R(f) \approx 0$. Why not?

$$\text{Noise} : Y = f(X) + N$$

Suppose we **knew** the joint distribution of our data, then you can ask "what is the optimal classification rule f^*"?

Consider (X, Y) pair where

- X is a random vector in $\mathbb{R}^d$
- $Y \in \{0, 1, ..., K-1\}$ is a random variable that depends on X.

Let $f : \mathbb{R}^d \to \{0, 1, ..., K-1\}$ be a some classifier with probability of error/risk given by

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

Let's denote **a posteriori** class probabilities by

$$\eta_k(x) \ := \ \mathbb{P}[Y = k | X = x]$$

for $k = 0, ..., K-1$. If you know this probability, you can immediately say what is the optimal classifier is.

Theorem: The classifier $f^*(x) \ := \ \arg\max_k \ \eta_k(x)$ satisfies

$$R(f^*) = \min \ R(f)$$

where the minimum is over all possible classifiers (every possible k). We are going to use the one with the highest **a posteriori** probability, we can show that it is the best we can ever get.

Terminology:

- f^* is called the **Bayes classifier**
- $R(f^*)$ is called the **Bayes risk**

For convenience, we will assume $X|Y = k$ is a continuous RV with density $g_k(x)$ (if X is discrete we can carry out everything we are going to do here, we just have to deal with PMF in that case, it is a little bit uglier, but everything generalizes to the discrete case). The idea here, if you tell me what class my data came from then I know the PDF of the data. We also need to keep track of **a priori** class probabilities, let's denote that by $\pi_k = P[Y = k]$–how likely is each class. Before we see any of the data, these are the probabilities we have. Once we see the data, it is going to revise our opinion.

Consider an arbitrary classifier f, and denote the decision regions:

$$\Gamma_k(f) \ := \ \{x \ : \ f(x) = k\}$$

f is a general classifier we can always do this (maybe an illustration here).

$$\begin{aligned} 1 - R(f) = \ & \mathbb{P}[f(X) = Y] \\ & = \sum_{k=0}^{K-1} \pi_k \times \mathbb{P}[f(X) \ = \ k \mid Y \ = \ k] \\ & = \sum_{k=0}^{K-1} \pi_k \int_{\Gamma_k(f)} g_k(x) dx \end{aligned}$$

If we want to maximize this expression, we should design our classifier f such that,

$$x \in \Gamma_k(f) \iff \pi_k g_k(x) \text{ is maximized}$$

Therefore, the optimal f has

$$f^*(x) = \arg\max_k \ \pi_k g_k(x)$$

Just for fun, I am going to write this expression slightly different

$$f^*(x) = \arg\max_k \ \frac{\pi_k g_k(x)}{\sum_{l=0}^{K-1} \pi_l g_l(x)}$$

By writing like this I can express this what I have start out (a posteriori class probabilities)– using Bayes rule

$$f^*(x) = \arg\max_k \ \mathbb{P}[Y = k \mid X = x]$$

This classifier makes sense! At the end of this rigorous derivation, it would be kind of disturbing if $\arg\max_k \ \mathbb{P}[Y = k \mid X = x]$ wasn't the right thing to do. Once I get to see my data, I should classify it based on which Y has the highest probability of being true. It will be weird if anything else turned out to be the answer for minimizing the probability of error.

There are lots of ways of expressing the Bayes classifier. Let's look at the couple variations because most likely you have seen this before.

- $f^*(x) = \arg\max_k \ \eta_k(x)$
- $f^*(x) = \arg\max_k \ \pi_k g_k(x)$
- When $K = 2$, we can do **likelihood ratio test**

$$\frac{g_1(x)}{g_0(x)} \underset{1}{\overset{0}{\gtrless}} \frac{\pi_0}{\pi_1}$$

- When $\pi_0 \ = \pi_1 = ... = \pi_{K-1}$

$$f^*(x) = \arg\max_k \ g_k(x)$$

This has a special name "ML-classifier", maximum likelihood classifier/detector.

Let's look at a $2-d$ example: suppose that $K = 2$ and that

$$X|Y = 0 \sim \mathcal{N}(0, 1) \text{ and } X|Y = 1 \sim \mathcal{N}(1, 1)$$

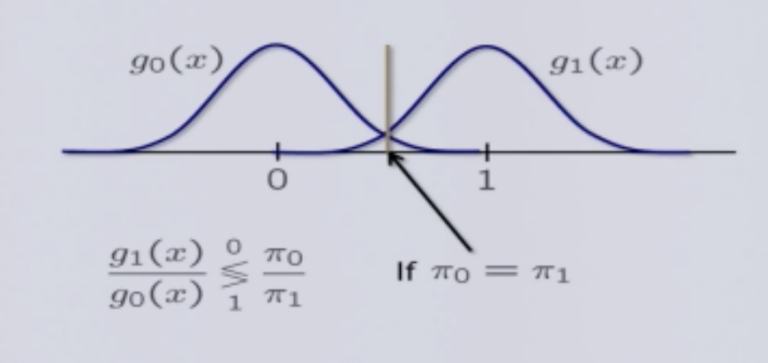
Here we are randomly choosing a class according to some distribution. And given which class we belong to, there is some class conditional distribution for X. What is the likelihood ration test say:

$$\frac{g_1(x)}{g_0(x)} \underset{1}{\overset{0}{\lessgtr}} \frac{\pi_0}{\pi_1}$$

If $\pi_0 = \pi_1$, threshold will be 1. And how do we calculate the Bayes risk in this case?

$$\begin{aligned} R(f^*) &= \mathbb{P}[f^*(X) \neq Y] \\ &= \mathbb{P}[declare\ 0 | Y = 1]\pi_1 \\ &\quad + \mathbb{P}[declare\ 1 | Y = 0]\pi_0 \end{aligned}$$

In the case where $\pi_0 = \pi_1 = 1/2$, our test reduced to declaring 1 iff $x \geq \frac{1}{2}$ (Let's plot how this looks like).



Thus,

$$\begin{aligned} R(f^*) &= \mathbb{P}[X < \frac{1}{2} | Y = 1] + \mathbb{P}[X > \frac{1}{2} | Y = 0] \\ &= \frac{1}{2} \int_{-\infty}^{\frac{1}{2}} g_1(t)dt + \frac{1}{2} \int_{\frac{1}{2}}^{\infty} g_0(t)dt \\ &= \Phi\left(-\frac{1}{2}\right) \end{aligned}$$

End note: Note that we don't know any of the g_ks, π_ks, and it is going to super hard in a lot of cases. Everything we did here, assumed we know the full distribution–what generated our data. If we know π_ks and all g_ks, then we can actually do the optimal thing. Not today, we will talk about this next lecture, we will look at the methods that are inspired by this approach.We might assume distributions as a start (Gaussian is always a good one), and estimate π_ks...