SLT - Summary

> By Joost Visser

LaTeX commands

Chapter 1 - Some definitions

Data spaces

Goal of statistical learning theory: *supervised learning*.

- Construct a classifier / function that takes an input and outputs the corresponding output.

Data spaces: Set that is given as input, and the set on which the output should be an element from.

- $\mathcal{X}$ represents the input space, $\mathcal{Y}$ represents the output space.

Training data: $\{(X_i, Y_i)\}_{i=1}^n$

Loss function

Loss function measures how different the predicted label $\hat{Y} \in \mathcal{Y}$ compares with the actual label $Y \in \mathcal{Y}$. This is a function that takes $\hat{Y}$ and Y, and maps it to a real value (i.e. the loss).

- $\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$ for binary loss.
  When we predict the wrong label, the loss is 1, else it is 0.

Note that these loss functions don't have to be symmetric, as is shown in the spam example where false positives should be punished much more than false negatives.

In *regression*, the *squarred error* loss function is often used: $\ell(\hat{y}, y) = (\hat{y} - y)^2$.

Probability measure and Expectation

Define a join probability destribution on $\mathcal{X} \times \mathcal{Y}$ and $\mathbb{P}_{XY}$, where (X, Y) is a pair of random varaibles distributed according to $\mathbb{P}_{XY}$. This can also be described as the marginal distribution of X ($\mathbb{P}_X$) and the conditional distribution of Y given X ($\mathbb{P}_{Y|X}$).

Let p denote the density function of $\mathbb{P}$. Then we have the expectation operator:

$$\mathbb{E}[f(X, Y)] \equiv \int f(x, y) d\mathbb{P}_{XY}(x, y) = \int f(x, y) p_{XY} dx dy$$

Statistical risk

The **risk** is how well the classifier does on average, also known as the expected loss.
Given prediction rule f and $\mathbb{P}_{XY}$, then the risk is defined as:

$$R(f) \equiv \mathbb{E}[\ell(f(X), Y | f]$$

The *minimum risk* is the best possible risk, which is where we're interested in.

$$R^* = \inf_{f \text{ measurable}} R(f)$$

Learning

To compute the risk, we need to know the distribution $\mathbb{P}_{XY}$. However, this generally is unknown, so we try to learn a good prediction rule f using the training data!

Assumption: We assume that the training data $D_n \equiv \{X_i, Y_i\}_{i=1}^n$ is i.i.d. according to $\mathbb{P}_{XY}$.

- *Independence:* Samples comes from different sources and do not influence each other.
- *Identical:* The samples we have are a representative sample of the type we might encounter in the future.

Learning goal: Use training data D_n to choose a function / mapping $\hat{f}_n$ out of a class of candidate prediction rules $\mathcal{F}$.

- Main goal of this course is to guarantee that $R(\hat{f}_n)$ is small with a very high probability.
- We could also consider the *expected risk*: $\mathbb{E}[R(\hat{f}_n]$.

Law of total expectation: $\mathbb{E}[\mathbb{E}[Y|X]] = \mathbb{E}[Y]$.

- Note that $\mathbb{E}[Y|X]$ is conditioned on X, which is random, thus $\mathbb{E}[Y|X]$ is also a random variable!

Chapter 2 - Binary classification and Regression

Binary classifiation

In this chapter, we will consider the **binary classification** setting and the **regression** setting. We will use the familiar 0/1 loss function:

$$\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$$

In this setting, the risk of classifier f can be rewritten to a probability:

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

Thus, the risk is simply the probability of making errors.

The performance of the best classification rule known is the **Bayes risk**:

$$R^* = \inf_{f \text{ measurable}} R(f)$$

The best classifier is known as the Bayes classifier:

$$f^*(x) = \mathbf{1}\{\eta(x) \geq \frac{1}{2}\}$$

- Where $\eta(x) = \mathbb{P}_{Y|X}(Y = 1 | X = x)$ is the feature conditional probability.

This basically tells that if you know for given $X = x$ that the probability of $Y = 1$ is greater than the probability of $Y = 0$, then we should classify 1.
This has the best risk we could hope for:

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

In practice, however, we don't know the distribution $\mathbb{P}_{XY}$, so we cannot construct $\eta(x)$.

Excess risk: risk we get over the best possible risk. Can be calculated as follows:

$$R(g) - R^* = \int_{G \Delta G^*} |2\eta(x) - 1| d\mathbb{P}_X(x)$$

- Where $G \Delta G^* = (G \cap \bar{G^*}) \cup (\bar{G} \cap G^*)$ is the symmetric set difference, the xor.

Example: *Linear classifiers*. These are classifiers where a line (/ hyperplane) is drawn on the feature space $\mathcal{X}$ and one part is classified as 1, while the other part is classified as 0.

$$\mathcal{F} = \left\{ x \mapsto f(x) = \text{sign}(w^T x) \right\}$$

Fun facts:

1. If there exists a hyperplane that separates the classes pefectly, then it can be found efficiently using e.g. the perceptron algorithm.
2. If there does not exist such hyperplane, finding the hyperplane that minimizes the risk is computationally very demanding.

Regression

In regression, the prediction rule is often called an *estimator*. For this setting, consider the squared loss function:

$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$

In this setting, we can rewrite the risk as follows:

$$R(f) = \mathbb{E}\left[(f(X) - Y^2) \middle| f \right]$$

The regression function has the lowest possible risk, which can be written as:

$$f^*(x) = \mathbb{E}[Y | X = x]$$

And this risk equals the Bayes risk:

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

Example: *(Linear) regression*. Let the possible estimators be:

$$\mathcal{F} = \{ \text{degree } d \text{ polynomials on } [-1, 1] \}$$

Then, the empirical risk minimizer (see below) is the classifier which minimizes the MSE. The best classifier can be found using some fancy linear algebra ((Chebyshev) Vandermonde matrix).

Empirical Risk Minimization

How do we actually use the data to choose a good prediction rule f? One way of doing so is to calculate the risk on the training data, $\hat{R}_n$, and pick the f that reduces this risk as much as possible.

Empirical risk: The risk of the training data.

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i)$$

So, the idea of picking the f that minimizes this empirical risk the most is called *empirical risk minimization*.

Empirical risk minimizer: Choose the rule that minimizes the empirical risk.

$$\hat{f}_n = \underset{f \in \mathcal{F}}{\arg\min}\, \hat{R}_n(f)$$

By the law of large numbers, we know that $\hat{R}_n(f)$ will eventually converge to $R(f)$ if n is large enough.

Model Complexity and Overfitting

Suppose our $\mathcal{F}$ is really large. Then we can make our empirical risk really small, as we have many different functions to choose from. In the most extreme case, we can even consider f_{bad} that predicts Y_i for all $x = X_i$, and 0 otherwise. However:

- This rule predicts zero for any value of the feature not in the dataset.
  - $\hat{R}_n(f_{bad})$ will be 0, but $R(f_{bad})$ will be very large.
- This rule fits the data "too well" (overfits) and doesn't learn anything outside teh dataset.

Thus, if we make $\mathcal{F}$ too large, we might fit the training data too well. So we do not want a class of models that is too large when considering empirical risk minimization.

Chapter 3 - Competing Goals: approximation vs estimation

As shown in the previous chapter, the size of $\mathcal{F}$ seems to play a crucial role on the performance of the estimator. One way of avoiding overfitting is to restrict $\mathcal{F}$ to some measure set. To analyze the performance of a $\hat{f}_n$ with this in mind, consider the following representation of the *expected excess risk*:

$$\mathbb{E}[R(\hat{f}_n)] - R^* = \underbrace{\left(\mathbb{E}[R(\hat{f}_n)] - \inf_{f \in \mathcal{F}} R(f)\right)}_{\texttt{estimation error}} + \underbrace{\left(\inf_{f \in \mathcal{F}} R(f) - R^*\right)}_{\texttt{approximation error}}$$

Note that $\inf_{f \in \mathcal{F}} R(f)$ is the best estimator $f \in \mathcal{F}$.

- **Approximation error:** The performance hit by imposing restrictions on $\mathcal{F}$.
  - Data does not play a role her, only $\mathcal{F}$ and $\mathbb{P}_{XY}$ do.
- **Estimation error:** How well we can use the data to identify the best element in $\mathcal{F}$.
  - Data plays a role here, as well as $\mathcal{F}$ and $\mathbb{P}_{XY}$.

Chapter 4 - Estimation of Lipschitz smooth functions

Chapter 5 - Introduction to PAC learning

Chapter 6 - Concentration Bounds

Chapter 7 - General bounds for bounded losses

Chapter 8 - Countably Infinite Model Spaces

Complexity Bounds

Theorem 6 - Complexity Regularized Model Selection

Let $\mathcal{F}$ be a *countable* collection of models and assign a real number $c(f)$ to each $f \in \mathcal{F}$ such that:

$$\sum_{f \in \mathcal{F}} e^{-c(f)} \leq 1$$

Let us define the minimum complexity regularized model (i.e. the ERM with regularization), this is also called the *minimum penalized empirical risk predictor*:

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} \left\{ \hat{R}_n(f) + \sqrt{\frac{c(f) + \frac{1}{2}\log n}{2n}} \right\}$$

Then we can bound our expected risk compared to the actual risk:

$$\mathbb{E}[R(\hat{f}_n)] \leq \inf_{f \in \mathcal{F}} \left\{ R(f) + \sqrt{\frac{c(f) + \frac{1}{2}\log n}{2n}} + \frac{1}{\sqrt{n}} \right\}$$

This theorem is quite useful, as it lets us bound the expected risk and thererfore gives us an idea of what a good estimator is for $\hat{f}_n$.

This estimator also tells us that the performance (risk) of $\hat{f}_n$ performs almost as well as the best possible prediction rule $\tilde{f}_n$.

Histogram example

The expected value of our empirical risk when choosing the number of bins automatically performs almost also as good as the best k (number of bins) possible! Since:

$$\mathbb{E}[R(\hat{f}_n)] \leq \inf_{k \in \mathbb{N}} \left\{ \min_{f \in \mathcal{F}_k} R(f) + \sqrt{\frac{(k + k^d)\log 2 + \frac{1}{2}\log n}{2n}} + \frac{1}{\sqrt{n}} \right\}$$

- $\hat{f}_n = \hat{f}_n^{(\hat{k}_n)}$ - Our best k we could automatically determine depending on the data using the empricial risk minimizer where the $\hat{k}_n$ is found by minimizing:
  - $\hat{k}_n = \arg\min_{k \in \mathbb{N}}$ ERM of each subclass + penalized term.

Chapter 9 - The Histogram Classifier revisited

Complexity regularization

Here, instead of a coding argument, we use an explicit map for $c(f)$, namely:

$$c(f) = \log(m_f) + \log(m_f + 1) + m_f \log(2)$$

- Where m_f is the smallest value of k for which $f \in \mathcal{F}_m$, aka a partitioning of m bins.

Now, via a similar method as the last chapter, we can grab the number of bins $\hat{m}_n$ via minimizing the (ERM + Reguralization term). Using this $\hat{m}_n$, we get our estimator $f_n = f_{n,\hat{m}_n}$.

The theory shows that the estimation error on the $\hat{f}_n$ is as almost the same (worst-case) as $f_{n,m_{f^*}}$, which is the best (im)possible choice of the number of bins.

Even if we do this for d input dimensions instead of 2, we can choose the number of bins in the histogram automatically in such a way that we do almost as well as if we actually knew the right number!

Note: The penalization is much more severe than deemed necessary, so we're a bit over-conservative. We can either try to create better bounds or use CV to solve this.

Leave-one-out Cross V alidation

The idea for leave-one-out cross valudation is to consider n splits of the data into training set ($n-1$) and validation set 1. After some mathematics, we can show that the cross-validation risk $\mathbb{E}[CV_{n,m}] = R(\hat{f}_{n-1,m})$. If n is not too small, then $R(\hat{f}_{n,m}) \approx R(\hat{f}_{n-1,m})$, therefore we get an unbiased estimator of the actual risk R of the classifier, not the estimated risk $\hat{R}$!

We can even choose the number of bins as follows:

$$\hat{f}_n^{(CV)} = \hat{f}_{n,\hat{m}_{CV}}$$

- $\hat{m}_{CV} = \arg\min_m CV_{n,m}$, the best possible number of bins found via CV.
- $CV_{n,m}$ is the CV risk, the formula is shown in chapter 9.

This often works very well in practice, but is not a silver bullet and an answer to all problems.

Chapter 10 - Decision Trees and Classification

Excess risk of the penalized empirical risk minimization

Suppose we grab the Complexity Regularization Bounds and rewrite it in terms of the excess risk $\mathbb{E}[R(\hat{f}_n)] - R^*$, where R^* is the Bayes risk. Then we get the following formula:

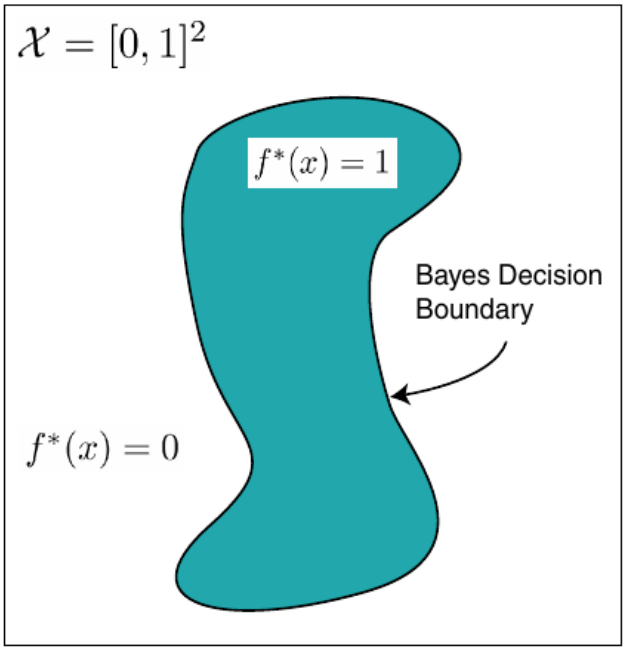
$$\mathbb{E}[R(\hat{f}_n)] - R^* \leq \inf_{f \in \mathcal{F}} \left\{ \underbrace{R(f) - R^*}_{\texttt{approximation error}} + \underbrace{\sqrt{\frac{c(f) + \frac{1}{2}\log n}{2n}} + \frac{1}{\sqrt{n}}}_{\texttt{bound on estimation error}} \right\}$$

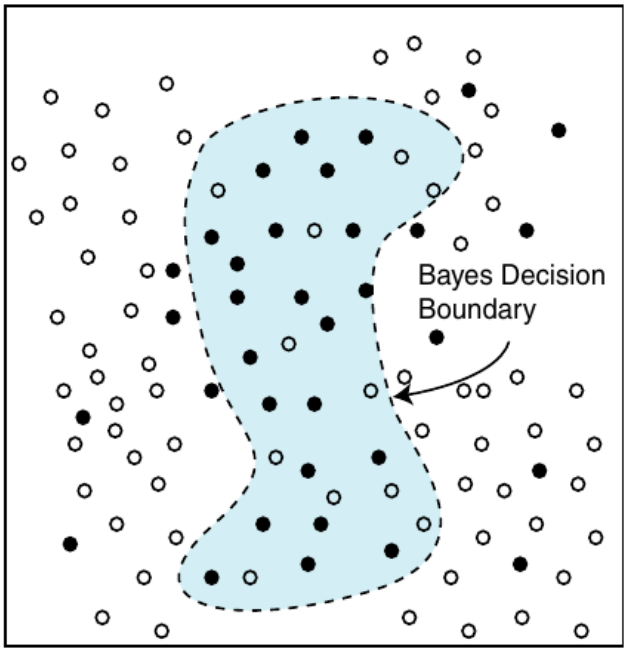
Thus, we see that complexity regularization automatically optimizes a balance between approximation and estimation error. Therefore it's *adaptive* to this unknown tradeoff.

This result is useful only if the bound in the estimation error is good and the class $\mathcal{F}$ has enough models for the approximation error. To check this, we need to make assumptions of $\mathbb{P}_{XY}$.

Binary classification

Suppose we have a universe $\mathcal{X} = [0,1]^2$, which you can idenity as a square. You can consider the best classifier (Bayes' classifier) as a set $G^* = \{x : \mathbb{P}(Y = 1 | X = x) \geq \frac{1}{2}\}$, such that our Bayes' classifier $f^*(x) = \mathbf{1}\{x \in G^*\}$. In words, this means that we have a subset of $\mathcal{X}$, namely G^*, and if our features x are inside this set/field, then we should predict 1. An illustration of this can be shown below:





The boundary of G^* is called the *Bayes Decision Boundary* and is given by $\{x : \eta(x) = \frac{1}{2}\}$.

The problem with the histogram classifier is that it tries to estimate $\eta(x)$ everywhere and thesholding the estimate at level $\frac{1}{2}$ (i.e. it looks for the Bayes Decision Boundary everywhere).

Binary classification trees

Growing

The idea of binary classification trees is similar to the histogram classifier; partition the feature space into separate sets and use a majority vote to choose the predicted label for each set. Where the difference lies is how we partition the set. The histogram partitions it *a priori* with constant sets, whereas the trees learn the partitioning from the data.

The idea of the trees is to:

1. *Grow* a very large, complicated tree. (So subdivide $\mathcal{X}$ a lot.)
2. *Prune* the tree. (Combining partitions to reduce overfitting._

The goal is to have a lot of paritions near the boundary to approximate them well, but to have large partitions far from the boundary to avoid overfitting. For this setting, we will consider *Recursive Dyadic Paritions (RDP)* as these are easy to analyse. These RDPs split the feature space $\mathcal{X}$ in half vertically, then horizontally in one of the two sub-paritions, then vertically and so forth.

So... where are the trees? Well, we can associate any partition of $\mathcal{X}$ with a decision tree. This is even the *most efficient way* to describe an RDP.

- Each *leaf* corresponds to a cell of the partition.
- The *nodes* correspond to the various partition cells that are generated in the *construction* of the tree.

Pruning to get our estimator

Let $\mathcal{F}$ be all possible RDPs of $\mathcal{X}$. To make use of our theory and bounds of Chapter 8, we need to find our $c(f)$. We do so by constructing a prefix code:

1. All internal nodes will be 0, all leaves will be 1.
2. Each leaf will have a decision label (either zero or one).
3. Read the code from top-down, left-right.

This results in total of $2k-1$ bits for a tree of k leaves, and k bits for each decision, so a total of $3k-1$ bits. Now, we want to solve the following formula to get our bounded estimator:

$$\hat{f}_n = \arg\min_{f\in\mathcal{F}} \left\{ \hat{R}_n(f) + \sqrt{\frac{(3k(f)-1)\log 2 + \frac{1}{2}\log n}{2n}} \right\}$$

Again, here we have a tradeoff between the *approximation error* and the *estimation error*, which we try to minimize. Using a bottom-up pruning process, we can solve this very efficiently as we balance the reduction of $k(f)$ with an increase with risk.

Comparing the histogram classifier and classification trees

So, now we have a bound on the *excess risk* of the histogram classifier and the Dyadic Decision trees. How do they compare?

Box-Counting assumption

To compute the excess risk, we first need to compure the approximation error $(R(f)-R^*)$. For this, we need to make an assumption about the distribution $\mathbb{P}_X$, namely the **Box-Counting assumption**.

The *Box-Counting assumption* is essentially stating that the overall length of the Bayes' decision boundary is finite (i.e. no fractals). More specifically, a length of at most $\sqrt{2}C$.

Furthermore, we assume that the marginal distribution of X satisfies $\mathbb{P}_X(A) \leq p_{max}\,\mathrm{vol}(A)$, so we can always bound the volume as well for each subset of A.

Histogram Risk Bound

After some fancy mathematics to calculate the expected excess risk of the histogram classifier, we can get a bound of:

$$\mathbb{E}[R(\hat{f}_n^H)] - R^* = \mathcal{O}(n^{-1/4})$$

So, as our sample size n grows, the expected excess risk will decrease w.r.t. $n^{-1/4}$.

Dyadic Decision Tree

Lemma 10.4.1 in the lecture notes tells us that there exists a DDT with at most $3Ck$ leafs that has the same risk as the best histogram with $\mathcal{O}(k^2)$ bins. Therefore, using a similar calculation but with a different mapping $c(k)$, we can get a better bound on the expected excess risk:

$$\mathbb{E}[R(\hat{f}_n^H)] - R^* = \mathcal{O}(n^{-1/3})$$

This is because our bound on the estimation error is smaller as we only have a factor k instead of $k + k^2$.

Final remarks

Histograms vs Trees

Trees generally work much better than histogram classifiers, because they approximate the Bayes' decision boundary in a much more efficient way:

- They only need a tree of $\mathcal{O}(k)$ bits instead of a histogram that requires $\mathcal{O}(k^2)$ bits.
- Because of this, the expected excess risk will converge faster and will require less memory.

In fact, the DDTs are very deep histogram classifiers followed by pruning to balance the approximation error and the estimation error.

Other remarks

1. There exists an even slightly tighter bounding procedure for the estimation error, shown by Scott and Nowak, by using the depth of the leafs as penalization variable.
2. These bounds will work, but are quite conservative as we are over-estimating the estimation error (as we have an upper bound on the estimation error). We can add an estimator variable for this estimation error and use CV to get a great value for it.

Chapter 11 - VC Bounds

Uncountable infinite classes

What happens if our class of candidate models $\mathcal{F}$ is infinite and uncountable? Then we cannot use the previous theory, as they all assumed that $\mathcal{F}$ is either finite or countable. An example of an uncountable class $\mathcal{F}$ would be:

$$\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y} : f(x) = \mathbf{1}\{x \geq t\}, t \in [0, 1]\}$$

In words, this is the class of models where everything $\geq t$ will be classified as 1, whereas everything $< t$ will be classified 0. We will use this example throughout the chapter.

There are two ways to fix this:

1. Discretize or Quantize the collection
2. Identical Empirical Errors

Discretize the collection

This approach is similar to the one in exercises of Chapter 7, where we discretize the class in Q parts. $\mathcal{T} = \left\{0, \frac{1}{Q}, \frac{2}{Q}, \cdots, \frac{Q}{Q}\right\}$ are the parts, and $\mathcal{F}_Q$ is the class for $t \in \mathcal{T}_Q$. Choosing Q large enough results in $\mathcal{F}_Q$ and $\mathcal{F}$ being nearly identical.

This trick works in many situations, but is quite tedious and messy and makes analyzing a tad more cumbersome.

Identical Empirical Errors

If you consider the class of models $\mathcal{F}$ above, there are many values for t that give rise to exactly the same error. If they give the same error, why should we differentiate between them? In terms of error, they act exactly the same. In particular, if we are using a dataset of size n, there will only be $\mathcal{O}(n)$ different error classes.

Therefore, in this setting, we will only consider the different empirical error classes, which allows us to get generalization bounds.

Vapnik-Cheronenkis (VC) Theory

> In the rest of the chapter, we will only consider binary classification with 0/1 loss.

Shatter-Coefficient

For a given training set D_n of size n, there are at most 2^n different labelling (and thus different error classes). However, in practice, there are often much fewer possibilities. Let $\mathcal{S}(\mathcal{F}, n)$ be the maximum number of labelling sequences of class $\mathcal{F}$ over n training points in $\mathcal{X}$. We call this the *Shatter-Coefficient*. More formally:

Shatter-Coefficient: $\mathcal{S}(\mathcal{F}, n) = \max_{x_1,\dots,x_n \in \mathcal{X}} |N_{\mathcal{F}}(x_1, \dots, x_n)|$

- Where $N_{\mathcal{F}}$ is the set of possible labellings for elements $x_1, \dots, x_n$.

Therefore, the shatter-coefficient is the maximum possible labellings of n training points when using the functions in $\mathcal{F}$.

In the example above, we have $\mathcal{S}(\mathcal{F}, n) = n + 1$, because all points to the left of t are 0 and all points to the right of t are 1, thus there are $n + 1$ total combinations of labellings.
(*Example for n=5:* $N_f = \{00000, 00001, 00011, 00111, 01111, 11111\}$)

Vapnik-Cheronenkis (VC) dimension

It's possible to derive meaningful dimensions, and in turn a meaningful bound in the risk, using this Shatter-Coefficient. But for this, we need a *VC dimension*.

Vapnik-Cheronenkis (VC) dimension: The largest integer k such that $\mathcal{S}(\mathcal{F}, k) = 2^k$. This is denoted as $VC(\mathcal{F})$.

Thus, this is the largest k for which all labellings are still possible. In the example above, we have that $VC(\mathcal{F}) = 1$.

Normally, to prove a certain class has VC dimension v, we need to show two things:

1. Easy - For $n \leq v$, there is a set of n points that can be shattered by this class.
   In other words, show that $\mathcal{S}(\mathcal{F}, v) = 2^v$ by giving an example.
2. Hard - No set of $n > v$ points can be shattered, no matter how it is chosen.
   In other words, we need to proof that some labelling is impossible with our class $\mathcal{F}$ no matter how the points $x_1, \dots, x_n$ are distributed.

Some extra things to note about the VC dimension:

- The VC dimension typically coincides more or less with the number of parameters needed to describe an element of the model class. (Not always, though)
- There are classes, such as $f(x) = \mathbf{1}\{\sin(\omega x) \geq 0\}$, where the VC dimension is infinite.

Using the VC dimension in bounds

Now we have the shatter coefficient, which measures the "effective" size of $\mathcal{F}$ when looking through the lens of n training points. Our goal is to get similar probablistic bounds on the excess risk as in earlier chapters. With the help of very clever probablistic arguments, a generalization bound has been derived, as well as the expected excess risk.

Theorem 11.3.1 (VC inquality)

We can use the shatter coefficient a *generalization bound* on the excess risk:

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}|\hat{R}_n(f) - R(f)| > \epsilon\right) \leq 8\mathcal{S}(\mathcal{F}, n)e^{\frac{-n\epsilon^2}{32}}$$

As well as using the shatter coefficient in an upper bound *expected excess risk:*

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}|\hat{R}_n(f) - R(f)|\right] \leq 2\sqrt{\frac{\log\mathcal{S}(\mathcal{F}, n) + \log 2}{n}}$$

If you compare these to the bounds in Chapter 7 [INSERT REFERENCE], then these are quite similar in terms of complexity. (Notice that the *supremum* just means that we grab the $f \in \mathcal{F}$ that makes the excess risk a large as possible.)

Still, how can we use the VC dimension for the bounds? This can be done using *Sauer's Lemma*.

Lemma 11.3.1 (Sayer's Lemma)

$$\mathcal{S}(\mathcal{F}, n) \leq (n+1)^{VC(\mathcal{F})}$$

Finally, if you use *empirical risk minimization* to pick your f, then we get the following bound on the expected excess risk.

Corollary 11.3.1

Let $\hat{f} = \arg\min_{f\in F}\hat{R}_n(f)$ be the empricial risk minimizer. Then it holds that:

$$\begin{aligned}\mathbb{E}[R(\hat{f}_n)] - \inf_{f\in F}R(f) &\leq 4\sqrt{\frac{\log\mathcal{S}(\mathcal{F}, n) + \log 2}{n}} \\ &\leq 4\sqrt{\frac{\log VC(\mathcal{F})\log(n+1) + \log 2}{n}}\end{aligned}$$

Linear classifiers

Normal Linear Classifiers

If you consider the class of linear classifiers, then this class is infinite.

$$\mathcal{F} = \left\{ f(x) = \mathbf{1}\{\mathbf{w}^T\mathbf{x} + w_0 > 0\} \; ; \; \mathbf{x}, \mathbf{w} \in \mathbb{R}^d, w_0 \in \mathbb{R} \right\}$$

However, for the case of two dimensions ($d = 2$), we can see that the VC dimension of this class is 3. This can be generalized for d dimensions, where $VC(\mathcal{F}) = d + 1$.

So, if $\mathcal{F}$ is the class of hyperplance classifiers and we use the empirical risk minimizer, then:

$$\mathbb{E}[R(\hat{f}_n)] \leq \inf_{f \in \mathcal{F}} R(f) + 4\sqrt{\frac{(d+1)\log(n+1) + \log 2}{n}}$$

Generalized Linear Classifiers

Creating more features

We can actually create nonlinear classifier using our linear classifier + forward generalization.Let $\phi(X) = (\phi_1(X), \phi_2(X), \ldots, \phi_{d'}(X))^T$ be a collection of function mappings. For example, if $X = (x_1, x_2)^T$, then we could consider the polynomial features $\phi = (x_1, x_2, x_1x_2, x_1^2, x_2^2)^T$ instead.

What we do is increase the dimension of the features from $d = 2 \rightarrow d = 5$ and the VC bounds immedidtely extends for this case, replacing d with d'.

SVM sidenote

However, solving empricial risk minimization with hyperplane leads to an NP-hard problem. If we instead use the hinge-loss, it's much easier to solve, giving rise to *Support Vector Machines (SVMs)*. Even better, as this problem only consists of inner products, we can replace it by a kernel (because this corresponds to an inner product in some different space). Basically, we're doing to same thing above without explicitly computing the higher dimensional feature space (*kernel-trick*).

Decision Trees

Tree class

Let $\mathcal{T}_k$ be a very general class of tree classifiers defined as follows:

$$\mathcal{T}_k = \left\{ \text{classifiers based on recursive rectangular partitions of } \mathbb{R}^d \text{ with } k+1 \text{ cells} \right\}$$

Each split can potentially shatter $d + 1$ points (similar to linear classifiers), so with k splits we have that $VC(\mathcal{T}_k) \leq (d+1)k$. When we apply this VC bound to trees, we get an excess risk where both the approximation error and estimation error depend on k. How to minize this? Via Structural Risk Minimization (SRM)!

Structural Risk Minimization

SRM is complexity regularization using VC type bounds instead of Hoeffding's inequality. The idea is to use the VC inquality and see how big the difference between the empirical and true risk can be for any element of $\mathcal{F}$. This way, we can get a bound on the excess risk. After some mathematics and a union bound, we get:

$$\forall_{f \in \mathcal{F}} |\hat{R}_n(f) - R(f)| \leq \sqrt{\frac{VC(\mathcal{F}_{k(f)}) \log(n+1) + 3 + k(f) \log 2 + log(1/\delta)}{2n}}$$

So, similar to Chapter 8, we can go for the *minimum penalized empirical risk predictor* by taking a combination of the error on the data set and a regularization term. After some more mathematics, taking $\delta = \frac{1}{\sqrt{n+1}}$, and applying the VC dimension of trees, we get the following bound for decision trees:

$$\mathbb{E}[R(\hat{f}_n)] - R^* \leq \inf_k \left\{ \inf_{f \in \mathcal{F}_k} \left\{ R(f) - R^* + 8\sqrt{\frac{(k(d+1) + 1/2) \log(n+1) + 3 + k \log 2)}{2n}} \right\} \right\} + \frac{1}{\sqrt{n+1}}$$

If you compare this with the bounds of Chapter 9, we get a bound with essentially the same form, although now we're condiering a much richer class of classification rules. Compared to recursive dyadic partitions, we have an uncountable number of trees with k cells here.

Chapter 12 - Denoising of Piecewise Smooth Functions