Learning Theory

Muchang Bahng

Spring 2025

Contents

Unlike unsupervised learning, which comes in many different shapes and forms (anomaly detection, feature extraction, density estimation, dimensionality reduction, etc.), supervised learning comes in a much cleaner format. In supervised learning, we consider an input space $\mathcal{X}$ and an output space $\mathcal{Y}$. We assume that there exists some unknown measure $\mathbb{P}$ over $\mathcal{X} \times \mathcal{Y}$, making this some probability space. We then assume that some data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}$ is generated sampled *independently and identically (iid)* from $\mathbb{P}$. Now this assumption is quite strong and is almost always not the case, as different data can be correlated, but we will relax this assumption later. Let's formally construct this from the bottom up.

1. We start off with a general probability space $(\Omega, \mathcal{F}, \mathbb{P})$. This is our model of the world and everything that we are interested in.
2. A measurable function $X : \Omega \to \mathcal{X}$ extracts a set of features, which we call the **covariates** and induces a probability measure on $\mathcal{X}$, say $\mathbb{P}_X$.
3. Another measurable function $Y : \Omega \to \mathcal{Y}$ extracts another set of features called the **labels** and induces another probability measure on $\mathcal{Y}$, the **label set**, say $\mathbb{P}_Y$.
4. At this point the function $X \times Y$ is all we are interested in, and we throw away Ω since we only care about the distribution over $\mathcal{X} \times \mathcal{Y}$.
5. We model the generation of data from Ω by sampling N samples from $\mathbb{P}_{X \times Y}$, which we assume to be iid (this assumption will be relaxed later). This gives us the **dataset**

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$$

Now our goal is to construct a function $f : \mathcal{X} \to \mathcal{Y}$ that predicts Y from X, but we want to define some measure of how good our function is. We can use a loss function L to talk about this.

Definition 0.1 (Risk)

The **risk**, or **expected risk**, of function f is defined as

$$R(f) = \mathbb{E}_{X \times Y}[L(Y, f(X))] = \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(x))\, d\mathbb{P}(x, y) \tag{1}$$

Clearly, we don't know what this risk is since we don't know the true measure $\mathbb{P}$, so we try to approximate it with the *empirical risk*.

Definition 0.2 (Empirical Risk)

The **empirical risk** of function f is defined as

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)})) \tag{2}$$

Definition 0.3 (Generlize)

A function f is said to **generalize** if

$$\lim_{n \to +\infty} \hat{R}_n(f) = R(f) \tag{3}$$

This gives us a way of computing with the actual data. Now two questions arise from this. First, how do we even choose the loss function L? Second, how do we know that the empirical risk is a good approximation of the true risk? The first question can be quite convoluted, but we introduce it with decision theory. The second has a simple answer with concentration of measure.

1 Decision Theory

How can we choose our loss functions? There are two ways of doing this, either through model assumptions or with domain knowledge. When talking about model assumptions, we assume that the residual distribution is of certain form, and the maximum likelihood formulation leads to a certain loss function. For example, assuming that the residuals are normally distributed leads to the squared loss or Laplacian residuals leads to the absolute value loss. These are just modeling assumptions, and if there are no specific assumptions, we are lost. The other way is through domain expertise which allows us to construct our own loss functions. Fortunately, there is a deeper theory behind the choice of loss functions, known as decision theory, which allows us to define loss functions from the get go rather than assume distributions taking particular forms.[1]

Definition 1.1 (Misclassification Loss)

The **misclassification loss** is defined as

$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{if } y \neq \hat{y} \end{cases} \tag{4}$$

Example 1.1 (Misclassification Risk)

Substituting the misclassification loss function into the risk gives the **misclassification risk**.

$$R(f) = \mathbb{E}[\mathbb{1}_{\{Y \neq f(X)\}}] = \mathbb{P}(Y \neq f(X)) \tag{5}$$

and therefore our empirical risk is

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{y^{(i)} \neq f(x^{(i)})\}} \tag{6}$$

which is just the number of misclassifications over the total number of samples.

However, depending on the context, the loss for misclassification one one label can be quite different from that of another label. Consider the medical example where you're trying to detect cancer. Falsely detecting a non-cancer patient as having cancer is not as bad as falsely detecting a cancer patient as not having cancer.

Definition 1.2 (Weighted Misclassification Loss)

The **loss matrix** K defines the loss that we incur when predicting the ith class on a sample with true label j.

$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ K_{ij} & \text{if } y = i \neq j = \hat{y} \end{cases} \tag{7}$$

Definition 1.3 (Squared Loss)

The **squared loss** is defined as

$$L(y, \hat{y}) = (y - \hat{y})^2 \tag{8}$$

[1] Credits to Edric for telling me this.

Example 1.2 (Mean Squared Risk)

Substituting the squared loss function into the risk gives the **mean squared risk**.

$$R(f) = \mathbb{E}[(Y - f(X))^2] \tag{9}$$

and therefore our empirical risk is

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - f(x^{(i)})^2 \tag{10}$$

Definition 1.4 (Absolute Loss)

The **absolute loss** is defined as

$$L(y, \hat{y}) = |y - \hat{y}| \tag{11}$$

2 Function Classes

Now that we've defined the risk and empirical risk, the true function that we want to find is the one that minimizes the empirical risk.

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \hat{R}(f) \tag{12}$$

However, this depends on the function space $\mathcal{F}$ that we are minimizing over. If we chose f to be the space of all functions, then we just interpolate (fit perfectly over) the data[2], which is not good since we're **overfitting**. This is a problem especially in nonparametric supervised learning, and there are generally two ways to deal with this. The first is to use *localization*, which deals with local smoothing methods. The second is with **regularization**. The third is to restrict our class of functions to a smaller set. Perhaps we assume that nature is somewhat smooth and so naturally we want to work with smooth functions. There are two ways that we define smoothness, through Holder spaces that focus on local smoothness and Sobolev spaces that focus on global smoothness.

Definition 2.1 (L^p Space)

The $L^p(\mu)$ space is the normed vector space of all functions from $f : \mathcal{X} \to \mathbb{R}$ such that

$$||f||_p = \left(\int |f(x)|^p \, d\mu \right)^{1/p} < \infty \tag{13}$$

Theorem 2.1 (Countable Basis)

You can construct a countable orthonormal basis in $L^2(\mu)$ space.

There are a lot of well known orthonormal bases. For example, the Fourier basis, Legendre polynomials, Hermite polynomials, or wavelets. Therefore, every function can be expressed as a linear combination of this basis, and you can calculate coefficients by taking the inner product with the basis functions.

$$f(x) = \sum_{i=1}^{\infty} \alpha_i \phi_i(x) \text{ and } \alpha_i = \langle f, \phi_i \rangle \tag{14}$$

Now we can define Holder spaces. Holder spaces are used whenever we want to talk about local smoothness. For example, when we want to talk about local smoothing methods for regression and classification, talking

[2] unless there were two different values of Y for the same X

about this smoothing is not quite possible if we don't have certain assumptions on the function. To make theory easier, we assume that the function has basic smoothness properties and this property is Holder smoothness. But note that these are ultimately assumptions.

Definition 2.2 (Holder Space)

For some $\beta \in \mathbb{N}$ and $L \in \mathbb{R}^+$, the $H(\beta, L)$ **Holder space** is the set of all functions $f : \mathcal{X} \subset \mathbb{R} \to \mathbb{R}$ such that

$$|f^{(\beta-1)}(y) - f^{(\beta-1)}(x)| \leq L||y - x|| \tag{15}$$

for all x, y. If we want $\mathcal{X}$ to be d-dimensional, then we want to bound the higher order total derivatives, and so $H(\beta, L)$ becomes all functions $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ such that for all $\mathbf{s} = (s_1, \dots, s_d)$ with $|\mathbf{s}| = \beta - 1$,

$$|D^{\mathbf{s}} f(x) - D^{\mathbf{s}} f(x)| \leq L||y - x|| \tag{16}$$

for all $x, y \in \mathcal{X}$, where

$$D^{\mathbf{s}} = \frac{\partial^{|\mathbf{s}|}}{\partial x_1^{s_1} \dots \partial x_d^{s_d}} \tag{17}$$

The higher β is, the more smoothness we're demanding.

If $\beta = 1$, then this reduces to the set of all Lipschitz functions. It is most common to assume that $\beta = 2$, which means that the derivative is Lipschitz. This is not rigorously true, but by dividing both sides by $||y - x||$ and taking the limit to 0, we can say that it implies that there exists some finite second derivative bounded by L.

Definition 2.3 (Sobolev Space)

The **Sobelov space** $W_{m,p}$ is the space of all functions $f \in L_p(\mu)$ such that

$$||D^m f||_p \in L^p(\mu) \tag{18}$$

This is slightly stronger than the usual definition of Sobolev spaces since we requiring the derivative rather than the weak derivative. So m tells us how many derivatives we want well behaved and p tells us under which norm are the derivatives well behaved.

Now there is a related definition of a Sobelov ellipsoid that we'll be working with.

Definition 2.4 (Sobelov Ellipsoid)

Let $\theta = (\theta_1, \theta_2, \dots)$ be a sequence of real numbers. Then the set

$$\Theta_m = \left\{ \theta \mid \sum_{j=1}^{\infty} a_j^2 \theta_j^2 < C^2 \right\} \tag{19}$$

where $a_j^2 = (\pi \cdot j)^{2m}$. Note that since a_j is exploding, to stay finite the θ_j must be decaying.

This is useful because of the following theorem.

Theorem 2.2 (Conditions for Function being in Sobelov Space)

Given a function $f \in L^2(\mu)$ expanded in some orthonormal basis ϕ_j, then $f \in W_{m,2}$ if and only if the coefficients α_j die off fast enough in the sense that it is in the Sobelov ellipsoid.

Now let's talk about RKHS. Let's take the $L^2(\mu)$ space of functions $f : [0, 1] \to \mathbb{R}$ with $||f|| = \int f^2 \, d\mu < \infty$ and inner product $\langle f, g \rangle = \int f(x) g(x) \, d\mu$. It is known that if f_n converges to f in L^2, then it is not necessarily

true that f converges pointwise since it can diverge on a sequence of sets that converge to measure 0. You probably don't want to work with functions that look like this, and that's what a RKHS is for. It gives you a nice class of functions that have good statistical properties but also are easy to compute with.

Definition 2.5 (Mercer Kernels)

A **Mercer kernel** is a function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that is symmetric and positive definite in the sense that for any collection $x_1, \dots, x_n$ of arbitrary size n,

$$\sum_i \sum_j c_i c_j K(x_i, x_j) \geq 0 \tag{20}$$

which is equivalent to saying that the matrix formed by evaluating these kernels at the pairs of points is positive semi-definite.

Example 2.1 (Gaussian Kernel)

The Gaussian kernel is defined

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{\sigma^2} \right) \tag{21}$$

Now this kernel should tell us roughly how similar two points x and y are. Using this kernel, we want to build a function space. For this, we need Mercer's theorem.

Theorem 2.3 (Mercer's Theorem)

If we have a kernel K that is bounded

$$\sup_{x,y} K(x, y) < \infty \tag{22}$$

we can define a new operator T_K that maps functions to functions

$$T_K f(x) = \int K(x, y) f(y)\, dy = \iint K(x, y) f(x) f(y)\, dx\, dy \tag{23}$$

This operator is linear, meaning that it has an eigendecomposition and therefore there exists eigenfunctions ϕ_i s.t.

$$T_K \phi_i(x) = \int K(x, y) \phi_i(y)\, dy = \lambda_i \phi_i(x) \tag{24}$$

Then these eigenvalues are bounded and we can write the kernel as a sum of the eigenfunctions.

$$\sum_i \lambda_i < \infty, \quad K(x, y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(y) \tag{25}$$

These ϕ_i's are the implicit high-dimensional features.

What do these eigenfunctions ϕ_i look like? Well, they tend to look like functions that tend to get wigglier and wigglier as i increases, indicating that λ_i must decrease in such a way that it still keeps the function smooth.

Now, we can fix the first term in the kernel and it will be function of the second term $K_x(\cdot) = K(x, \cdot)$. We do this for all $x \in \mathbb{R}$, which form the basis of our RKHS, and it consists of all functions that are linear

combinations of these K_x's. For example, the functions

$$f = \sum_i \alpha_i K_{x_i} \text{ and } g = \sum_j \beta_j K_{x_j} \tag{26}$$

can consist of a finite number of perhaps different basis functions. Now this is clearly a vector space, and to upgrade this to a Hilbert space, we must define an inner product. This inner product (with respect to some kernel K) is defined as

$$\langle f, g \rangle_K = \sum_{i,j} \alpha_i \beta_j K(x_i, x_j) \tag{27}$$

Exercise 2.1 (Inner Product of RKHS)

Show that the inner product of the RKHS is indeed an inner product.

The inner product induces a norm, and so by taking the completion of all linear combinations of the kernel basis functions we get our RKHS. Now since K_x is itself in the RKHS, we can take the inner product of f and K_x, which just gives us back the evaluation of f at x.

Definition 2.6 (Reproducing Kernel Hilbert Space)

Given a kernel K, the **reprducing kernel Hilbert space** $\mathcal{H}$ is the Hilbert space of all functions $f : \mathcal{X} \to \mathcal{Y}$ that can be expressed as a linear combination of the functions $\{K_x = K(x, \cdot)\}$. It has the inner product

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i,j} \alpha_i \beta_j K(x_i, x_j) \tag{28}$$

and also includes all of its limit points under this norm, making it a complete space.

Theorem 2.4 (Reproducing Property of RKHS)

An RKHS satisfies the **reproducing property**, which means that taking the inner product of a function f and a kernel K_x gives you the evaluation of f at x.

$$\langle f, K_x \rangle_{\mathcal{H}} = f(x) \tag{29}$$

and therefore it also means that $\langle K_x, K_x \rangle_{\mathcal{H}} = K(x, x)$. This also means that K_x is the evaluation functional in the dual space of $\mathcal{H}$ and this evaluation functional δ_x is continuous, which is not always true in functional analysis.

Proof.

We can evaluate from the inner product

$$f = \sum_i \alpha_i K_{x_i} \implies \langle f, K_x \rangle_K = \sum_i \alpha_i \langle K_{x_i}, K_x \rangle_K = \sum_i \alpha_i K(x_i, x) = f(x) \tag{30}$$

This reproducing property tends to be very useful, especially in the corollary below.

Corollary 2.1 (Convergence in RKHS)

Convergence in norm implies pointwise convergence in RKHS.

Proof.

Given that $f_n \to f$ in norm, we have that $||f_n - f|| \to 0$. Then for all points $x \in \mathcal{X}$,

$$|f_n(x) - f(x)| = |\langle f_n - f, K_x \rangle_{\mathcal{H}}| \leq ||f_n - f|| \cdot ||K_x|| \to 0 \tag{31}$$

Theorem 2.5 (Moore-Aronszajn)

Any positive definite function K is a reproducing kernel for some RKHS.

Proof.

We won't be too rigorous about this since this is not a functional analysis course. Assume that we have a positive definite kernel $K : X \times X \to \mathbb{R}$, where X is some measurable set, and we will show how to make a RKHS $\mathcal{H}_k$ such that K is the reproducing kernel on $\mathcal{H}$. It turns out that $\mathcal{H}_k$ is unique up to isomorphism. Since X exists, let us first define the set $S = \{k_x \mid x \in X\}$ such that $k_x(y) \coloneqq K(x, y)$. Now let us define the vector space V to be the span of S. Therefore, each element $v \in V$ can be written as

$$v = \sum_i \alpha_i k_{x_i}$$

Now we want to define an inner product on V. By expanding out the vectors w.r.t. the basis and the properties of bilinearity, we have

$$\langle k_x, k_y \rangle_V = \left\langle \sum_i \alpha_i k_{x_i}, \sum_i \beta_i k_{y_i} \right\rangle = \sum_{i,j} \alpha_i \beta_j K(x_i, y_j)$$

At this point, V is not necessarily complete, but we can force it to be complete by taking the limits of all Cauchy sequences and adding them to V. In order to complete the construction, we need to ensure that K is continuous and doesn't diverge, i.e.

$$\iint K^2(x, y)\, dx\, dy < +\infty$$

which is a property known as finite trace.[a]

[a] Too much to write down here at this point, but for further information look at *thearticlehere*.

Now at first glance, this abstract construction makes it hard to determine what kind of functions there are in a RKHS generated by some kernel. Conversely, given some RKHS, it's not always easy to know which kernel it came from.

Example 2.2 (Fourier Basis)

Let us take the vector space of all real functions f for which its Fourier transform is supported on some finite interval $[-a, a]$. This is a RKHS with the kernel function

$$K(x, y) = \frac{\sin(a(y - x))}{a(y - x)} \tag{32}$$

with the inner product $\langle f, g \rangle = \int f(x) g(x)\, dx$.

Example 2.3 (Some Sobelov Spaces are RKHS)

Let us take the Sobelov space $W_{1,2}$ of all functions $f : [0, 1] \to \mathbb{R}$ satisfying

$$\int (f'(x))^2 \, dx < \infty \tag{33}$$

This is a RKHS with the kernel function

$$K(x, y) = \begin{cases} 1 + xy + \frac{xy^2}{2} - \frac{y^3}{6} & \text{if } 0 \leq y \leq x \leq 1 \\ 1 + xy + \frac{x^2 y}{2} - \frac{x^3}{6} & \text{if } 0 \leq x \leq y \leq 1 \end{cases} \tag{34}$$

Finally, remembering Mercer's theorem, we can decompose the Kernel into its eigenfunctions

$$K(x, y) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(y) \tag{35}$$

When you talk about feature maps (e.g. in support vector machines), you're really just creating the map from $x \in \mathcal{X}$ into the infinite dimensional vector space

$$x \mapsto \Phi(x) = \left(\sqrt{\lambda_1} \phi_1(x), \sqrt{\lambda_2} \phi_2(x), \dots \right) \tag{36}$$

and the inner product between two functions is actually the inner product between their feature maps. Therefore, you can either just work with x in the RKHS or work with the features Φ in a higher dimensional Euclidean space. Therefore, we can either work with f as a combination of kerenels or a linear combination of the eigenfunctions. The eigenfunctions are easier conceptually, but when we actually do computations, the kernel expansion is much easier.

$$f(x) = \sum_i \alpha_i K(x_i, x) = \sum_j \beta_j \phi_j(x) \tag{37}$$

When you're expanding with the eigenfunctions, you can just compute the inner product as

$$\langle f, g \rangle = \sum_i \frac{\alpha_i \beta_i}{\lambda_i} \tag{38}$$

and because f, g must satisfy some smoothness constraints, the α_i and β_i must die off quickly, making the sum finite. But we're never going to be actually computing this way since it's much easier to compute with the kernel expansion. This means that the ϕ_i's, which get wigglier (think of sine and cosine eigenbases) as i increases, must have decreasing coefficients.

When working with function classes, we tend to divide them into two broad categories.

Definition 2.7 (Parametric Models)

A **parametric model** is a set of functions $\mathcal{M}_{\boldsymbol{\theta}}$ that can be parameterized by a finite-dimensional vector. The elements of this model are hypotheses functions $h_{\boldsymbol{\theta}}$, with the subscript used to emphasize that its parameters are $\boldsymbol{\theta}$. We have the flexibility to choose any form of h that we want, and that is ultimately a model assumption that we are making.

Example 2.4 (Examples of Parametric Models)

1. If we assume $h : \mathbb{R}^D \to \mathbb{R}$ to be linear, then h lives in the dual of $\mathbb{R}^D$, which we know to be D-dimensional.
2. If we assume h to be affine, then this just adds one more dimension.

3. If we assume $h : \mathbb{R} \to \mathbb{R}$ to be a kth degree polynomial, then g can be parameterized by a $k+1$ dimensional θ.

However, parametric models may be limited in the way that we are assuming some form about the data. For certain forms of data, where we may have domain knowledge, it is reasonable to use parametric models, but there are cases when we will have absolutely no idea what the underlying distribution is. For example, think of classifying a $3 \times N \times N$ image as a cat or a dog. There is some underlying distribution in the space $[255]^{3N^2} \times \{\text{cat}, \text{dog}\}$, but we have absolutely no idea how to parameterize this. Should it be a linear model or something else? This is when nonparametric models come in. They are not restricted by the assumptions concerning the nature of the population from which the sample is drawn.

Definition 2.8 (Nonparametric Models)

Nonparametric models are ones that cannot be expressed in a finite set of parameters. They may be countably or uncountably infinite.

3 Concentration of Measure

Concentration of measure is a tool used to prove a lot of theorems in statistical machine learning. I have another series of notes on this, but we'll stick to the key points.

Definition 3.1 (Hoeffding's Inequality)

Given $X_1, \dots, X_n$ are iid random variables with $a \leq X_i \leq b$, then for any $\epsilon > 0$,

$$\mathbb{P}\left(\left| \frac{1}{n} \sum_{i=1}^{n} X_i - \mathbb{E}[X] \right| \geq \epsilon \right) \leq 2 \exp\left(- \frac{2n\epsilon^2}{(b-a)^2} \right) \tag{39}$$

Therefore, if we apply it to some binary classifier $f : \mathcal{X} \to \{0, 1\}$, then we can say that the probability that the empirical risk deviates from the true risk is exponentially small.

$$\mathbb{P}(|\hat{R}(f) - R(f)| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \tag{40}$$

But when we do empirical risk minimization (ERM), we not given a classifier, but we must *choose* it. So given our space of classifiers f, we can plot the true risk and the noisy empirical risk. The equation above states that at any given point the probability of it deviating by more than ϵ is exponentially small. But we want something stronger: we want to bound the probability of the supremum of the difference over the whole class $\mathcal{F}$.

$$\mathbb{P}\big(\sup_{f \in \mathcal{F}} |\hat{R}(f) - R(f)| \geq \epsilon \big) \tag{41}$$

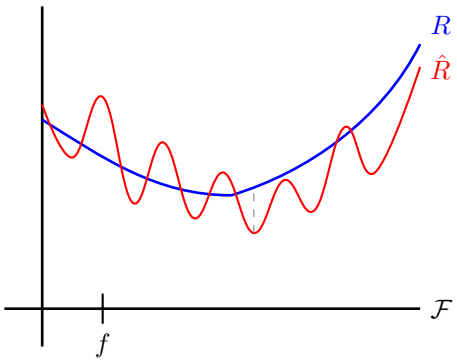


Figure 1: True risk of functions over $\mathcal{F}$ and its noisy empirical risk. We want to bound the maximum deviation of these two over the whole class.

This bound will depend on how *complex* the function class $\mathcal{F}$ is, and to measure this complexity, we introduce some definitions.

Definition 3.2 (Rademacher Complexity)

Given **Rademacher random variables** $\sigma_1, \dots, \sigma_n$ with $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$, the **Rademacher complexity** of a function class $\mathcal{F}$ is defined

$$\text{Rad}_n(\mathcal{F}) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(Z_i) \right| \right] \tag{42}$$

where the expectation is across the random σ_i's and the Z_i's, which are independent.

To get some intuition of what this is, let's consider a function class of a single function f. Then, the sup disappears and the term inside the absolute value sign becomes a 0-mean random variable. Now if we have a very complex function class $\mathcal{F}$ with a lot of "wiggly" functions, then this value should be large. In this case, imagine a game where you pick generate some random variables σ_i and the Z_i. Then, I pick a function that maximizes this value. How can I do that? If I can find a function f that matches the sign of the σ_i's ($+1$ or -1) at each of the values of Z_i, then this would be maximized. Therefore, if I have a sufficiently complex class, then I can pick a function that tracks your σ_i's. Another way of looking at it is given noise variables σ and Z, we're looking at the correlation between σ and $f(Z)$. If we can maximize this correlation, then this is a complex class.

Now this is the most natural way of defining the complexity of the class, and in some cases it can be explicitly computed. However, in most cases it cannot be, but it can be bounded be something that is computable, like the VC dimension.

Lemma 3.1 (Bigger Class, Bigger Complexity)

If $\mathcal{F} \subset \mathcal{G}$, then $\text{Rad}_n(\mathcal{F}) \leq \text{Rad}_n(\mathcal{G})$.

Lemma 3.2 (Convex Hull)

If $\mathcal{F}$ is a convex set, then $\text{Rad}_n(\mathcal{F}) = \text{Rad}_n(\text{conv}(\mathcal{F}))$, where $\text{conv}(\mathcal{F})$ is the convex hull of $\mathcal{F}$.

This lemma is quite useful since if we have a certain finite set of functions, then their convex hull can encompass quite a bit, and we can also easily compute that convex hull's Rademacher complexity. Since the extremes haven't changed, the complexity doesn't change, and this might suggest that the Rademacher complexity is a good measure.

Lemma 3.3 (Change of Complexity with Lipschitz Functions)

Consider a L-Lipschitz function g with $g(0) = 0$ and consider the class $\mathcal{F}$, then we can bound the class of functions $g \circ \mathcal{F} = \{g \circ f \mid f \in \mathcal{F}\}$.

$$\text{Rad}_n(g \circ \mathcal{F}) \leq 2L\text{Rad}_n(\mathcal{F}) \tag{43}$$

This constant multiplicative bound is also useful.

Definition 3.3 (Projection of Function Class onto Points)

Given a binary function class $\mathcal{F}$ with functions $f : \mathcal{X} \to \{0, 1\}$, let us denote the projection of $\mathcal{F}$ onto a set of points $z_1, \dots, z_n \in \mathcal{X}$ to be

$$\mathcal{F}_z = \mathcal{F}_{z_1,\dots,z_n} = \{(f(z_1), \dots, f(z_n)) \mid f \in \mathcal{F}\} \tag{44}$$

This projection determines the set of all possible binary labels that can be perfectly classified by some function f.

Definition 3.4 (Shattering Number)

The **shattering number** of $\mathcal{F}$ is defined

$$s_n(\mathcal{F}) = s(\mathcal{F}, n) = \sup_{z_1,\dots,z_n} |\mathcal{F}_{z_1,\dots,z_n}| \tag{45}$$

The highest number that this can be is 2^n, since this is the number of possible binary vectors of length n. Given a set of n points $z_1, \dots, z_n$, we say that the function class $\mathcal{F}$ **shatters** this set if $\mathcal{F}_{z_1,\dots,z_n} = |2^n|$. That is, for every one of the 2^n labels on the points, there exists a function that can perfectly classify them.

Example 3.1 (Binary Functions)

Consider the function class $\mathcal{F}$ of all binary functions of the form

$$f(x) = \begin{cases} 1 & \text{if } x > t \\ 0 & \text{if } x \leq t \end{cases} \tag{46}$$

Then, the projection of $\mathcal{F}$ onto some $n = 3$ points is the set

$$\{(0, 0, 0), (0, 0, 1), (0, 1, 1), (1, 1, 1)\} \tag{47}$$

and this is true no matter how I pick the z_1, z_2, z_3, and so the Shattering number is $s_n(\mathcal{F}) = 4$.

Definition 3.5 (VC Dimension)

We know that the shattering number is bounded above by 2^n. For $n = 1$, it is reasonable that it achieves this bound, but as n grows, the Shattering number may die off. The **VC dimension** is the largest n number of points that can be shattered by the function class without misclassification.

$$n^{\text{VC}} := \sup_n \{s_n(\mathcal{F}) = 2^n\} \tag{48}$$

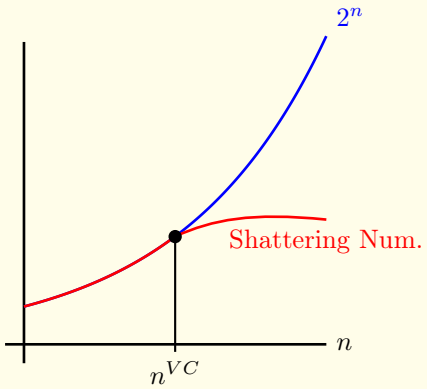


Figure 2: The Shattering number of $\mathcal{F}$ will grow exponentially until it reaches the VC dimension, at which point it will grow polynomially. The point at which it "dies off" is the VC dimension.

It turns out that there are very interesting properties about the VC dimension. One such fact is Sawyer's lemma, which states that if the VC dimension is finite, then the rate of growth of the shattering number suddenly changes from exponential 2^n to polynomial n^{VC}, and this is what makes a lot of machine learning work.

Definition 3.6 (Subgaussian Random Variables)

A random variable X is **subgasussian** if

$$\mathbb{E}[e^{\lambda X}] \leq e^{\frac{\lambda^2 \sigma^2}{2}} \tag{49}$$

Gaussians and bounded random variables are subgaussian.

Lemma 3.4 (Bound on Subgaussian Random Variables)

Given a set of iid subgaussian random variables $X_1, \dots, X_n$

$$\mathbb{E}\big[\max_{1 \leq i \leq d} X_i\big] \leq \sigma \sqrt{2 \log d} \tag{50}$$

Theorem 3.1 (Bound of Rademacher Complexity with Shattering Number)

The Rademacher complexity of a binary function class $\mathcal{F}$ is bounded by

$$\text{Rad}_n(\mathcal{F}) \leq \sqrt{\frac{2 \log s_n(\mathcal{F})}{n}} \tag{51}$$

Proof.

Given the projection $\mathcal{F}_{z_1,\dots,z_n}$, we can use the law of iterated expectations on the Rademacher complexity.

$$\text{Rad}_n(\mathcal{F}) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(Z_i) \right| \right] \tag{52}$$

$$= \mathbb{E}_Z \left[\mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(Z_i) \right| \;\middle|\; Z_1, \dots Z_n \right] \right] \tag{53}$$

Note that in the inner expectation, since $f(Z_i)$ is now fixed, then are bounding a linear combination of a bunch of σ_i's, which are subgaussian. Using the bound above, we can reduce it to

$$\mathbb{E}_Z\left[\sqrt{\frac{2\log|F_{z_1,\dots,z_n}|}{n}}\right] \leq \sqrt{\frac{2\log s_n(\mathcal{F})}{n}} \leq \sqrt{\frac{2d\log n}{n}} \tag{54}$$

However, this is not the best possible bound, and in cases such as K means clustering in high dimensions, this VC bound is terrible. Now we move onto the big VC theorem which now bounds the supremum of the difference between the empirical risk and the true risk. To prove this, we need a few tricks, the first being the symmetrization trick using ghost samples.

Lemma 3.5 (Symmetrization Lemma)

Given a set of random variables $Z_1,\dots,Z_n$ and a function class $\mathcal{F}$, we can define ghost samples $Z_1',\dots,Z_n'$ that are iid copies of $Z_1,\dots,Z_n$. Then, we can bound the Rademacher complexity of the function class $\mathcal{F}$ by

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}|\hat{R}(f)-R(f)|\geq\epsilon\right) \leq 2\mathbb{P}\left(\sup_{f\in\mathcal{F}}|\hat{R}(f)-\hat{R}'(f)|\geq\epsilon/2\right) \tag{55}$$

where $\hat{R},\hat{R}'$ is the empirical risk over the original and ghost samples, respectively.

Proof.

Assume that we have a function f that achieves this minimum. By the triangle inequality,

$$|\hat{R}(f)-R(f)|>t \text{ and } |\hat{R}'(f)-R(f)|<\frac{t}{2} \implies |\hat{R}(f)-\hat{R}'(f)|>\frac{t}{2} \tag{56}$$

We write this again as an indicator function.

$$\mathbb{1}(|\hat{R}(f)-R(f)|>t, |\hat{R}'(f)-R(f)|<\frac{t}{2}) \implies \mathbb{1}(|\hat{R}(f)-\hat{R}'(f)|>\frac{t}{2}) \tag{57}$$

and since the samples and the ghost samples are independent, we can take the probability over the ghost samples to get

$$\mathbb{1}(|\hat{R}(f)-R(f)|>t)\,\mathbb{P}_{Z'}(|\hat{R}'(f)-R(f)|<\frac{t}{2}) \implies \mathbb{P}_{Z'}(|\hat{R}(f)-\hat{R}'(f)|>\frac{t}{2}) \tag{58}$$

and the rest of the proof can be found online.

The reason we want this is that it removes the $R(f)$, which is some unknown true mean that can be hard to deal with since it takes infinite values. It's easier to work with two empirical risks than deal with the true risk.

Theorem 3.2 (VC Theorem/Inequality)

Given a binary function class $\mathcal{F}$, we have

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}|\hat{R}(f)-R(f)|\geq\epsilon\right) \leq 2S(\mathcal{F},n)e^{-n\epsilon^2/8} \approx n^d e^{-n\epsilon^2/8} \tag{59}$$

You can see that the exponential term is from Hoeffding but there is an extra cost of taking the supremum over the whole function class, which is the shattering number.

Proof.

Given $Z_1, \dots, Z_n \sim \mathbb{P}$, we take a new set of random variables $Z_1', \dots, Z_n'$ that are iid copies of $Z_1, \dots, Z_n$, called *ghost samples*.

Therefore, for some classes of sets with finite VC dimension, the shattering term will grow polynomially in n but the exponential term decays faster, which is what makes this work. That's why as n grows, we can get a good bound on the supremum of this difference.

Theorem 3.3 ()

With probability $\geq 1 - \delta$, we have

$$\sup_{f \in \mathcal{F}} |\hat{R}(f) - R(f)| \leq 2\text{Rad}_n(\mathcal{F}) + \sqrt{\frac{\log(2/\delta)}{2n}} \tag{60}$$

4 Bias Variance Noise Decomposition

Let's do some further analysis on this. When you take a supremum over a function class, it decomposes into 3 terms.

1. One of which quantifies how big the function class is (more variance).
2. One of which quantifies the distance between the truth and the function class (bias).
3. One is the noise term, which is the irreducible error.

Example 4.1 (Bias and Variance Tradeoff in Polynomial Regression)

Let's motivate this by trying to fit a polynomial on some data.

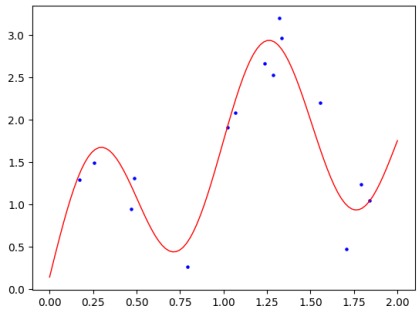
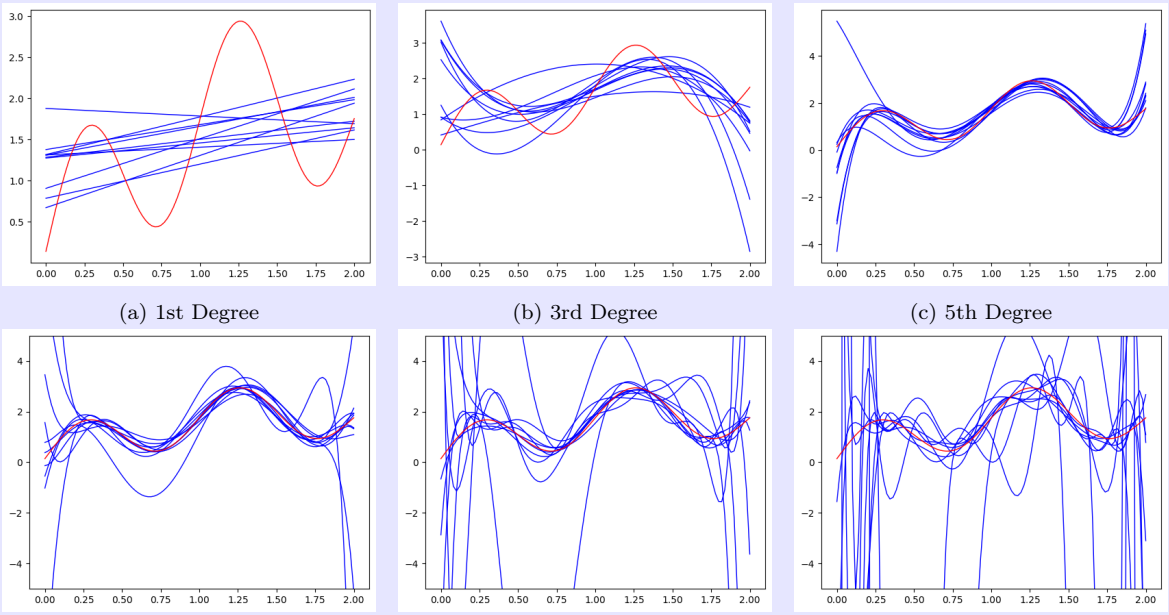


Figure 3: A sample of $|\mathcal{D}| = 15$ data points are generated from the function $f(x) = \sin(2\pi x) + 2\cos(x - 1.5)$ with Gaussian noise $N(0, 0.3)$ on the interval $[0, 1]$.

If we try to fit a polynomial function, how do we know which degree is best? Well the most simple thing is to just try all of them. To demonstrate this even further, I generated 10 different datasets $\mathcal{D}$ of size 15 taken from the same true distribution. The best fitted polynomials for each dataset is shown below.



(a) 1st Degree (b) 3rd Degree (c) 5th Degree

(d) 7th Degree (e) 9th Degree (f) 11th Degree

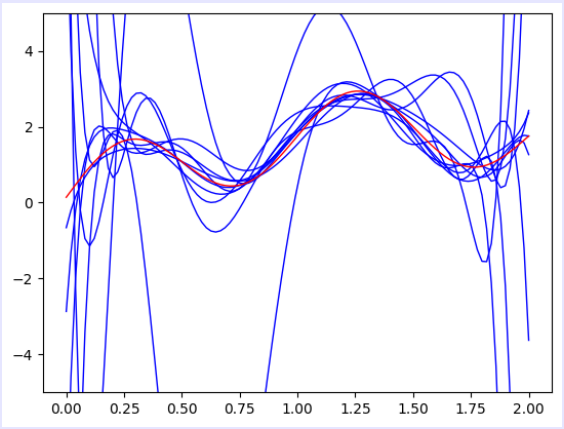
Figure 4: Different model complexities (i.e. different polynomial degrees) lead to different fits of the data generated from the true distribution. The lower degree best fit polynomials don't have much variability in their best fits but have high bias, while the higher degree best fit polynomials have very high variability in their best fits but have low bias. The code used to generate this data is here.

We already know that the 5th degree approximation is most optimal, and the lower degree ones are **underfitting** the data, while the higher degree ones are **overfitting**. As mentioned before, we can describe the underfitting and overfitting phenomena through the bias variance decomposition.

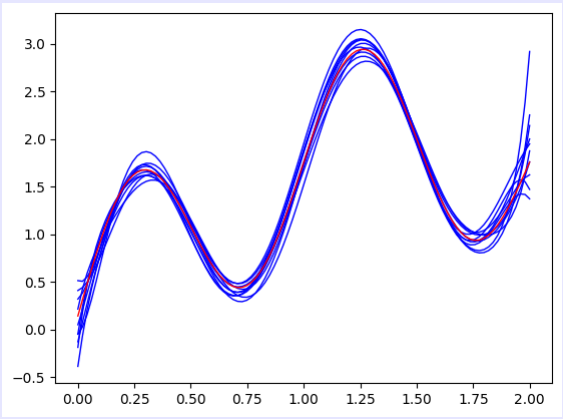
1. If we underfit the data, this means that our model is not robust and does not capture the patterns inherent in the data. It has a high bias since the set of function it encapsulates is not large enough to model $\mathbb{E}[Y \mid X]$. However, it has a low variance since if we were to take different samples of the dataset $\mathcal{D}$, the optimal parameters would not fluctuate.
2. What overfitting essentially means is that our model is too complex to the point where it starts to fit to the *noise* of the data. This means that the variance is high, since different samples of the dataset $\mathcal{D}$ would cause huge fluctuations in the optimal trained parameters $\boldsymbol{\theta}$. However, the function set would be large, and thus it would be close to $\mathbb{E}[Y \mid X]$, leading to a low bias.

Example 4.2 (Polynomial Regression Continued)

Another way to reduce the overfitting problem is if we have more training data to work with. That is, if we were to fit a 9th degree polynomial on a training set of not $N = 15$, but $N = 100$ data points, then we can see that this gives a much better fit. This makes sense because now the random variable $\mathcal{D}$, as a function of more random variables, has lower variance. Therefore, the lower variance in the dataset translates to lower variance in the optimal parameter.



(a) $M = 9, N = 15$



(b) $M = 9, N = 100$

Figure 5: Increasing the number of data points helps the overfitting problem. Now, we can afford to fit a 9th degree polynomial with reasonable accuracy.

5 Minimax Theory