# Elements of Statistical Learning

Nothing is more practical than a good theory (Vapnik, V)

## Basic Concepts

We use $\mathcal{X}$ and $\mathcal{Y}$ to denote the input space and the output space, where typically we have $\mathcal{X} = \mathbb{R}^p$. A joint probability distribution on $\mathcal{X} \times \mathcal{Y}$ is denoted as $P_{X,Y}$. Let $(X, Y)$ be a pair of random variables distributed according to $P_{X,Y}$. We also use $P_X$ and $P_{Y|X}$ to denote the marginal distribution of $X$ and the conditional distribution of $Y$ given $X$.

Let $\mathcal{D}_n = \{(x_1, y_1), ..., (x_n, y_n)\}$ be an i.i.d. random sample from $P_{X,Y}$. The goal of **supervised learning** is to find a mapping $h : \mathcal{X} \to \mathcal{Y}$ based on $\mathcal{D}_n$ so that $h(X)$ is a good approximation of $Y$ . When $Y = \mathbb{R}$ the learning problem is often called regression and when $Y = \{0, 1\}$ or $\{-1, 1\}$ it is often called (binary) classification.

The dataset $\mathcal{D}_n$ is often called the training set (or training data), and it is important since the distribution $P_{X,Y}$ is usually unknown. A learning algorithm is a procedure $\mathcal{A}$ which takes the training set $\mathcal{D}_n$ and produces a predictive model: $\hat{h} = \mathcal{A}(\mathcal{D}_n)$, as the output. Typically the learning algorithm will search over a space of functions $\mathcal{H}$, which we call the **hypothesis space**.

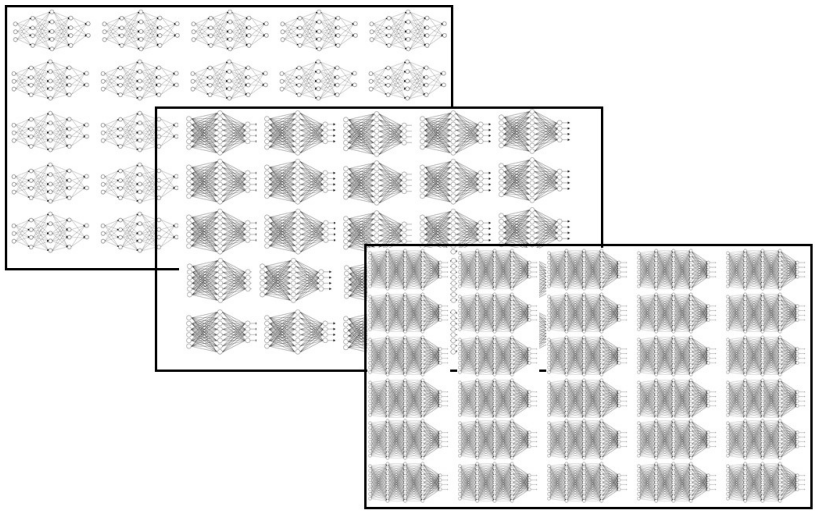

Figure 1: Hypothesis space scheme

## Important Tasks in Machine Learning

1. Finding patterns in large datasets. Association rule discovery in data mining.
2. Clustering
   - Grouping data into clusters that "belong" together - objects within a cluster are more similar to each other than to those in other clusters.
   - Kmeans, Kmedians
   - Input: $\{x_i\}_{i=1}^n$, $x_i \in \mathcal{X} \subset \mathbb{R}^p$.
   - Output: $f : \mathcal{X} \to \{1, ..., K\}$ ($K$ clusters).
   - clustering consumers for market research, clustering genes into families, image segmentation (medical imaging).
3. Classification
   - Input: $\{x_i, y_i\}_{i=1}^n$ "examples," "instances with labels," "observations"
   - $x_i \in \mathcal{X}$, $y_i \in \{-1, 1\}$ "binary"
   - Output: $f : \mathcal{X} \to \mathbb{R}$ and use sign$(f)$ to classify.
   - automatic handwriting recognition, speech recognition, biometrics, document classification.
4. Regression
   - Input: $\{x_i, y_i\}_{i=1}^n$ $x_i \in \mathcal{X}$, $y_i \in \mathbb{R}$
   - Output: $f : \mathcal{X} \to \mathbb{R}$
   - predicting an individual's income, predict house prices, predict stock prices, predict test scores.
5. Ranking in between classification and regression. Search engines use ranking methods
6. Density Estimation
   - predict conditional probabilities
   - Input: $\{x_i, y_i\}_{i=1}^n$
   - $x_i \in \mathcal{X}$, $y_i \in \{-1, 1\}$ "binary"
   - Output: $f : \mathcal{X} \to [0, 1]$ as "close" to $P(y = 1|x)$ as possible
   - estimate probability of failure, probability to default on loan, probability of disease
7. Representation Learning. Low dimensional latent space.
8. Generative modeling.

# Loss Function and Risk

A **loss function** is a mapping $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ (sometimes $\mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$). For example, in binary classification the 0-1 loss function $\ell(y, z) = \mathbf{1}[y \neq z]$ and in regression the squared error loss $\ell(y, z) = (y - z)^2$. Other loss functions include the following:

- misclassification loss: $\mathbf{1}[yz \leq 0] = \mathbf{1}[\text{sign}(z) \neq y]$
- logistic loss: $\log(1 + e^{-yz})$ (logistic regression)
- hinge loss: $\max(0, 1 - yz)$ (SVM)
- exponential loss: $e^{-yz}$ (AdaBoost)

This leads to the idea of convex surrogate loss functions. Since 0-1 loss is hard to optimize (the non-convexity of the 0-1 loss renders intractable a direct minimization of probability of error), you want to optimize something else, instead. Since convex functions are easy to optimize, we want to approximate zero/one loss with a convex function. This approximating function will be called a **surrogate loss**. The surrogate losses we construct will always be upper bounds on the true loss function: this guarantees that if you minimize the surrogate loss, you are also pushing down the real loss.

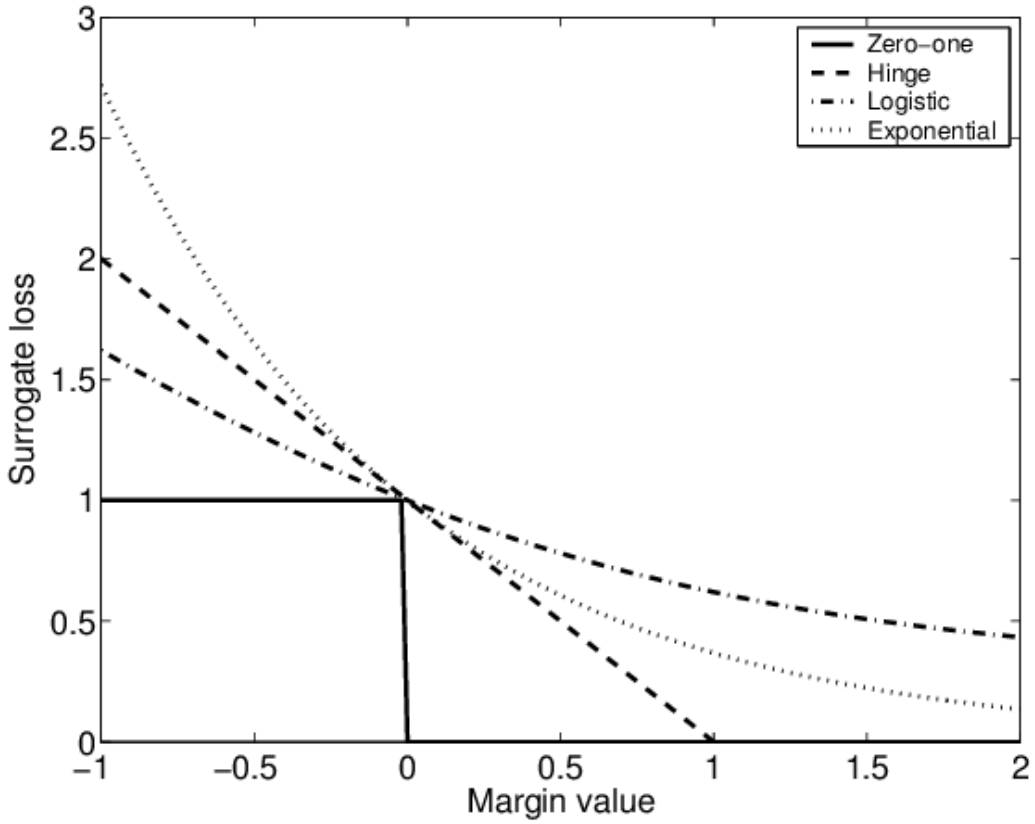


Figure 2: Loss functions

There are two big differences in these loss functions (Fig. 1, margin value means $yf(x)$). The first difference is how "upset" they get by erroneous predictions. In, the case of hinge loss and logistic loss, the growth of the function as $yf(x)$ goes negative is linear. For squared loss and exponential loss, it is super-linear. This means that exponential loss would rather get a few examples a little wrong than one example really wrong. The other difference is how they deal with very confident correct predictions. Once $yf(x) > 1$, hinge loss does not care any more, but logistic and exponential still think you can do better.

The performance of a predictive model $h : \mathcal{X} \to \mathcal{Y}$ is measured by the expected loss, that is, the risk or generalization error:

$$R(h) = E_{X,Y}\Big(\ell(Y, h(X))\Big)$$

where the expectation is taken with respect to the distribution $P_{X,Y}$. Can we calculate $R(h)$?

# Binary classification

Recall that the goal of classification is to learn a mapping from the feature space, $\mathcal{X}$, to a label space, $\mathcal{Y}$. This mapping, $f$, is called a classifier. For example, we might have: $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \{0, 1\}$.

Recalling the risk is defined to be the expected value of the loss function, we have

$$R(f) = E_{X,Y}[\ell(Y, f(X)] = E_{X,Y}\Big[\mathbf{1}[Y \neq f(X)]\Big] = P_{X,Y}(f(X) \neq Y)$$

**Definition (Bayes' Risk)** The Bayes' risk is the infimum of the risk for all classifiers:

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

The performance of a given classifier can be evaluated in terms of how close its risk is to the Bayes' risk. We can prove that the Bayes risk is achieved by the Bayes classifier.

**Definition (Bayes Classifier)** The Bayes classifier is the following mapping

$$h^*(x) = \begin{cases} 1, & P(Y = 1|X = x) \geq 1/2 \\ 0, & \text{otherwise} \end{cases}$$

**Theorem (Risk of the Bayes Classifier)** For any classifier $h$ we have $R(h) \geq R(h^*)$, i.e.

$$R(h^*) = R^*$$

Proof.

Let $g(x)$ be any classifier. We will show that

$$P(g(X) \neq Y) \geq P(h^*(X) \neq Y).$$

For any $X = x$, the conditional error probability of any $g$, may be expressed as

$$\begin{aligned}
P(g(X) \neq Y|X = x) &= 1 - P(g(X) = Y|X = x) \\
&= 1 - \Big(P(Y = 1, g(X) = 1|X = x) + P(Y = 0, g(X) = 0|X = x)\Big) \\
&= 1 - \Big(E\Big[\mathbf{1}[Y = 1]\mathbf{1}[g(X) = 1]|X = x)\Big] + E\Big[\mathbf{1}[Y = 0]\mathbf{1}[g(X) = 0]|X = x)\Big]\Big) \\
&= 1 - \Big(\mathbf{1}[g(x) = 1]E[\mathbf{1}[Y = 1]|X = x)] + \mathbf{1}[g(x) = 0]E[\mathbf{1}[Y = 0]|X = x)]\Big) \\
&= 1 - \Big(\mathbf{1}[g(x) = 1]P(Y = 1|X = x) + \mathbf{1}[g(x) = 0]P(Y = 0|X = x)\Big)
\end{aligned}$$

Letting $\eta(x) := P(Y = 1|X = x)$, we have for any $X = x$.

$$\begin{aligned}
P(g(X) \neq Y|X = x) &- P(h^*(X) \neq Y|X = x) \\
&= \eta(x)\Big(\mathbf{1}[h^*(x) = 1] - \mathbf{1}[g(x) = 1]\Big) + (1 - \eta(x))\Big(\mathbf{1}[h^*(x) = 0] - \mathbf{1}[g(x) = 0]\Big) \\
&= \eta(x)\Big(\mathbf{1}[h^*(x) = 1] - \mathbf{1}[g(x) = 1]\Big) - (1 - \eta(x))\Big(\mathbf{1}[h^*(x) = 1] - \mathbf{1}[g(x) = 1]\Big) \\
&= (2\eta(x) - 1)(\mathbf{1}[h^*(x) = 1] - \mathbf{1}[g(x) = 1])
\end{aligned}$$

where the second equality follows by noting that $\mathbf{1}[g(x) = 0] = 1 - \mathbf{1}[g(x) = 1]$. Next recall

$$h^*(x) = \begin{cases} 1 & \eta(x) \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

For $x$ such that $\eta(x) \geq 1/2$, we have

$$\underbrace{(2\eta(x) - 1)}_{\geq 0} \underbrace{(\underbrace{\mathbf{1}[h^*(x) = 1]}_{1} - \underbrace{\mathbf{1}[g(x) = 1])}_{0 or 1}}_{\geq 0}$$

For $x$ such that $\eta(x) < 1/2$, we have

$$\underbrace{(2\eta(x) - 1)}_{< 0} \underbrace{(\underbrace{\mathbf{1}[h^*(x) = 1]}_{0} - \underbrace{\mathbf{1}[g(x) = 1])}_{0 or 1}}_{\leq 0}$$

which implies

$$(2\eta(x) - 1)(\mathbf{1}[h^*(x) = 1] - \mathbf{1}[g(x) = 1]) \geq 0$$

or

$$P(g(X) \neq Y|X = x) \geq P(h^*(X) \neq Y|X = x)$$

The result follows by integrating both sides with respect $x$. Note that while the Bayes classifier achieves the Bayes risk, in practice this classifier is not realizable because we do not know the distribution $P_{X,Y}$ and so cannot construct $\eta(x)$.

# Regression

In regression we typically have $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \mathbb{R}$. And the risk is often measured by the squared error loss, $\ell(y, z) = (y - z)^2$. The following result shows that for squared error regression, the optimal predictor is the conditional mean function $E[Y|X = x]$.

**Theorem**. Suppose the loss function $\ell(\cdot, \cdot)$ is the squared error loss. Let $h^*(x) = E[Y|X = x]$, then we have $R(h^*) = \inf_f R(f)$.

The proof will be left as an exercise. Thus regression with squared error can be thought as trying to estimate the conditional mean function.

# The estimation-approximation trade-off

Suppose that the learning algorithm chooses the predictive model from the hypothesis space $\mathcal{H}$, and define

$$h^* = \operatorname*{arginf}_{h \in \mathcal{H}} R(h)$$

i.e. $h^*$ is the best predictor among $\mathcal{H}$. Then the *excess risk* of the output $\hat{h}_n$ of the training algorithm is defined and can be decomposed as follows:

$$R(\hat{h}_n) - R^* = \Big(R(\hat{h}_n) - R(h^*)\Big) + \Big(R(h^*) - R^*\Big)$$

The two terms on the right hand side have particular names: the first one is called the **estimation error** and the second one the **approximation error** The reasons for these names are as follows.

The first term deals with the uncertainty introduced by the random sampling process. That is, given the finite sample, we need to estimate the best function in $\mathcal{H}$. Of course, in this process we will make some (hopefully small) error. This error is called the estimation error.

The second term is not influenced by any random quantities, it is deterministic. It deals with the error we make by looking for the best function in a (small) function space $\mathcal{H}$, rather than looking for the best function in the entire space of all functions.

In statistics, estimation error is also called the **variance**, and the approximation error is called the **bias** of an estimator. Originally, these terms were coined for the special situation of regression with squared error loss, but by now people use them in more general settings, like the one outlined above. The intuitive meaning is the same: the first term measures the variation of the risk of the function $\hat{h}_n$ estimated on the sample, the second one measures the "bias" introduced in the model by choosing too small a function class.

At this point, we would already like to point out that the space $\mathcal{H}$ is the means to balance the trade-off between estimation and approximation error. If we choose a very ''large space" $\mathcal{H}$, then the approximation term will become small (the Bayes classifier might even be contained in $\mathcal{H}$ or can be approximated closely by some element in $\mathcal{H}$). The estimation error, however,

will be rather large in this case: the space $\mathcal{H}$ will contain complex functions which will lead to overfitting. The opposite effect will happen if the function class $\mathcal{H}$ is very ''small".

# Empirical Risk Minimization

Given a loss function $\ell(\cdot, \cdot)$, the risk $R(h)$ is not computable as $P_{X,Y}$ is unknown. Thus we may not able to directly minimize $R(h)$ to obtain some predictor. Fortunately we are provided with the training data $\mathcal{D}_n = \{(x_1, y_1), ..., (x_n, y_n)\}$ which represents the underlying distribution $P_{X,Y}$.

Instead of minimizing $R(h) = E_{X,Y}[\ell(Y, h(X))]$, one may replace $P_{X,Y}$ by its empirical distribution and thus obtain the following minimization problem:

$$\hat{h}_n = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i))$$

which we call *empirical risk minimization* (ERM). Furthermore, we also define the *empirical risk* $\hat{R}_n(h)$ as

$$\hat{R}_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i))$$

Because under some conditions $\hat{R}_n(h) \to_P R(h)$ by the law of large numbers, the usage of ERM is at least partially justified.

ERM covers many popular methods and is widely used in practice. For example, if we take $\mathcal{H} = \{h(x) : h(x) = \theta^\intercal x, \theta \in \mathbb{R}^p\}$ and $\ell(y, p) = (y - p)^2$, then ERM becomes the well-known least squares estimation.

The celebrated maximum likelihood estimation (MLE) is also a special case of ERM where the loss function is taken to be the negative log-likelihood function. Example: in binary classification $(x_1, y_1), ..., (x_n, y_n)$ where $y_i \in \{-1, 1\}$ and $\mathcal{H} = \{h(x) : h(x) = \theta^\intercal x, \theta \in \mathbb{R}^p\}$, the logistic regression is computed by minimizing the logistic loss:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \theta^\intercal x_i))$$

which is equivalent to MLE.

# Overfitting

ERM works by minimizing the empirical risk $\hat{R}_n(h)$, while the goal of learning is to obtain a predictor with a small risk $R(h)$. Although under certain conditions the former will converge to the latter as $n \to \infty$, in practice we always have a finite sample and as a result, there might be a large discrepancy between those two targets, especially when $\mathcal{H}$ is large and $n$

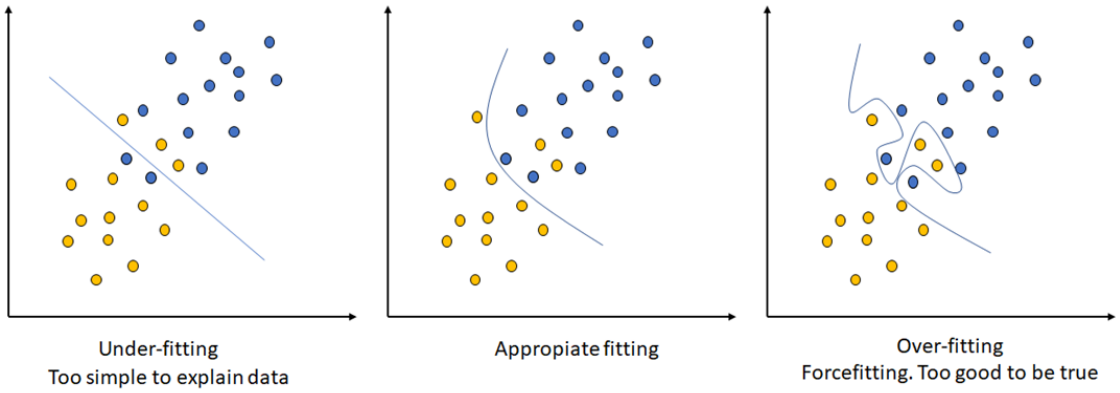


Figure 3: Overfitting

is small. Overfitting refers to the situation where we have a small empirical risk but still a relatively large true risk.

Learning is more about generalization than memorization.

# Statistical Learning Theory

Statistical Learning Theory provides probabilistic bounds on the distance between the empirical and true risk of any function (therefore including the minimizer of the empirical risk in a function space that can be used to control overfitting). The bounds involve the number of examples' and the capacity $C$ of the function space, a quantity measuring the "complexity" of the space. Appropriate capacity quantities are defined in the theory, the most popular one being the VC-dimension (Vapnik and Chervonenkis, 1971). The bounds have the following general form: with probability at least $\eta$

$$R(f) < \hat{R}_n(f) + \varphi\left(\sqrt{\frac{C}{m}}, \eta\right), \tag{1}$$

where $C$ is the capacity and $\varphi$ an increasing function of $C/m$ and $\eta$. Intuitively, if the capacity of the function space in which we perform empirical risk minimization is very large and the number of examples is small, then the distance between the empirical and expected risk can be large and overfitting is very likely to occur (Fig. 4).

Since the space $\mathcal{H}$ is usually very large (e.g. $\mathcal{H}$ could be the space of square integrable functions), one typically considers a smaller hypothesis space $\mathcal{H}$. Moreover, inequality (1) suggests an alternative method for achieving good generalization: instead of minimizing the empirical risk, find the best trade off between the empirical risk and the complexity of the hypothesis space measured by the second term in the r.h.s. of inequality (1).

Here we list two commonly used approaches:

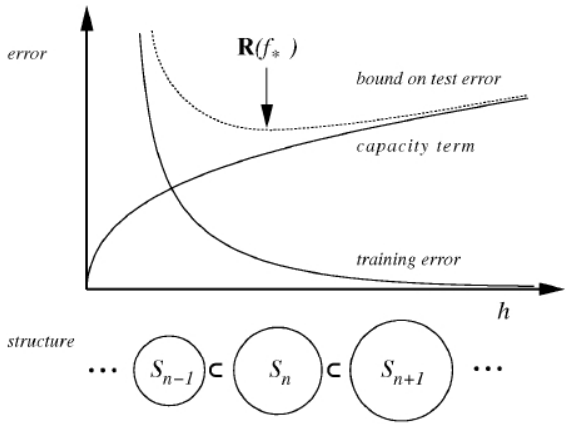


Figure 4: Model complexity

## Structural Risk Minimization

The idea of Structural Risk Minimization (SRM) is to define a nested sequence of hypothesis spaces $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_N$, where each hypothesis space $\mathcal{H}_n$ has finite capacity $c_n$ and larger than that of all previous sets, that is: $c_1 \leq c_2 \leq \cdots \leq c_N$. For example $\mathcal{H}_n$ could be the set of neural networks with one hidden layer with $n$ nodes. Using such a nested sequence of increasingly more complex hypothesis spaces, SRM consists of choosing the minimizer of the empirical risk in the space $\mathcal{H}_{n^*}$ for which the bound on the structural risk, as measured by the right hand side of inequality (1), is minimized.

## Penalized emprical risk minimization

An implicit way of working with nested function spaces is the principle of regularization. Instead of minimizing the empirical risk $\hat{R}_n(f)$ and then expressing the generalization ability of the resulting classifier $f_n$ using some capacity measure of the underlying function class $\mathcal{H}$, one can pursue a more direct approach: one directly minimizes a the so-called penalized risk

$$\frac{1}{m}\sum_{i=1}^{m} \ell(f(x_i), y_i)) + \lambda\Omega(f)$$

Here $\Omega(f)$ is the so-called regularizer (for instance $\Omega(f) = ||f||^2_{\mathcal{H}}$). This expression is kind of omnipresent. The inclusion of the $\Omega(f)$ factor above is central in Statistical Learning Theory. This form captures many algorithms: SVM, boosting, ridge regression, LASSO, ...

Within the Statistical Learning Theory, the regularizer $\Omega(f)$ is supposed to punish overly complex functions. For example, one often chooses a regularizer which punishes functions with large fluctuations, that is one chooses $\Omega(f)$ such that it is small for functions which vary slowly, and large for functions which fluctuate a lot.

The $\lambda$ in the definition of the regularized risk is a trade-off constant. It "negotiates" between the importance of $\hat{R}_n(f)$ and of $\Omega(f)$. If $\lambda$ is very large, we take the punishment induced

by $\Omega(f)$ very seriously, and might prefer functions with small $\Omega(f)$ even if they have a high empirical risk. On the other hand, if $\lambda$ is small, the influence of the punishment decreases, and we merely choose functions based on their empirical risks. The principle of regularization consists in choosing the classifier $f_n$ that minimizes the penalized risk.

In practice we often need to select $\mathcal{H}_n$ or $\lambda$ based on the training data to achieve a good balance between goodness-of-fit and model complexity.

Consider the following regression problem: let $\mathcal{H} = \{h(x) : h(x) = \theta^\intercal x, \forall \theta \in \mathbb{R}^p\}$ and we are trying to find an estimator $\hat{\theta}$ which minimizes the risk $E_{X,Y}(Y - \theta^\intercal X)^2$. For the first approach, we could define a sequence of increasing constants $0 \leq \eta_1 \leq \eta_2 \leq ... \leq \eta_k \leq ...$ and define $\mathcal{H}_k = \{h(x) : h(x) = \theta^\intercal x, \theta^\intercal \theta \leq \eta_k\}$.

For the second approach we define $\Omega(h) = \theta^\intercal \theta$. Then it is well-known from optimization that those two approaches become mathematically equivalent (i.e. for any $\eta_k$ there exists a $\lambda$ such that those two optimization problems have the same solution).

# Cross-validation

The obvious question is: how do we pick $\lambda$? This, of course, has been the subject of much theoretical study. What is commonly done in practice is this:

1. Divide the data into a training set and validation set.
2. Fit the model for a range of different $\lambda$s using only the training set.
3. For each of the models fit in step 2, check how well the resulting weights fit the validation data.
4. Output the weights that perform best on validation data.

One can also retrain on all the data using the $\lambda$ that did best in step 2 (see Fig. 5).

Validation set anticipates the test set (the unseen data).

## k-fold cross-validation

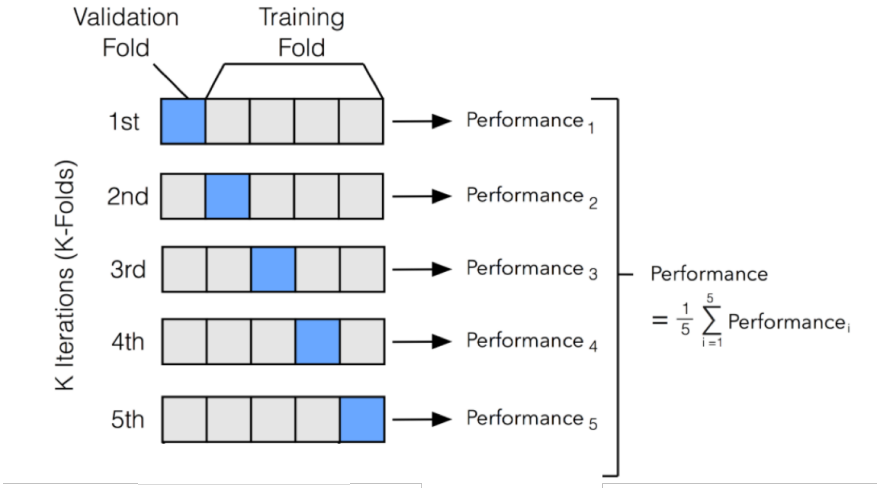


Figure 5: Hyperparameter tuning

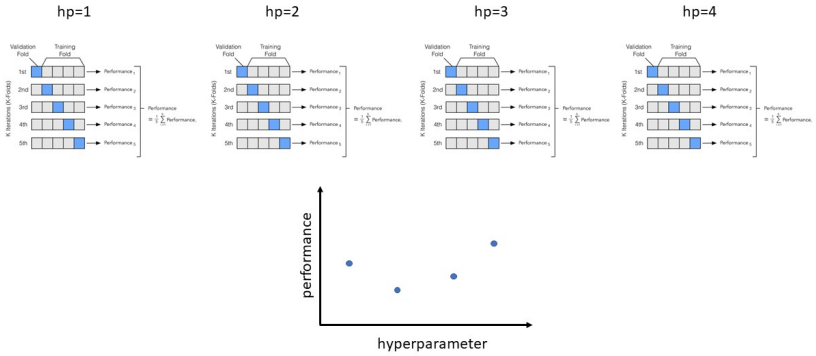


Figure 6: Hyperparameter tuning

# Bibliography

Introduction to Statistical Learning Theory. Olivier Bousquet, Stephane Boucheron, and Gabor Lugosi. Advanced Lectures on Machine Learning pp 169-207. 2004.

Statistical Learning Theory: Models, Concepts, and Results. Ulrike von Luxburg and Bernhard Scholkopf. 2008.