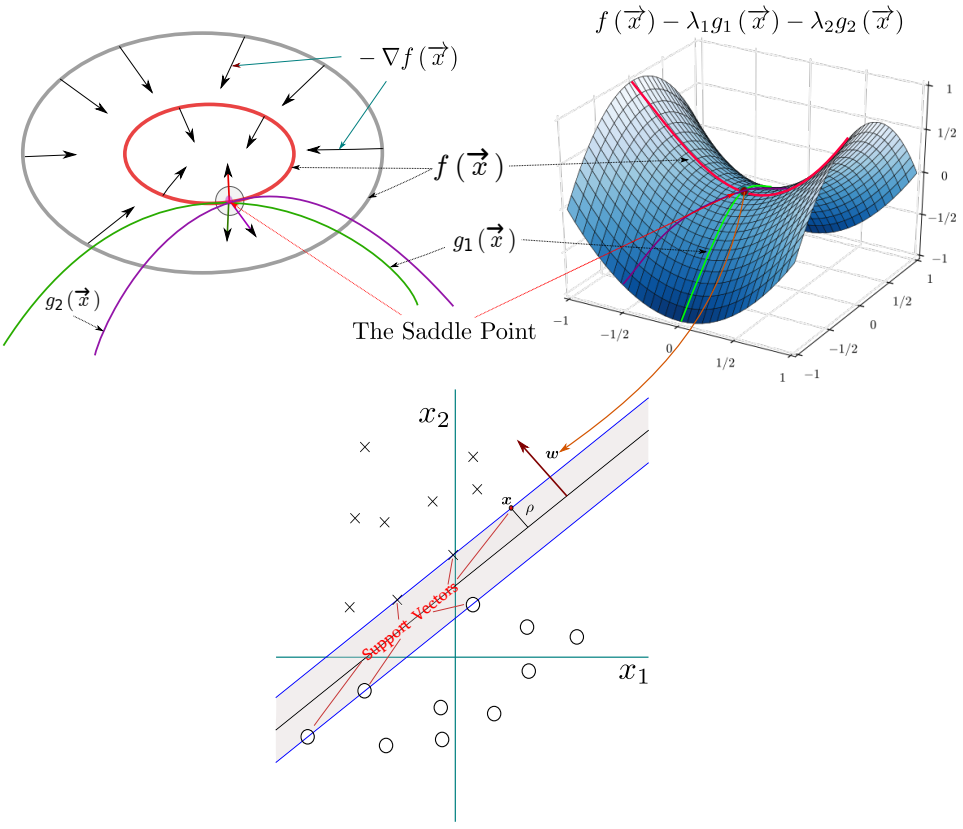
Introduction to Machine Learning
Notes
A Basic Introduction to Learning

Andres Mendez-Vazquez

June 6, 2018



Contents

List of Figures

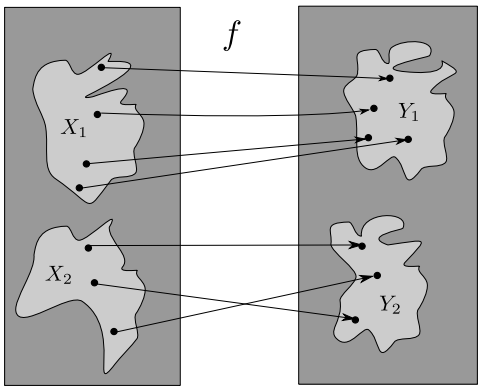


Figure 1: Mapping from the input space into the output space with two classes

1 Introduction

As in any other field of Science, it is necessary to obtain a basic understanding of the subject before getting into the main matter of the subject. In our particular case, the concept of "Learning" in Machine Learning [2, 3, 4, 5]. As we will see through our study of Machine Learning, we could give the following definition of "Learning."[6]

Definition 1.1. *Given that the information of an object has been summarized by d features comprised as a feature vector* $\boldsymbol{x} \in \mathbb{R}^d$*, and each of these objects has been labeled by elements in a set* $\{y_i \in \mathbb{R}\}$*. This allows to split the set of object into a series classes, as for example* $y_i \in \{-1, 1\}$*. Then, for example, the process of learning is the generation of a mapping* $f : \mathbb{R}^d \mapsto \{y_i\}$ *(Figure 1) such that the squared error estimation of the class label of a new sample is minimized (Equation 1).*

$$\min_{\widehat{f}} E_{\mathcal{X},\mathcal{Y}} \left[\left(\widehat{f}\left(\boldsymbol{x}\right) - y \right)^2 | \boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d, y \in \mathcal{Y} \subseteq \mathbb{R} \right] \tag{1}$$

Although this task looks quite simple, actually it can be quite complex! Therefore, a good starting point to this endeavors is to use our analytic geometry [7] from high school to try to obtain such mapping. Thus, if we use a geometric approach, and assuming two linearly separable classes, we could start by splitting the samples into two set of elements,

$$C_1 = \left\{ \boldsymbol{x}_i \in \mathbb{R}^d | y_i = 0 \right\}$$
$$C_2 = \left\{ \boldsymbol{x}_i \in \mathbb{R}^d | y_i = 1 \right\}$$

Then, we could try to obtain a geometric function allowing to split the space of inputs into two classes. For example, we could use (Figure 2) .

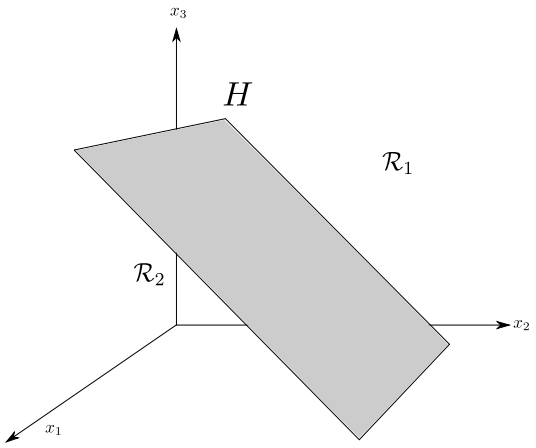


Figure 2: A Hyperplane splitting the space in

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0 \tag{2}$$

Question, How does this function split the space in $\mathbb{R}^3$? The answer comes from the equation (Equation 3).

$$\boldsymbol{w}^T\boldsymbol{x} = \|\boldsymbol{w}\|\,\|\boldsymbol{x}\|\cos\theta \tag{3}$$

Basically, θ represents the angle between vectors $\boldsymbol{w}$ and $\boldsymbol{x}$. Thus, we have the following rule:

1. If $0 \leq \theta \leq 90$ then $\boldsymbol{w}^T\boldsymbol{x} \geq 0$
2. If $\theta > 90$, then $\boldsymbol{w}^T\boldsymbol{x} < 0$

Therefore, $\boldsymbol{w}$ defines a positive subspace and a negative subspace (Figure 3) given the direction defined by the vector.

A question arises naturally, from this first approach, How do you obtain such function? We will see a first approach for this in the following section.

2 Loss Functions and Empirical Risk

2.1 Empirical Risk as Error Minimization

The first thing that you need to think about, while trying to obtain an estimation mapping between the input space $\mathcal{X}$ and the output space $\mathcal{Y}$, is our decision about how to relate the two variables, the vectors $\boldsymbol{x} \in \mathbb{R}^d$and the outputs $y \in \mathbb{R}$.

Example 2.1. We could try to relate the variations in prices at supermarkets with thSkycatch e variations in salary by assuming a growing economy [8]. Given such scenario, you could think a linear relation like (Equation 4).

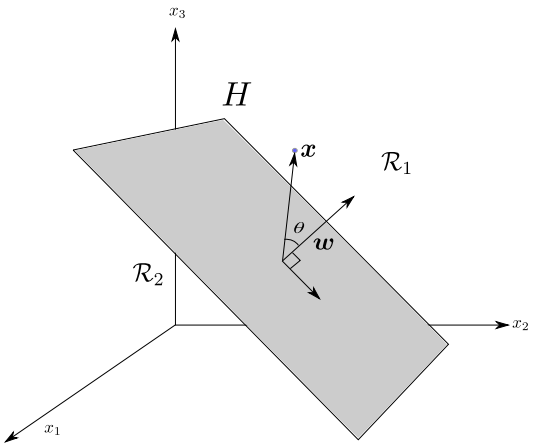


Figure 3: The positive subspace $\mathcal{R}_1$ and the negative subspace $\mathcal{R}_2$

$$\Delta P = w_0 + w_1 \Delta S + \epsilon \tag{4}$$

with ΔP and ΔS are the price and salary variations, w_0 and w_1 are unknown coefficients and $\epsilon \sim N(0, \sigma^2)$.

This example depicts one of the classic models in the literature, the linear models where a random source of noise is used to model the variability on the y_i being observed. Now, given the previous linear random relation, anybody will start to wonder how to minimize the noise or error (Equation 5),

$$\epsilon = \Delta P - (w_0 + w_1 \Delta S)\,. \tag{5}$$

Given that we want to minimize such error, we require to compose the previous functions with an specific one to obtain the desired results. For this, we introduce the idea of statistical risk as a functional that evaluates the expected value of a function that describe a measure of loss.

Definition 2.2. (Principle of Empirical Risk) [9] Given a sequence of data samples, $x_1, x_2, ..., x_N$ sampled iid from a distribution $P(x|\Theta)$, and an hypothesis function $h : X \mapsto Y$ that allows to map the samples x_i into a particular output y_i. A measure of the risk of missing the estimation, $h(x)$, is found by using a function, called loss function, measuring the difference between the desired output y_i and the estimation $h(x_i)$. Thus, the Empirical Risk is defined as the expected value of the loss function based in the joint distribution $P(x, y)$ (Equation 6).

$$R(h) = E_{X,Y}\left[\ell\left(h\left(x\right), y\right)\right] = \int_{X,Y} \ell\left(h\left(x\right), y\right) p\left(x, y\right) dxdy \tag{6}$$

Examples of such loss functions are (Equation 7):

$$
\begin{aligned}
h(w_0, w_1) &= [\Delta P - (w_0 + w_1 \Delta S)]^2 \text{ (Least Squared Error)}, \\
h(w_0, w_1) &= |\Delta P - (w_0 + w_1 \Delta S)| \text{ (Absolute Difference)}, \\
h(w_0, w_1) &= \frac{1}{1 + \exp\{-(w_0 + w_1 \Delta S)\}} \text{ (Sigmoidal)}.
\end{aligned} \tag{7}
$$

After looking at such functions loss functions, one could ask, Is there a way to obtain the optimal function h that minimizes their Empirical Risks? Is there a way to find the optimal loss function for a given data set $\{(x_i, y_i)\}$? These simple questions have driven the Machine Learning endeavor for the last 60 years.

2.2 Example of Risk Minimization, Least Squared Error

To exemplify the idea of minimizing the Risk Function [3], let us to select a convenient loss function:

$$
L(y, f(x)) = (y - f(x))^2 \text{ (Least Squared Error)} \tag{8}
$$

Here, we select a functional relation between the outputs $y's$ and inputs $x's$:

$$
Y_{noisy}(x) = f(y) + \epsilon \text{ with } \epsilon \sim N(0, 1) \tag{9}
$$

Then, we have the following Risk functional [9, 3]:

$$
R(f) = E(Y - f(X))^2 = \int_X \int_Y [y - f(x)]^2 p_{xy}(x, y)\, dxdy
$$

Now, we can condition the probability density function with respect to X:

$$
p(X, Y) = p(Y|X)\, p(X)
$$

Therefore, we have

$$
\begin{aligned}
\int_X \int_Y [y - f(x)]^2 p_{xy}(x, y)\, dxdy &= \int_X \int_Y [y - f(x)]^2 p_{y|x}(y|x)\, p_x(x)\, dxdy \\
&= \int_X \left[\int_Y [y - f(x)]^2 p_{y|x}(y|x)\, dy \right] dx \\
&= E_X \left[\int_Y [y - f(x)]^2 p_{y|x}(y|x)\, dy \right] \\
&= E_X E_{Y|X} \left[(Y - f(X))^2 |X \right]
\end{aligned}
$$

Thus, if we fix $X = \boldsymbol{x}$, we have that the Risk functional is with respect to such $\boldsymbol{x}$:

$$
R(f)_{X=\boldsymbol{x}} = E_{Y|X=\boldsymbol{x}} \left[(Y - f(\boldsymbol{x}))^2 |X = \boldsymbol{x} \right]
$$

This function can be optimized by realizing that

$$E_{Y|X=\boldsymbol{x}}\left[\left(Y-f\left(\boldsymbol{x}\right)\right)^{2}|X=\boldsymbol{x}\right]=E_{Y|X=\boldsymbol{x}}\left[\left(Y-\overline{Y}\right)^{2}|X=\boldsymbol{x}\right]+...\ =$$
$$E_{Y|X=\boldsymbol{x}}\left[\left(\overline{Y}-f\left(\boldsymbol{x}\right)\right)^{2}|X=\boldsymbol{x}\right]+...$$
$$2\left(\overline{Y}-f\left(\boldsymbol{x}\right)\right)E_{Y|X=\boldsymbol{x}}\left[\left(Y-\overline{Y}\right)|X=\boldsymbol{x}\right]$$

Then, you notice that

$$\begin{aligned}E_{Y|X=\boldsymbol{x}}\left[\left(Y-\overline{Y}\right)|X=\boldsymbol{x}\right]&=E_{Y|X=\boldsymbol{x}}\left[Y\right]-E_{Y|X=\boldsymbol{x}}\left[\frac{1}{N}\sum_{i=1}^{N}Y_{i}\right]\\&=\mu_{Y}-\frac{1}{N}\sum_{i=1}^{N}E_{Y|X=\boldsymbol{x}}\left[Y_{i}\right]\\&=\mu_{Y}-\frac{N\mu_{Y}}{N}\\&=0\end{aligned}$$

Thus, we have

$$E_{Y|X=\boldsymbol{x}}\left[\left(Y-f\left(\boldsymbol{x}\right)\right)^{2}|X=\boldsymbol{x}\right]=E_{Y|X=\boldsymbol{x}}\left[\left(Y-\overline{Y}\right)^{2}|X=\boldsymbol{x}\right]+...$$
$$E_{Y|X=\boldsymbol{x}}\left[\left(\overline{Y}-f\left(\boldsymbol{x}\right)\right)^{2}|X=\boldsymbol{x}\right]$$

Therefore, if we choose

$$f\left(X\right)=\overline{Y}\approx E_{Y}\left[Y|X=\boldsymbol{x}\right]$$

Thus, we finish with

$$E_{Y|X=\boldsymbol{x}}\left[\left(Y-f\left(\boldsymbol{x}\right)\right)^{2}|X=\boldsymbol{x}\right]=E_{Y|X=\boldsymbol{x}}\left[\left(Y-E_{Y}\left[Y|X=\boldsymbol{x}\right]\right)^{2}|X=\boldsymbol{x}\right]=\sigma_{Y}^{2}=1,$$

the variance of $Y\sim N\left(E_{Y}\left[Y|X=\boldsymbol{x}\right],1\right)$ when choosing $f\left(X\right)=E_{Y}\left[Y|X=\boldsymbol{x}\right]$ given $\epsilon\sim N\left(0,1\right)$. This means that the optimal estimator is the conditional mean for a point $X=\boldsymbol{x}$, when measured by the expected squared error.

3 Two Methods of Learning

Therefore, given the Empirical Risk, we can think on the process of "learning" as the process of finding an optimal function h that minimizes such risk under an specific loss function/learning model. Clearly, this process does not take in account an important issue in the human process of learning:

- How a human being uses few samples to generalize a concept?

For example, once a child learns the concept of "cat," she/he will be able to recognize almost every cat over the planet. Although, the concept of generalization exists in Machine Learning as part of the Bias-Variance problem [2, 3, 5], little progress has been done toward a sparse inductive generalization. Nevertheless,

there have been claims in the Deep Neural Network field of new generalization powers [10] by those devices. However, those powers require massive amounts of data, unlikely as the human child learning something as not so simple as a "cat."

After the previous digression, it is time to go back to our main topic, the process of "Learning" in Machine Learning. For this, we will take a look at two different methods of learning that could be seen as the extremes of a line where the Machine Learning methods live.

3.1 Least Squared Error

Here, we have the following situation about the data samples of a given classification problem:

- A sequence of samples with their respective labels, $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_N, y_N)\}$
- Here, each $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ with $i = 1, 2, ..., N$.

Then, it is possible to use the squared loss function, under a function g estimating the outputs $y_i's$:

$$\ell\left(g\left(\boldsymbol{x}\right), y\right) = \left(g\left(\boldsymbol{x}\right) - y\right)^2 \tag{10}$$

Thus, we have:

$$R\left(g\right) = \int_{X,Y} \ell\left(g\left(\boldsymbol{x}\right), y\right) p\left(\boldsymbol{x}, y\right) d\boldsymbol{x}dy = \sum_{i=1}^{N} \left(g\left(\boldsymbol{x}_i\right) - y_i\right)^2 p\left(\boldsymbol{x}_i, y_i\right). \tag{11}$$

Now, if assume two classes at the data i.e. $y_i \in \{l_1, l_2 | l_i \in \mathbb{R}\}$ such that $|C_{l_1}| = |C_{l_2}|$

$$p\left(\boldsymbol{x}_i, y_i\right) = \frac{|C_{l_1}|}{N} \approx \frac{1}{2} \text{ if } \boldsymbol{x}_i \in C_{l_1},$$
$$p\left(\boldsymbol{x}_i, y_i\right) = \frac{|C_{l_2}|}{N} \approx \frac{1}{2} \text{ if } \boldsymbol{x}_i \in C_{l_2}.$$

Therefore,

$$R\left(g\right) = \sum_{i=1}^{N} \left(g\left(\boldsymbol{x}_i\right) - y_i\right)^2 p\left(\boldsymbol{x}_i, y_i\right) = \frac{1}{2} \sum_{i=1}^{N} \left(g\left(\boldsymbol{x}_i\right) - y_i\right)^2.$$

Now, in the specific case of a linear model like $g\left(\boldsymbol{x}, \boldsymbol{w}\right) = w_0 + \boldsymbol{w}^T\boldsymbol{x}$, it is possible to relate the output through a random error ϵ:

$$y = g\left(\boldsymbol{x}, \boldsymbol{w}\right) + \epsilon, \tag{12}$$

thus, the error ϵ can be seen graphically as (Figure 4).

Here, a simplifying assumption is that the error comes from a Gaussian Distribution with mean 0 and variance σ^2, $N\left(0, \sigma^2\right)$. Therefore, the Risk Functional can be rewritten as

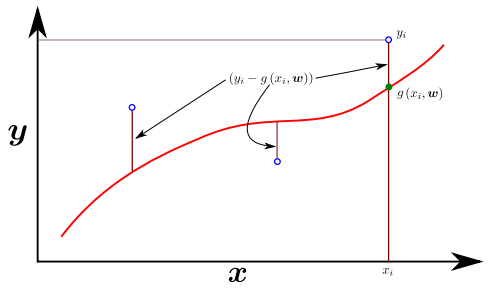


Figure 4: ϵ as the difference $\epsilon = y - g(\boldsymbol{x}, \boldsymbol{w})$

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\epsilon^2 = \frac{1}{2}\sum_{i=1}^{N}\left(w_0 + \boldsymbol{w}^T\boldsymbol{x} - y_i\right)^2 \tag{13}$$

Finally, using our Linear Algebra knowledge, we can rewrite this Risk Functional as

$$E(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) \tag{14}$$

with

$$\boldsymbol{X} = \begin{pmatrix} 1 & (\boldsymbol{x}_1)_1 & \cdots & (\boldsymbol{x}_1)_j & \cdots & (\boldsymbol{x}_1)_d \\ \vdots & & & \vdots & & \vdots \\ 1 & (\boldsymbol{x}_i)_1 & & (\boldsymbol{x}_i)_j & & (\boldsymbol{x}_i)_d \\ \vdots & & & \vdots & & \vdots \\ 1 & (\boldsymbol{x}_N)_1 & \cdots & (\boldsymbol{x}_N)_j & \cdots & (\boldsymbol{x}_N)_d \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \tag{15}$$

The final solution, the estimation $\widehat{\boldsymbol{w}}$, is

$$\widehat{\boldsymbol{w}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} \tag{16}$$

A question arises, how do we use this $\widehat{\boldsymbol{w}}$ to obtain the labels of our classes? We define a threshold saying $(l_1+l_2)/2$ and we define the rule of classification

$$\widehat{G}_{LSE}(\boldsymbol{x}) = \begin{cases} \text{Class 1} & \text{if } \widehat{\boldsymbol{w}}^T\boldsymbol{x} > (l_1+l_2)/2 \\ \text{Class2} & \text{if } \widehat{\boldsymbol{w}}^T\boldsymbol{x} \leq (l_1+l_2)/2 \end{cases}$$

Thus, we have that the hyperplane defined by $\widehat{\boldsymbol{w}}$ is a decision boundary that allows to split the space $\mathbb{R}^{d+1}$ into two classes (Figure 5). As you can see, decision boundary is splitting the space into two sub-planes where, as you can see, you can still have miss-classifications when estimating labels for the data samples.

This represent one of the most popular models in Statistics [11]. Additionally, it is one of the basic models of Learning in Machine Learning.

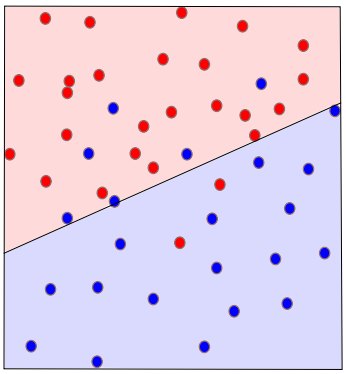


Figure 5: Decision Boundary $\left\{\boldsymbol{x}|\widehat{\boldsymbol{w}}^T\boldsymbol{x} = {}^{(l_1+l_2)}/_2\right\}$

3.2 Nearest Neighborhood

The other basic model in Machine Learning is the well known k-Nearest Neighbor (k-NN) model. This model can be though as an answer to the problem of increasing complexity of estimation of h, when the number of dimensions increases beyond a certain level. The rule of classification is quite simple:

$$\widehat{y}\left(\boldsymbol{x}\right) = \frac{1}{k}\sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i$$

where $N_K\left(\boldsymbol{x}\right)$ is the neighborhood of $\boldsymbol{x}$ defined by the k closest point $\boldsymbol{x}_i$ [1]. Here, an important questions is: What is the metric used to establish the concept of "near"? For example, we have the concept of near in documents by using the metric:

$$J\left(D_1, D_2\right) = \frac{|D_1 \cap D_2|}{|D_1| + |D_2| - |D_1 \cap D_2|} \text{ (Jaccard Index)}$$

when D_i is the vector that registers if a word appears or not in the document. Another example comes from the idea of using the sample mean and the sample covariance of Data Matrix (Equation 15).

$$\overline{\boldsymbol{X}} = \frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix} x_{i1} & x_{i2} & \cdots & x_{ip} \end{pmatrix}^T \text{ (Sample Mean)},$$

$$C_{\boldsymbol{X}} = \frac{1}{N-1}\left[\boldsymbol{X} - \overline{\boldsymbol{X}}\right]^T\left[X - \overline{\boldsymbol{X}}\right] \text{ (Sample Covarance)}.$$

This allows to define the Mahalanobis Distance (Figure 6), and as in the previous Least Squared Error, we can define a new rule of classification based

[1] Here, an efficient data structure to find the elements of $N_K\left(\boldsymbol{x}\right)$ are know as the $K-d$ trees [12]. This is necessary, if we want to have efficient Learning algorithms for Large Data sets.

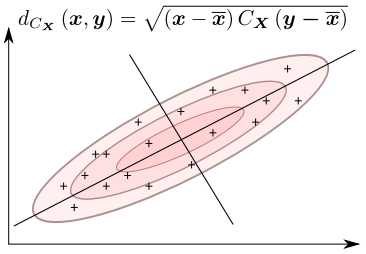


Figure 6: Mahalanobis as a distance measuring the maximum change in each feature.

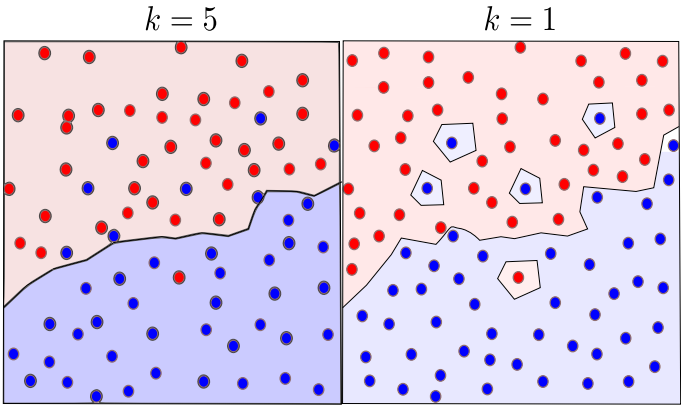


Figure 7: The Decision Boundary $k = 5$ and $k = 1$. Here k=1 correspond to a Voronoi diagram [1].

on:

$$\widehat{G}_{k\text{-NN}}(\boldsymbol{x}) = \begin{cases} \text{Class 1} & \text{if } \frac{1}{k}\sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i > {}^{(l_1+l_2)}/_2 \\ \text{Class 2} & \text{if } \frac{1}{k}\sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i \leq {}^{(l_1+l_2)}/_2 \end{cases},$$

where different values for k can produce different boundary decisions (Figure 7).

3.3 Machine Learning Methods as combination and improvements of LSE and k-NN

Something of great interest, as people looking for Patterns in the mathematical structures, mathematical models and data, is the realization that many methods in Machine Learning are related in many ways. In this regards, Hastie et al. [3] have identified two main algorithms for Learning, the Least Squared Error (Section 3.1) and k-Nearest Neighbor (Section 3.2), that can bee seen as the two main basis for the algorithms in Machine Learning.

For example, in the case of Support Vector Machines [13], when looking at its dual solution by Vapnik [4]:

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i \tag{17}$$

It is know that the elements α_i are different from zero when its associated vector is a support vector. Using an arbitrary kernel in (Equation 17), it is possible to rewrite the equation as:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(\boldsymbol{x}_j, \boldsymbol{x}_i). \tag{18}$$

Thus, if we assume

$$\ell(\alpha_i, \alpha_j, d_i, d_j) = I(\alpha_i > t, \alpha_j > t)\, sign(d_i d_j),$$

and $p = 1$, we have

$$\sum_{i=1}^{N} \left(\alpha_i - \frac{1}{2} \sum_{j=1}^{N} \ell(\alpha_i, \alpha_j, d_i, d_j) K(\boldsymbol{x}_j, \boldsymbol{x}_i) \right)^p \tag{19}$$

Which looks like a Linear Regression with α_i as the desired output and

$$f(\boldsymbol{x}_i) = \frac{1}{2} \sum_{j=1}^{N} \ell(\alpha_i, \alpha_j, d_i, d_j) K(\boldsymbol{x}_j, \boldsymbol{x}_i)$$

as the estimator of sample $\boldsymbol{x}_i$ based on their projections by $K(\boldsymbol{x}_j, *)$.

This is one example of many where algorithms in Machine Learning can be seen as an extension of LSE or k-NN. These could be seen as a little bit limited, but it is useful when trying to analyze and categorize a Machine Learning Algorithm when doing model selection [14].

4 Supervised Learning as Function Approximation

4.1 Regression as Controlled Overfitting

In an exemplifying case, we observe a real-valued input variable $x \in \mathbb{R}$, and we are looking to predict the value of a real valued variable $y \in \mathbb{R}$. Thus, we have a vector of tuples:

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \,|i = 1, 2, ..., N\} \tag{20}$$

for the N samples with their respective outputs. For example, we have the function:

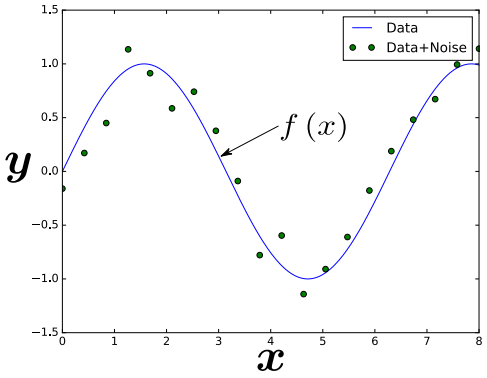


Figure 8: The data and the random observations.

$$g\,(x) = f\,(x) + \alpha\epsilon \tag{21}$$

with $\epsilon \sim U\,(0,1)$, $\alpha > 0$, $f\,(x) = \sin\{x\}$ and $x \in \mathbb{R}$ (Figure 8).

Thus, our goal is to exploit this training set to reduce the noise introduced by the system and our observations (Kalman [15]). Basically, we want make predictions of the value $\widehat{y} = g\,(x)$ given a new value $\widehat{x}$. For this is possible to propose the following polynomial:

$$y = g\,(x, \boldsymbol{w}) = \sum_{i=0}^{d} w_i x^i,$$

which can be seen as a generation of new features in a polynomial way (Polynomial Kernels [13]). These functions are linear at the parameter $\boldsymbol{w}$ and are called linear models. The question, How do we fit the model to our data? Here, we can use the sum of squared errors:

$$R\,(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\left[g\,(x_i, \boldsymbol{w}) - y_i\right]^2 \tag{22}$$

Later in the section of linear models, we will look at ways to get the the canonical solution (Equation 16) for the problem. However, in general, the goal is to to obtain a useful approximation (fitting) to $g(\boldsymbol{x}, \boldsymbol{w})$ for all $\boldsymbol{x}$ in some region of $\mathbb{R}^d$, given the representations in $\mathcal{D}$. This is an advantage because we can use all the tools generated in the last 200 years for function approximation. Making the task of Approximation/Supervised Learning as **Controlled Overfitting** where in each model a series of parameters are approximated to obtain the desired fitting.

Examples of different approximations that relate the labels and samples in $\mathcal{D}$ are:

1. The linear model $f(x) = \boldsymbol{x}^T \boldsymbol{w}$ with $\theta = \boldsymbol{w}$.
2. The linear basis expansion

$$f_\theta(\boldsymbol{x}) = \sum_{k=1}^{K} h_k(\boldsymbol{x}) \theta_k$$

   with $h_k(\boldsymbol{x}) = x_1^2$ or $h_k(\boldsymbol{x}) = \frac{1}{1+\exp\{-\boldsymbol{x}^T \theta_k\}}$ and many other ones.
3. The residual sum of squared errors

$$RSS(f_{\boldsymbol{w}}, \boldsymbol{x}_0) = \sum_{i=1}^{N} K_\lambda(\boldsymbol{x}_i, \boldsymbol{x}_0) (y_i - f_{\boldsymbol{w}}(\boldsymbol{x}_i))^2$$

Although these are some examples that are quite important in the process of approximation, one should ask the following question, When do we have fittings that are not good at all for the data space? In the following section, we try to answer this question.

4.2 Extreme Fitting, Bias-Variance Problem

For the following development [16, 17], we will take a look at the underlying probability of the samples $\boldsymbol{x}'s$ by having the following assumptions:

1. There is an underlying distribution for the data $\boldsymbol{x}_i \sim p(\boldsymbol{x}|\Theta)$ [18]. Here, the notation $p(\boldsymbol{x}|\Theta)$ represent the random dependencies of the samples of the underlaying distribution p which is parametrized by Θ.
2. Some Learning Algorithm has been able to find an estimation function $g(\boldsymbol{x}|\mathcal{D})$ where the notation $|\mathcal{D}$ represents the dependency of the function g to the training data $\mathcal{D}$.
3. As in the Least Squared Error, we have seen that the optimal solution for the regression $E_{\mathcal{D}}[y|\boldsymbol{x}]$ where the subscript $\mathcal{D}$ means that the expected value depends on the distribution $P(X, Y)$.
4. The noise added by the mapping $y = g(x) + \epsilon$ has a distribution $\epsilon \sim N(0, \sigma_\epsilon^2)$.

Then, from our probability theory the $Var(X) = E\left[(X - \mu)^2\right]$ which can be used to measure the error (Equation 23) of the function $g(\boldsymbol{x}|\mathcal{D})$ with respect to such expected label of a given sample, $E[y|\boldsymbol{x}]$.

$$Var_{\mathcal{D}}(g(\boldsymbol{x}|\mathcal{D})) = E_{\mathcal{D}}\left(\left(g(\boldsymbol{x}|\mathcal{D}) - E_{\mathcal{D}\|\text{-NN}}[y|\boldsymbol{x}]\right)^2\right) \tag{23}$$

Clearly as part of measuring the error, we can use the expected value of the trained function/machine $g(\boldsymbol{x}|\mathcal{D})$ (Equation 24).

$$E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] \tag{24}$$

the expected output of the machine. Thus, by using a little of algebra, we can rewrite (Equation 23) as

$$\begin{aligned} Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) &= E_{\mathcal{D}}\left[\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)^2\right] \\ &= E_{\mathcal{D}}\left[\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] + E\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)^2\right] \\ &= E_{\mathcal{D}}\left[\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2 + ...\right. \\ &...2\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)\left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right) + ... \\ &\left.... \left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)^2\right]. \end{aligned}$$

We can split the terms using the linearity of the expected value:

$$\begin{aligned} Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) &= E_{\mathcal{D}}\left[\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right] + ... \\ &...2E_{\mathcal{D}}\left[\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)\left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)\right] + ... \\ &...E_{\mathcal{D}}\left[\left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)^2\right]. \end{aligned}$$

The central term is one of interest

$$E_{\mathcal{D}}\left[\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)\left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)\right] = *$$

and by using the linearity of the expected value:

$$\begin{aligned} * &= E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E^2_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - ...\right. \\ &\left.... \; g\left(\boldsymbol{x}|\mathcal{D}\right)E_{\mathcal{D}}\left[y|\boldsymbol{x}\right] + E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right] \\ &= E^2_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E^2_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - ... \\ &...E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]E_{\mathcal{D}}\left[y|\boldsymbol{x}\right] + E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]E_{\mathcal{D}}\left[y|\boldsymbol{x}\right] \\ &= 0. \end{aligned}$$

Finally,

$$Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) = \underbrace{E_{\mathcal{D}}\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE} + \underbrace{\left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS} \tag{25}$$

Using a little bit of statistics, we can recognize the terms on the $Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right)$:

1. $E_{\mathcal{D}}\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)$ is **the variance** of output of the learned function $g\left(\boldsymbol{x}|\mathcal{D}\right)$.
2. $\left(E_{\mathcal{D}}\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E_{\mathcal{D}}\left[y|\boldsymbol{x}\right]\right)^2$ is a measure on how the expected value of the learned function differs from the expected output, **the bias**.

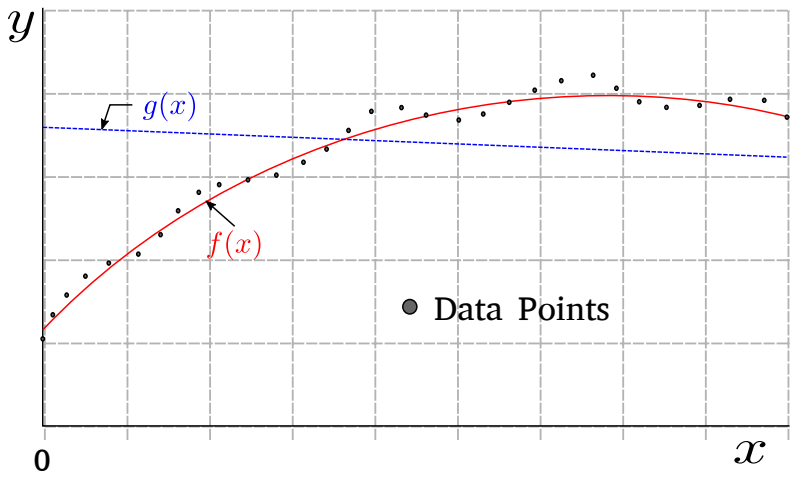


Figure 9: The function used to estimate the output y is highly biased given the mapping.

Thus, we realized that the variance of $g(\boldsymbol{x}|\mathcal{D})$ ranges between two terms: From variance of the output of the learned function to the bias on the expected output of $g(\boldsymbol{x}|\mathcal{D})$. It is easy to see this, when X and Y are random variables, then the $g(\boldsymbol{x}|\mathcal{D})$ is also a random variable. Making of the entire process of learning a random process where we want to avoid extreme cases of fitting.

4.2.1 "Extreme" Cases of Fitting

Case High Bias Imagine that $\mathcal{D} \subset [x_1, x_2]$ in which x lies. For example, we can choose $x_i = x_1 + \frac{x_2 - x_1}{N-1}(i-1)$ with $i = 1, 2, ..., N$. Further, we can choose the estimate of $f(x)$, $g(x|\mathcal{D})$, to be independent of $\mathcal{D}$. Thus, we can use any function, we can imagine. For example, we could have:

$$g(x) = w_0 + w_1 x$$

Thus, we have the following situation (Figure 9). There, the estimated output is highly biased and far away from the expected outputs of the data samples.
Mathematically, given that $g(x)$ is fixed, we have:

$$E_{\mathcal{D}}[g(x|\mathcal{D})] = g(x|\mathcal{D}) \equiv g(x),$$

with

$$Var_{\mathcal{D}}[g(x|\mathcal{D})] = 0.$$

On the other hand, because $g(x)$ was chosen arbitrarily the expected bias must be large (Equation 26).

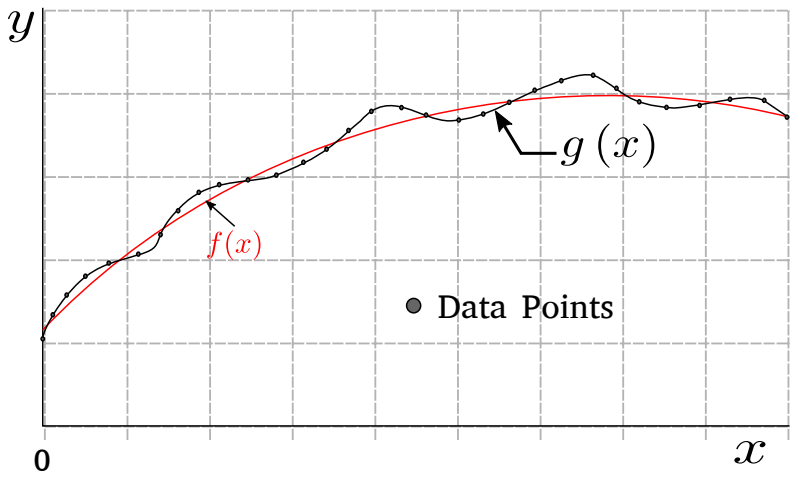


Figure 10: The high variance case of fitting

$$\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS} \tag{26}$$

Clearly, this limited fitting is not what we want, but what could happen if we avoid this high bias and accept a high variance?

Case High Variance In the other hand, $g\left(x\right)$ corresponds to a polynomial of high degree so it can pass through each training point in $\mathcal{D}$ (Figure 10).

Now, due to the zero mean of the noise source, we have that:

$$E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]=f\left(x\right)=E\left[y|x\right] \text{ for any } x=x_i \tag{27}$$

Thus, at the training points the bias is zero in (Equation 25). However, the variance increases

$$\begin{aligned} E_D\left[\left(g_1\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g_1\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right] &= E_D\left[\left(f\left(x\right)+\epsilon-f\left(x\right)\right)^2\right] \\ &= \sigma_\epsilon^2, \text{ for } x=x_i, i=1,2,...,N \end{aligned}$$

In other words, the bias becomes zero (or approximately zero) but the variance is now equal to the variance of the noise source. Thus, at least in linear models, the Learning procedure needs to make a balance between the Bias and the Variance, this is called the **Bias-Variance trade-off**.

Nevertheless, some observations about this Bias-Variance are due. First, everything that has been said so far applies to both the regression and the classification tasks. However, the Mean Squared Error is not the best way to measure the power of a classifier. This is a classifier that sends everything far away of the hyperplane i.e. away from the values $(+)-1$.

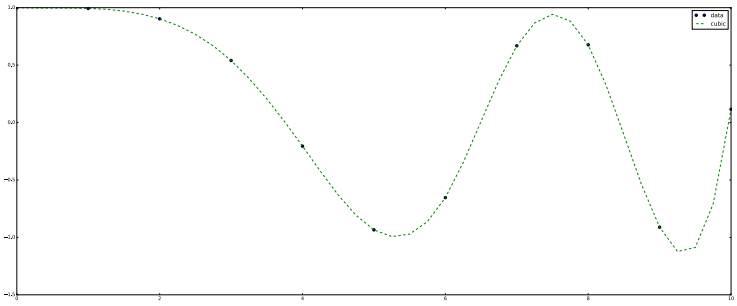


Figure 11: The cubic smooth spline

5 Some Classes of Estimators

Here, Hastie et al. [3] has observed that many of the learning methods fall into different classes of estimators depending on the restrictions imposed by the models. These are only a few ones, but the list is quite larger given the works done in the field, and the need of new algorithms.

5.1 Roughness Penalty

These methods are based on the idea of penalizing the parametrized mapping f to force a regularization over those parameters (Equation 28).

$$R\left(f;\lambda\right) = RSS\left(f\right) + \lambda J\left(f\right) \tag{28}$$

A classic rule to force the regularization is to use a convenient large functional $J\left(\lambda\right)$ and a large enough λ to dampen f with high variations over small regions of the input space. For example, the cubic smoothing spline (Figure 11) for one-dimensional input is the solution to the penalized least-squared criterion:

$$R\left(f;\lambda\right) = \sum_{i=1}^{N}\left(y_i - f\left(x_i\right)\right)^2 + \lambda\int\left[\frac{d^2 f\left(x\right)}{dx^2}\right]^2 dx \tag{29}$$

Here, the Roughness penalty and the λ controls the high variance of the function f that can be thought as the change in the curvature of the function. Therefore, the penalty works as a way to dampen the high variations in the function f. For example:

1. Linear functions in x, f, appear when $\lambda \to \infty$.
2. Any function f will stay the same when $\lambda = 0$.

Therefore, penalty or regularization functions express our prior knowledge about the smoothness of the function we are seeking. Thus, it is possible to cast

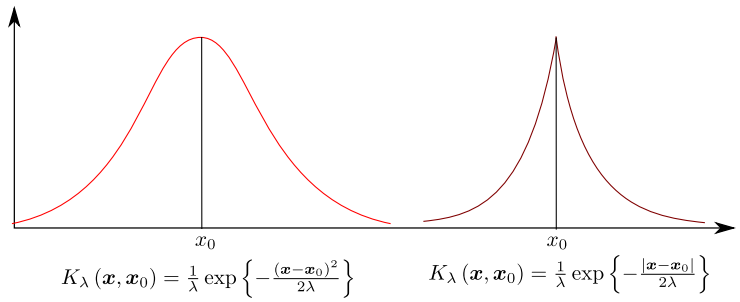


Figure 12: Two types of Kernels

this penalty functions in terms of a Bayesian Framework [19]. The penalty J is the prior distribution, $\sum_{i=1}^{N}\left(y_i - f\left(x_i\right)\right)^2$ correspond to the likelihood and $PRSS\left(f;\lambda\right)$ is the posterior distribution. Thus, minimizing $R\left(f;\lambda\right)$ amounts to find the mode in the posterior distribution given that you have a single mode distribution.

5.2 Kernel Methods and Local Regression

You can think on these methods as estimations of regression functions or conditional expectations by specifying:

1. The properties of the local neighborhood.
2. The class of regular functions fitted locally.

For this, they use kernels as

$$K_\lambda\left(\boldsymbol{x},\boldsymbol{x}_0\right)=\frac{1}{\lambda}\exp\left\{-\frac{\left\|\boldsymbol{x}-\boldsymbol{x}_0\right\|^2}{2\lambda}\right\}.$$

For example, we have the following kernels (Figure 12). And as in Linear Regression, we can define a way of doing estimation by:

$$R\left(f_{\boldsymbol{w}},\boldsymbol{x}_0\right)=\sum_{i=1}^{N}K_\lambda\left(\boldsymbol{x}_i,\boldsymbol{x}_0\right)\left(y_i-f_{\boldsymbol{w}}\left(\boldsymbol{x}_i\right)\right)^2, \tag{30}$$

where $f_{\boldsymbol{w}}$ could be defined as

1. $f_{\boldsymbol{w}}\left(\boldsymbol{x}\right)=w_0$ the constant function (Nadaraya–Watson Estimate).
2. $f_{\boldsymbol{w}}\left(\boldsymbol{x}\right)=\sum_{i=0}^{d}x_i w_i$ the classic local linear regression models.

In another example the Nearest-Neighbor methods, the following kernel can be defined as:

$$K_k\left(\boldsymbol{x},\boldsymbol{x}_0\right)=I\left[\left\|\boldsymbol{x}-\boldsymbol{x}_0\right\|\leq\left\|\boldsymbol{x}_{(i)}-\boldsymbol{x}_0\right\|\left|i=1,2,\ldots,k\right.\right]$$

where

1. $\boldsymbol{x}_{(i)}$ is the training observation ranked i^{th} in distance from $\boldsymbol{x}_0$.
2. $I(S)$ is the indicator of the set S.

5.3 Basis Functions and Dictionary Methods

These are a more wide variety of flexible models defined by the use of linear and polynomial expansions (Equation 31).

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{m=1}^{M} w_m h_m(\boldsymbol{x}), \tag{31}$$

where

- h_m is a function on $\boldsymbol{x}$.
- with the linear term w_m acting on the function h_m.

Other examples are:

1. Radial basis functions

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{m=1}^{M} w_m K_{\lambda_m}(\mu_m, \boldsymbol{x}) \text{ with } K_\lambda(\mu, \boldsymbol{x}) = \exp\left\{-\frac{\|\boldsymbol{x}-\mu\|^2}{2\lambda}\right\}$$

2. A single-layer feed-forward neural network

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{m=1}^{M} w_m S\left(\alpha_m^T \boldsymbol{x} + b_m\right) \text{ with } S(y) = \frac{1}{1+\exp\{-y\}}$$

6 Conclusion

Thus, Machine Learning is a quite wide and vast field with many opportunities that will keep growing as many industries take advantage of this sui generis way of Learning. Nevertheless, this takes time, effort and humility to master because there is no silver bullet that can help to avoid some of the fundamental problems in the field. Only creativity, ingenuity and effort can take the practitioner to calm waters in the vast ocean of Machine Learning.

References

[1] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, "Computational geometry," in *Computational geometry*, pp. 1–17, Springer, 2000.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics, Springer New York, 2009.

[4] S. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2008.

[5] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 1st ed., 2015.

[6] S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh, "Learning pattern classification-a survey," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2178–2206, 1998.

[7] R. Silverman, *Modern Calculus and Analytic Geometry*. Dover Books on Mathematics, Dover Publications, 1969.

[8] C. Robert, *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[9] V. Vapnik, *Statistical learning theory*. Wiley, 1998.

[10] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[11] A. Rencher, *Linear models in statistics*. Wiley series in probability and statistics, Wiley, 2000.

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, Sept. 1975.

[13] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[14] G. Claeskens and N. L. Hjort, *Model Selection and Model Averaging*. No. 9780521852258 in Cambridge Books, Cambridge University Press, April 2008.

[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[16] P. Domingos, "A unified bias-variance decomposition," in *Proceedings of 17th International Conference on Machine Learning*, pp. 231–238, 2000.

[17] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[18] R. Ash, *Basic Probability Theory*. Dover Books on Mathematics Series, Dover Publications, Incorporated, 2012.

[19] M. A. T. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 1150–1159, Sept. 2003.