

# Class Notes (experimental)

Jonathan Rosenblatt

April 10, 2015

## Contents

<b>1</b>	<b>Estimation</b>	<b>3</b>
1.1	Moment matching . . . . .	3
1.2	Quantile matching . . . . .	3
1.3	Maximum Likelihood . . . . .	4
1.4	M-Estimation and Empirical Risk Minimization . . . . .	6
1.5	Notes . . . . .	7
<b>2</b>	<b>From Estimation to Supervised Learning</b>	<b>7</b>
2.1	Empirical Risk Minimization (ERM) and Inductive Bias . . . .	7
2.2	Ordinary Least Squares (OLS) . . . . .	8
2.3	Ridge Regression . . . . .	9
2.4	LASSO . . . . .	9
2.5	Logistic Regression . . . . .	10
2.6	Regression Classifier . . . . .	11
2.7	Linear Support Vector Machines (SVM) . . . . .	11
2.8	Generalized Additive Models (GAMs) . . . . .	13
2.9	Projection Pursuit Regression (PPR) . . . . .	13
2.10	Neural Networks (NNETs) . . . . .	14
2.10.1	Single Hidden Layer . . . . .	14
2.11	Classification and Regression Trees (CARTs) . . . . .	15
2.12	Smoothing Splines . . . . .	16
<b>3</b>	<b>Non ERM Supervised Learning</b>	<b>17</b>
3.1	k-Nearest Neighbour (KNN) . . . . .	17
3.2	Kernel Smoothing . . . . .	18
3.3	Local Likelihood and Local ERM . . . . .	19
3.4	Local Regression (LOESS) . . . . .	19

<b>4</b>	<b>Statistical Decision Theory</b>	<b>19</b>
4.1	Train, Validate, Test . . . . .	21
4.2	Unbiased Estimators of the Risk . . . . .	22
4.3	Jackknifing . . . . .	22
4.4	Cross Validation . . . . .	23
4.5	Bootstrapping . . . . .	23
<b>5</b>	<b>Unsupervised Learning</b>	<b>24</b>
5.1	Density Estimation . . . . .	24
5.2	Agglomerative Clustering . . . . .	24
5.3	K Means Clustering . . . . .	25
5.4	Mixtures . . . . .	25
5.4.1	Association Rules . . . . .	25
<b>6</b>	<b>Dimensionality Reduction</b>	<b>25</b>
6.1	In Supervised Learning . . . . .	25
6.1.1	Model Selection . . . . .	25
6.1.2	LASSO . . . . .	25
6.2	In unsupervised Learning . . . . .	25
6.2.1	PCA . . . . .	25
6.2.2	Multidimensional Scaling (MDS) . . . . .	25
6.2.3	Self Organizing Maps . . . . .	25
6.2.4	Principal Curves . . . . .	26
6.2.5	ICA . . . . .	26
6.2.6	kPCA . . . . .	26
6.2.7	ISomap . . . . .	26
6.2.8	LLE . . . . .	26
<b>7</b>	<b>Latent Space Models</b>	<b>26</b>
7.1	Mixtures . . . . .	26
7.2	Regression Switching . . . . .	26
7.3	Factor Analysis . . . . .	26
7.4	Hidden Markov . . . . .	26
7.5	Collaborative Filtering . . . . .	27

This text draws from Hastie et al. [2003] and Shalev-Shwartz and Ben-David [2014]. The former is freely available online. For a softer introduction, with more hands-on examples, see James et al. [2013]. All books are very well written and strongly recommended.

# 1 Estimation

In this section, we present several estimation principles. Their properties are not discussed, as the section is merely a reminder and a preparation for Section 2. These concepts and examples can be found in many introductory books to statistics. I particularly recommend [Wasserman, 2004].

## 1.1 Moment matching

The fundamental idea: match empirical moments to theoretical. I.e., estimate

$$E[g(X)]$$

by

$$\mathbb{E}[g(X)]$$

where  $\mathbb{E}[g(X)] := \frac{1}{n} \sum_i g(X_i)$ , is the empirical mean.

**Example 1** (Exponential Rate). Estimate  $\lambda$  in  $X_i \sim \exp(\lambda)$ ,  $i = 1, \dots, n$ , i.i.d.  $E[X] = 1/\lambda \Rightarrow \hat{\lambda} = 1/\mathbb{E}[X]$

**Example 2** (Linear Regression). Estimate  $\beta$  in  $Y \sim \mathcal{N}(X\beta, \sigma^2 I)$ , a  $p$  dimensional random vector.  $E[Y] = X\beta$  and  $\mathbb{E}[Y] = y$ . Clearly, moment matching won't work because no  $\beta$  satisfies  $X\beta = Y$ . A technical workaround: Since  $\beta$  is  $p$  dimensional, I need to find some  $g(Y) : \mathbb{R}^n \mapsto \mathbb{R}^p$ . Well,  $g(Y) := XY$  is such a mapping. I will use it, even though my technical justification is currently unsatisfactory. We thus have:  $E[X'Y] = X'X\beta$  which I match to  $\mathbb{E}[X'Y] = X'y$ :

$$X'X\beta = X'y \Rightarrow \hat{\beta} = (X'X)^{-1}X'y.$$

## 1.2 Quantile matching

The fundamental idea: match empirical quantiles to theoretical. Denoting by  $F_X(t)$  the CDF of  $X$ , then  $F_X^{-1}(\alpha)$  is the  $\alpha$  quantile of  $X$ . Also denoting by  $\mathbb{F}_X(t)$  the Empirical CDF of  $X_1, \dots, X_n$ , then  $\mathbb{F}_X^{-1}(\alpha)$  is the  $\alpha$  quantile of  $X_1, \dots, X_n$ . The quantile matching method thus implies estimating

$$F_X^{-1}(\alpha)$$

by

$$\mathbb{F}_X^{-1}(\alpha).$$

**Example 3** (Exponential rate). Estimate  $\lambda$  in  $X_i \sim \exp(\lambda)$ ,  $i = 1, \dots, n$ , i.i.d.

$$\begin{aligned} F_X(t) &= 1 - \exp(-\lambda t) = \alpha \Rightarrow \\ F_X^{-1}(\alpha) &= \frac{-\log(1 - \alpha)}{\lambda} \Rightarrow \\ F_X^{-1}(0.5) &= \frac{-\log(0.5)}{\lambda} \Rightarrow \\ \hat{\lambda} &= \frac{-\log(0.5)}{\mathbb{F}_X^{-1}(0.5)}. \end{aligned}$$

### 1.3 Maximum Likelihood

The fundamental idea is that if the data generating process (i.e., the *sampling distribution*) can be assumed, then the observations are probably some high probability instance of this process, and not a low probability event: Let  $X_1, \dots, X_n \sim P_\theta$ , with density (or probability)  $p_\theta(X_1, \dots, X_n)$ . Denote the likelihood, as a function of  $\theta$ :  $\mathcal{L}(\theta) : p_\theta(X_1, \dots, X_n)$ . Then

$$\hat{\theta}_{ML} := \operatorname{argmax}_\theta \{\mathcal{L}(\theta)\}$$

Using a monotone mapping such as the log, does not change the *argmax*. Denote

$$L(\theta) := \log(\mathcal{L}(\theta))$$

**Example 4** (Exponential rate). Estimate  $\lambda$  in  $X_i \sim \exp(\lambda)$ ,  $i = 1, \dots, n$ , i.i.d. Using the exponential PDF and the i.i.d. assumption

$$\mathcal{L}(\lambda) = \lambda^n \exp(-\lambda \sum_i X_i),$$

and

$$L(\lambda) = n \log(\lambda) - \lambda \sum_i X_i.$$

By differentiating and equating 0, we get  $\hat{\lambda}_{ML} = 1/\mathbb{E}[X]$ .

**Example 5** (Discrete time Markov Chain). Estimate the transition probabilities,  $p_1$  and  $p_2$  in a two state,  $\{0, 1\}$ , discrete time, Markov chain where:  $P(X_{t+1} = 1 | X_t = 0) = p_1$  and  $P(X_{t+1} = 1 | X_t = 1) = p_2$ . The likelihood:

$$\mathcal{L}(p_1, p_2) = P(X_2, \dots, X_T; X_1, p_1, p_2) = \prod_{t=1}^T P(X_{t+1} = x_{t+1} | X_t = x_t).$$

We denote  $n_{ij}$  the total number of observed transitions from  $i$  to  $j$  and get that  $\hat{p}_1 = \frac{n_{01}}{n_{01}+n_{00}}$ , and that  $\hat{p}_2 = \frac{n_{11}}{n_{11}+n_{10}}$ .

**Remark 1.** Well, this is a rather artificial example, as because of the Markov property, and the stationarity of the process, we only need to look at transition events, themselves Brenoulli distributed. This example does show, however, the power of the ML method to deal with non i.i.d. samples. As does the next example.

**Example 6** (Brownian motion with drift). Estimate the drift parameter  $a$ , in a discrete time Gaussian process where:  $X_{t+1} = X_t + \varepsilon; \varepsilon \sim \mathcal{N}(0, \sigma^2) \Rightarrow X_{t+1}|X_t \sim \mathcal{N}(aX_t, \sigma^2)$ .

We start with the conditional density at time  $t + 1$ :

$$p_{X_{t+1}|X_t=x_t}(x_{t+1}) = (2\pi\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(x_{t+1} - ax_t)^2\right).$$

Moving to the likelihood:

$$\mathcal{L}(a) = (2\pi\sigma^2)^{-T/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{t=1}^T (x_{t+1} - ax_t)^2\right).$$

Differentiating with respect to  $a$  and equating 0 we get  $\hat{a}_{ML} = \frac{\sum x_{t+1}x_t}{\sum x_t^2}$ .

We again see the power of the ML device. Could we have arrive to this estimator by intuition alone? Hmmmm... maybe. See that  $Cov[X_{t+1}, X_t] = a Var[X_t] \Rightarrow a = \frac{Cov[X_{t+1}, X_t]}{Var[X_t]}$ . So  $a$  can also be derived using the moment matching method which is probably more intuitive.

**Example 7** (Linear Regression). Estimate  $\beta$  in  $Y \sim \mathcal{N}(X\beta, \sigma^2 I)$ , a  $p$  dimensional random vector. Recalling the multivariate Gaussian PDF:

$$p_{\mu, \Sigma}(y) = (2\pi)^{-n/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(y - \mu)' \Sigma^{-1} (y - \mu)\right)$$

So in the regression setup:

$$\mathcal{L}(\beta) = p_{\beta, \sigma^2}(y) = (2\pi)^{-n/2} |\sigma^2 I|^{-1/2} \exp\left(-\frac{1}{2\sigma^2} \|y - X\beta\|^2\right)$$

and  $\hat{\beta}_{ML}$  equals

$$\hat{\beta}_{ML} = (X'X)^{-1} X'y. \tag{1}$$

## 1.4 M-Estimation and Empirical Risk Minimization

M-Estimation, known as Empirical Risk Minimization (ERM) in the machine learning literature, is a very wide framework which stems from statistical decision theory. The underlying idea is that each realization of  $X$  incurs some loss, and we seek to find a "policy", in this case a parameter,  $\theta^*$  that minimizes the average loss. In the econometric literature, we do not incur a loss, but rather a utility, we thus seek a policy that maximizes the average utility.

Define a loss function  $l(X; \theta)$ , and a risk function, being the expected loss,  $R(\theta) := E[l(X; \theta)]$ . Then

Risk

$$\theta^* := \operatorname{argmin}_{\theta} \{R(\theta)\}.$$
 (2)

As we do not know the distribution of  $X$ , we cannot solve Eq.(2), so we minimize the *empirical* risk. Define the empirical risk as  $\mathbb{R}(\theta) := \mathbb{E}[l(X; \theta)]$ , then

Empirical Risk

$$\hat{\theta} := \operatorname{argmin}_{\theta} \{\mathbb{R}(\theta)\}.$$
 (3)

**Remark 2.** The risk function,  $R(\theta)$  defined above

**Example 8** (Squared Loss). Let  $l(X; \theta) = (X - \theta)^2$ . Then  $R(\theta) = E[(X - \theta)^2] = (E[X] - \theta)^2 + \operatorname{Var}[X]$ . Clearly  $\operatorname{Var}[X]$  does not depend on  $\theta$  so that  $R(\theta)$  is minimized by  $\theta^* = E[X]$ . **We thus say that the expectation of a random variable is the minimizer of the squared loss.**

How do we estimate the population expectation? Well a natural estimator is the empirical mean, which is also the minimizer of the empirical risk  $\mathbb{R}(X)$ . The proof is immediate by differentiating.

**Example 9** (Least Squares Regression). Define the loss  $l(Y, X; \beta) := \frac{1}{2}(Y - X\beta)^2$ . Computing the risk,  $E[\|Y - X\beta\|^2]$  will require dealing with the  $X$ 's by either assuming the *Generative Model*<sup>1</sup>, as expectation is taken over  $X$  and  $Y$ . We don't really care about that right now. We merely want to see that the empirical risk minimizer, is actually the classical OLS Regression. And well, it is, by definition...

Generative Model

$$\mathbb{R}(\beta) = \sum_{i=1}^n \frac{1}{2}(y - x_i\beta)^2 = \frac{1}{2}\|y - X\beta\|^2.$$

<sup>1</sup>A Generative Model is a supervised learning problem where we use the assumed distribution of the  $X$ s and not only  $Y|X$ . The latter are known as Discriminative Models.

Minimization is easiest with vector derivatives, but I will stick to regular derivatives:

$$\frac{\partial \mathbb{R}(\beta)}{\partial \beta_j} = \sum_i \left[ (y_i - \sum_{j=1}^p x_{ij} \beta_j) (-x_{ij}) \right]$$

Equating 0 yields  $\hat{\beta}_j = \frac{\sum_i y_i x_{ij}}{\sum_i x_{ij}^2}$ . Solving for all  $j$ 's and putting in matrix notation we get

$$\hat{\beta}_{OLS} = (X'X)^{-1}X'y. \quad (4)$$

## 1.5 Notes

**Maximum Likelihood** If we set the loss function to be the negative log likelihood of the (true) sampling distribution, we see that maximum likelihood estimators in independent samples are actually a certain type of M-estimators.

## 2 From Estimation to Supervised Learning

### 2.1 Empirical Risk Minimization (ERM) and Inductive Bias

In Supervised Learning, or *learning with teacher* problems where we want to extract the relation  $y = f(x)$  between attributes  $x$  and some outcome  $y$ . It is implicit, and essential, that the outcomes are observed. If this is not the case, see Unsupervised Learning (§5) The attributes, also known as features, or predictors, are assumed to belong to some *feature space*  $\mathcal{X}$ .

Feature  
Space

In particular, we don't need to explain the causal process relating the two, so there is no need to commit to a sampling distribution. The implied ERM problem is thus

$$\hat{f}(x) = \operatorname{argmin}_f \left\{ \sum_i l(y_i - f(x_i)) \right\}. \quad (5)$$

Alas, there are clearly infinitely many  $f$  for which  $\mathbb{R}(\hat{f}(x)) = 0$ , in particular, all those where  $\hat{f}(x_i) = y_i$ . All these  $f$  feel like very bad predictors, as they *overfit* the observed data, at a cost of generalizability. We will formalize this intuition in Section 4.

Overfit-  
ting

We need to make sure that we do not learn overly complex poor predictors. Motivated by the fact that humans approach new problems equipped with

their past experience, this regularization is called *Inductive Bias*. There are several ways to introduce this bias, which can be combined:

Induc-  
tive  
Bias

**The Hypothesis Class** We typically do not allow  $f$  to be “any function” but rather restrict it to belong to a certain class. In the machine learning terminology,  $f$  is a Hypothesis, and it belongs to  $\mathcal{F}$  which is the Hypothesis Class.

**Prior Knowledge** We do not need to treat all  $f \in \mathcal{F}$  equivalently. We might have prior preferences towards particular  $f$ 's and we can introduce these preference in the learning process. This is called *Regularization*.

**Non ERM Approaches** Many learning problems can be cast as ERM problems, but another way to introduce bias is by learning  $f$  via some other scheme, which cannot be cast as an ERM problem. Learning algorithms that cannot be cast as ERMs include: Nearest Neighbour, Kernel Smoothing, Boosting.

We now proceed to show that many supervised learning algorithms are in fact ERMs with some type of inductive bias.

## 2.2 Ordinary Least Squares (OLS)

As seen in Example 9, by adopting a squared error loss, and restricting  $\mathcal{F}$  by assuming  $f$  is a linear function of  $x$ , we get the OLS problem. In this case, learning  $f$  is effectively the same as learning  $\beta$  as they are isomorphic.

**Regularization** While the OLS problem seemingly has no regularization parameters, we are in no way restricted to the original  $X$  matrix supplied to us. We are free to choose any subset of the  $X$  variables. Selecting a subset of the  $X$  variables is known as *model selection*. We will discuss many different methods to choose the regularization level, in this case, the variables in the OLS, in Section 4.

Model  
Selec-  
tion

We are not only free to select a subset of  $X$ s, but we can also augment  $X$  with new variables, consisting of non linear transformation of the original ones. This is known as *basis augmentation*. The idea of basis augmentation is a useful and fundamental one that will revisit us throughout the course, even if we typically don't call it by its name.

Basis  
Aug-  
menta-  
tion

**Remark 3** (OLS and Linear Regression). In this text, we distinguish between OLS and Linear Regression. In these notes, we refer to linear regression when we assume that the data generating process is actually  $y = x\beta + \varepsilon$ ,



whereas in OLS we merely fit a linear function without claiming it is the data generating one.

**Remark 4** (OLS Extensions). Most if not all extensions of OLS, such as Generalized Least Squared (GLS) and Generalized Linear Models (§2.5) are ERM problems.

## 2.3 Ridge Regression

Consider the Ridge regression problem:

$$\operatorname{argmin}_{\beta} \left\{ \frac{1}{n} \sum_i (y_i - x_i \beta)^2 + \frac{\lambda}{2} \|\beta\|^2 \right\} \quad (6)$$

$$\hat{\beta}_{\text{Ridge}} = (X'X + \lambda I)^{-1} X'y \quad (7)$$

We can see that again,  $\mathcal{F}$  is restricted to be the space of linear functions of  $x$ , but we also add a regularization that favors the linear functions with small coefficients.

The regularization of  $\beta$  can have several interpretations and justifications.

**A mathematical device** Strengthening the diagonal of  $X'X$  makes it more easily invertible. This is a standard tool in applied mathematics called Tikhonov Regularization. It is also helpful when dealing with multicollinearity, as  $(X'X + \lambda I)$  is always invertible.

**A Subjective Bayesian View** If we believe that  $\beta$  should be small; say our beliefs can be quantified by  $\beta \sim \mathcal{N}(0, \lambda I)$ , then the Ridge solution is actually the mean of our posterior beliefs on  $\beta|y$ .

Whatever the justification, it can be easily shown that  $\frac{\partial R(\lambda, \beta)}{\partial \lambda}$  at  $\lambda = 0$  is negative, thus, we can only improve the predictions by introducing some regularization.

For more on Ridge regression see Hastie et al. [2003].

## 2.4 LASSO

Consider the LASSO problem:

$$\operatorname{argmin}_{\beta} \left\{ \frac{1}{n} \sum_i (y_i - x_i \beta)^2 + \lambda \|\beta\|_1^2 \right\} \quad (8)$$

As can be seen, just like in Ridge regression,  $\mathcal{F}$  is restricted to linear functions. The regularization however differs. Instead of  $l_2$  penalty, we use an

$l_1$  penalty. Eq.(8) does not have a closed form solution for  $\hat{\beta}$  but the LARS algorithm, a quadratic programming algorithm, solves it efficiently.

The LASSO has gained much popularity as it has the property that  $\hat{\beta}_{LASSO}$  has many zero entries. It is thus said to be *sparse*. The sparsity property is very attractive as it acts as a model selection method, allowing to consider  $X$ s where  $p > n$ , and making predictions computationally efficient.

Spar-  
sity

The sparsity property can be demonstrated for the orthogonal design case ( $X'X = I$ ) where  $\hat{\beta}$  admits a closed form solution:

$$\hat{\beta}_{j,LASSO} = \text{sign}(\beta_j) \left[ |\hat{\beta}_{j,OLS}| - \frac{\lambda}{2} \right]_+ . \quad (9)$$

We thus see that the LASSO actually performs *soft thresholding* on the OLS estimates.

Soft  
Thresh-  
olding

## 2.5 Logistic Regression

The logistic regression is the first categorical prediction problem. I.e., the outcome  $y$  is not a continuous variable, but rather takes values in some finite set  $\mathcal{G}$ . In the logistic regression problem, it can take two possible values. In the statistical literature,  $y$  is encoded as  $\mathcal{G} = \{0, 1\}$  and  $f$  is assumed to take to take the following form:

Cate-  
gorical  
Predic-  
tion

$$P(y = 1|x) = \Psi(x\beta) \quad (10)$$

$$\Psi(t) = \frac{1}{1 + e^{-t}} \quad (11)$$

The hypothesis class  $\mathcal{F}$  is thus all  $f(x) = \Psi(x\beta)$ . In the  $\{0, 1\}$  encoding, the loss is the negative log likelihood, i.e.:

$$l(y, x, \beta) = -\log [\Psi(x\beta)^y (1 - \Psi(x\beta))^{1-y}] . \quad (12)$$

In the learning literature it is more common for  $\{1, -1\}$  encoding of  $y$  in which case the loss is

$$l(y, x, \beta) = -\log [1 + \exp(-yf(x))] . \quad (13)$$

**How to classify?** In the  $\{0, 1\}$  encoding, we predict class 1 if  $\Psi(x\beta) > 0.5$  and class 0 otherwise. The logistic problem thus defines a separating hyperplane  $\mathbb{L}$  between the classes:  $\mathbb{L} = \{x : f(x) = 0.5\}$ .

In the  $\{1, -1\}$  encoding, we predict class 1 if  $\Psi(x\beta) > 0$  and class 0 otherwise. The plane  $\mathbb{L}$  is clearly invariant to the encoding of  $y$ .

**Remark 5** (Log Odds Ratio). The formulation above, implies that the log odds ratio is linear in the predictors:

$$\log \frac{P(y = 1|x)}{P(y = 0|x)} = x\beta$$

**Remark 6** (GLMs). Logistic regression is a particular instance of the very developed theory of Generalized Linear Models. These models include the OLS, Probit Regression, Poisson Regression, Quasi Likelihood, Multinomial Regression, Proportional Odds regression and more. The ultimate reference on the matter is McCullagh and Nelder [1989]. GLM

## 2.6 Regression Classifier

Can we use the OLS framework for prediction? Yes! With proper encoding of  $y$ . Solving the same problem from Example 9 by encoding  $y$  as  $\{0, 1\}$  gives us the linear separating hyperplane  $\mathbb{L} : \{x : x\hat{\beta}_{OLS} = 0.5\}$ .

**Remark 7.** We can interpret  $\hat{y}$  as the probability of an event, but there a slight technical difficulty as  $\hat{y}$  might actually be smaller than 0 or larger than 1.

## 2.7 Linear Support Vector Machines (SVM)

We will now not assume anything on the data, and seek for a hyperplane that separates two classes. This, purely geometrical intuition, was the one that motivated Vapnik's support vector classifier [Vapnik, 1998]. In this section we will see the this geometrical intuition can also be seen as an ERM problem over a linear hypothesis class.

**Problem Setup** Encode  $y$  as  $\mathcal{G} = \{-1, 1\}$ . Define a plane  $\mathbb{L} = \{x : f(x) = 0\}$ , and assume a linear hypothesis class,  $f(x) = x\beta + \beta_0$ . Now find the plane  $\mathbb{L}$  that maximizes the (sum of) distances to the data points. We call the minimal distance from  $\mathbb{L}$  to the data points, the *Margin*, and denote it by  $M$ .

To state the optimization problem, we need to note that  $f(x) = x\beta + \beta_0$  is not only the value of our classifier, but it is actually proportional to the signed distance of  $x$  from  $\mathbb{L}$ .

*Proof.* The distance of  $x$  to  $\mathbb{L}$  is defined as  $\min_{x_0 \in \mathbb{L}} \{\|x - x_0\|\}$ . Note  $\beta^* := \beta / \|\beta\|$  is a normal vector to  $\mathbb{L}$ , since  $\mathbb{L} = \{x : x\beta + \beta_0 = 0\}$ , so that for  $x_1, x_2 \in \mathbb{L} \Rightarrow \beta(x_1 - x_2) = 0$ . Now  $x - x_0$  is orthogonal to  $\mathbb{L}$  because  $x_0$  is,

by definition, the orthogonal projection of  $x$  onto  $\mathbb{L}$ . Since  $x - x_0$  are both orthogonal to  $\mathbb{L}$ , they are linearly dependent, so that by the Cauchy Schwarz inequality  $\|\beta^*\| \|x - x_0\| = \|\beta^*(x - x_0)\|$ . Now recalling that  $\|\beta^*\| = 1$  and  $\beta^* x_0 = -\beta_0 / \|\beta\|$  we have  $\|x - x_0\| = \frac{1}{\|\beta\|} (x\beta + \beta_0) = \frac{1}{\|\beta\|} f(x)$ .  $\square$

Using this fact, then  $y_i f(x_i)$  is the distance from  $\mathbb{L}$  to point  $i$ , positive for correct classifications and negative for incorrect classification. The (linear) support vector classifier is defined as the solution to

$$\max_{\beta, \beta_0} \{M \quad \text{s.t.} \quad \forall i : y_i f(x_i) \geq M, \quad \|\beta\| = 1\} \quad (14)$$

If the data is not separable by a plane, we need to allow some slack. We thus replace  $y_i f(x_i) \geq M$  with  $y_i f(x_i) \geq M(1 - \xi_i)$ , for  $\xi_i > 0$  but require that the missclassifications are controlled using a regularization parameter  $C$ :  $\sum_i \xi_i \leq C$ . Eq.(14) now becomes [Hastie et al., 2003, Eq.(12.25)]

$$\max_{\beta, \beta_0} \left\{ M \quad \text{s.t.} \quad \forall i : y_i f(x_i) \geq M(1 - \xi_i), \|\beta\| = 1, \sum_i \xi_i \leq C, \forall i : \xi_i \geq 0 \right\} \quad (15)$$

This is the classical geometrical motivation for support vector classification problem. The literature now typically discusses how to efficiently optimize this problem, which is done via the dual formulation of Eq.(15). We will not go in this direction, but rather, note that Eq.(15) can be restated as an ERM problem:

$$\min_{\beta, \beta_0} \left\{ \sum_i [1 - y_i f(x_i)]_+ + \frac{\lambda}{2} \|\beta\|_2^2 \right\} \quad (16)$$

Eq.(16) thus reveals that the linear SVM is actually an ERM problem, over a linear hypothesis class, with  $l_2$  regularization of  $\beta$ .

See Section 12 in Hastie et al. [2003] for more details on SVMs.

**Remark 8** (Name Origins). SVM takes its name from the fact that  $\hat{\beta}_{SVM} = \sum_i \hat{\alpha}_i y_i x_i$ . The explicit form of  $\hat{\alpha}_i$  can be found in [Hastie et al., 2003, Section 12.2.1]. For our purpose, it suffices to note that  $\hat{\alpha}_i$  will be 0 for all data points far away from  $\mathbb{L}$ . The data points for which  $\hat{\alpha}_i > 0$  are the *support vectors*, which give the method its name.

**Remark 9** (Solve the right problem). Comparing with the logistic regression, and the linear classifier, we see that the SVM cares only about the decision boundary  $\mathbb{L}$ . Indeed, if only interested in predictions, estimating probabilities is a needless complication. As Put by Vapnik:

When solving a given problem, try to avoid a more general problem as an intermediate step.

Then again, if the assumed logistic model of the logistic regression, is actually a good approximation of reality, then it will outperform the SVM as it borrows information from all of the data, and not only the support vectors.

## 2.8 Generalized Additive Models (GAMs)

A way to allow for a more broad hypothesis class  $\mathcal{F}$ , that is still not too broad, so that overfitting is hopefully under control, is by allowing the predictor to be an additive combination of simple functions. We thus allow  $f(x) = \beta_0 + \sum_{j=1}^p f_j(x_j)$ . We also not assume the exact form of  $\{f_j\}_{j=1}^p$  but rather learn them from the data, while constraining them to take some simple form. The ERM problem of GAMs is thus

$$\operatorname{argmin}_{\beta_0, f \in \mathcal{F}} \left\{ \frac{1}{n} \sum_i (y_i - f(x_i))^2 \right\} \quad (17)$$

where  $f$  is as defined above.

**Remark 10** (Not a pure ERM). The learning of  $\{f_j\}_{j=1}^p$  is in fact not performed by optimization, but rather by kernel smoothing (§ 3.2). The solution to Eq.(17) is not a pure ERM problem, but a hybrid between ERM and kernel smoothing.

## 2.9 Projection Pursuit Regression (PPR)

Another way to generalize the hypothesis class  $\mathcal{F}$ , which generalizes the GAM model, is to allow  $f$  to be some simple function of a linear combination of the predictors. Let

$$f(x) = \sum_{m=1}^M g_m(w_m x) \quad (18)$$

where both  $\{g_m\}_{m=1}^M$  and  $\{w_m\}_{m=1}^M$  are learned from the data. The regularization is now performed by choosing  $M$  and the class of  $\{g_m\}_{m=1}^M$ . The ERM problem is the same as in Eq.(17), with the appropriate  $f$ .

**Remark 11** (Not a pure ERM). Just like the GAM problem, in the PPR problem  $\{g_m\}_{m=1}^M$  are learned by kernel smoothing. Solving the PPR problem is thus a hybrid of ERM and Kernel smoothing.

**Remark 12** (Universal Approximator). By choosing a sufficiently large  $M$ , the class  $\mathcal{F}$  can approximate any continuous function. This property of the class is called a *Universal Approximator*.

Univer-  
sal  
Ap-  
proxi-  
mator

## 2.10 Neural Networks (NNETs)

### 2.10.1 Single Hidden Layer

In the spirit of [Hastie et al., 2003, Section 11], we introduce the NNET model via the PPR model, and not through its historically original construction. In the language of Eq.(18), a single-layer-feed-forward neural network, is a model where  $\{g_m\}_{m=1}^M$  are not learned from the data, but rather assumed a-priori.

$$g_m(xw_m) := \beta_m \sigma(\alpha_0 + x\alpha_m)$$

where only  $\{w_m\}_{m=1}^M = \{\beta_m, \alpha_m\}_{m=1}^M$  are learned from the data. A typical *activation function*,  $\sigma(t)$  is the standard logistic CDF:  $\sigma(t) = \frac{1}{1+e^{-t}}$ . Another popular alternative are Gaussian radial functions.

Activa-  
tion  
Func-  
tion

As can be seen, the NNET is merely a non-linear regression model. The parameters of which are often called *weights*.

NNETs have gained tremendous popularity, as it strikes a good balance between model complexity and regularity; particularly in the machine vision, sound analysis and natural language processing domains, where data samples are abundant.

**Loss Functions** For regression the squared loss is used. For classification, one can still use the squared error (as in the Regression Classifier), or the binomial likelihood leading to what is know as the *deviance*, or *cross-entropy* loss.

De-  
viance  
& Cross  
En-  
tropy

**Universal Approximator** Like the PPR, even when  $\{g_m\}_{m=1}^M$  are fixed beforehand, the class is still a universal approximator (although it might require a larger  $M$  than PPR).

**Regularization** Regularization of the model is done via the selection of the  $\sigma$ , the number of nodes/variables in the network and the number of layers. More that one hidden layer leads to *Deep Neural Networks* which offer even more flexibility at the cost of complexity, thus requiring many data samples for fitting.

Deep  
Neural  
Net-  
works

**Back Propagation** The fitting of such models is done via a coordinate-wise gradient descent algorithm called *back propagation*.

**Further Reading** For more on NNETs see [Hastie et al., 2003, Chapter 11]. For a recent overview of Deep Learning in Neural Networks see Schmidhuber [2015].

## 2.11 Classification and Regression Trees (CARTs)

CARTs are a type of ERM where  $f(x)$  include very non smooth functions that can be interpreted as "if-then" rules, also know as *decision trees*.

Deci-  
sion  
Tree

The hypothesis class of CARTs includes functions of the form

$$f(x) = \sum_{m=1}^M c_m I_{\{x \in R_m\}} \quad (19)$$

where  $I_A$  is the indicator function of the event  $A$ . The parameters of the model are the different conditions  $\{R_m\}_{m=1}^M$  and the function's value at each condition  $\{c_m\}_{m=1}^M$ .

Regularization is done by the choice of  $M$  which is called the *tree depth*.

Tree  
Depth

As  $f(x)$  is defined over indicator functions, CARTs have no difficulty to deal with categorical variables and even missing values, on top of the more standard continuous predictors. More on the many advantages of CARTs can be found in the references.

**Optimization** As searching over all possible partitions of the  $x$ 's to optimize  $\{R_m\}_{m=1}^M$  is computationally impossible, optimization is done in a greedy fashion, by splitting on each variable  $x_j$  at a time. The problem presented in Eq. 19 is actually a decision *list*. It is the nature of the optimization that makes the solution a decision *tree*.

Deci-  
sion  
List

**Loss Functions** As usual, a squared loss can be used for continuous outcomes  $y$ . For categorical outcomes, the loss function is called the *impurity measure*. One can use either a misclassification error, the multinomial likelihood (known as the deviance, or cross-entropy), or a first order approximation of the latter known as the *Gini Index*.

Impu-  
rity  
Mea-  
sure

**Dendogram** As CARTs are essentially decision trees, they can typically be viewed as a hierarchy of if-then rules applied on the data. This type of visualization is called a *dendogram*.

Dendo-  
gram

**A Personal Note** While CARTs are very rich hypothesis classes, and easily interpretable, I have poor personal experience with them. I suspect it is their very non smooth nature that is simply inadequate for the problems I have encountered. Then again, the Bagging algorithm, deals with this matter nicely by averaging many trees.

Bag-  
ging

**Further Reading** For more on CARTs and Bagging see [Hastie et al., 2003, Section 9].

## 2.12 Smoothing Splines

The GAM (§2.8) model is an example of learning a non-parametric model, since we did not constrain the parametric form of the functions  $\{f_j\}_{j=1}^p$ . The regularization was part of the optimization algorithm, and not explicit in the problem's definition. *Smoothing Splines* is another non-parametric approach, since we do not assume the parametric form of  $\{f_j\}_{j=1}^p$ . Unlike GAMs, however, the regularization is indeed explicit in the formulation of the problem. Regularization is achieved by penalizing the second derivative of  $f$ , thus forcing  $f$  to be smooth. Considering a single predictor, the ERM problem for *Smoothing Spline*

$$\operatorname{argmin}_f \left\{ \frac{1}{n} \sum_i (y_i - f(x_i))^2 + \lambda \|\nabla^2 f\|_2^2 \right\}. \quad (20)$$

It is quite surprising, and useful, that even though the optimization is performed over an infinite dimensional function space, the solution belongs to a finite dimensional parametric sub-space. The magic continues! It turns out that the fitted values,  $\{\hat{f}(x_i)\}_{i=1}^n$  are linear in  $\{y_i\}_{i=1}^n$ . A fact that greatly facilitates the analysis of the statistical properties of these models.

In the presence of  $p > 1$  predictors, Eq.(20) can be generalized as

$$\operatorname{argmin}_f \left\{ \frac{1}{n} \sum_i (y_i - f(x_i))^2 + \lambda J(\nabla^2 f) \right\}. \quad (21)$$

where  $J$  is some matrix norm. A natural extension is the squared Frobenius norm<sup>2</sup>, which returns a *thin plate spline* as the risk minimizer.

Thin  
Plate  
Spline

---

<sup>2</sup>The Frobenius matrix norm is the sum of squares over all elements:  $\|A\|_F^2 = \sum_{i,j} A_{i,j}^2$



### 3 Non ERM Supervised Learning

Up until now, we have focused on purely ERM, or hybrid ERM methods. Inductive bias, however, can also be introduced by avoiding the optimization approach of ERM. ERM decouples the motivation of a method and the particular learning algorithm (ultimately, some optimization algorithm). In the following methods, the learning algorithm is an integral part of the method. This restricts the learnable hypothesis class, thus acting as a regularizer.

Note that in most of the previous sections<sup>3</sup>, the restriction of the hypothesis class  $\mathcal{F}$  has been imposed by some parametric representation of  $\mathcal{F}$ , over which optimization is performed. The methods in this section do not impose any such parametrization. They are thus also known as *non parametric*.

Non  
Para-  
metric

#### 3.1 k-Nearest Neighbour (KNN)

The fundamental idea behind the KNN approach is that an observation is similar in  $y$  to its surroundings in  $x$ . So say I want to classify the type of a bird ( $y$ ), I merely need to look at the classes of birds which are similar in their attributes ( $x$ 's). Clearly this requires some notion of distance in the feature space  $\mathcal{X}$ , as typically all non-parametric methods do<sup>4</sup>.

The predicted value of a continuous  $y$  at some point  $x$  has the form

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i \quad (22)$$

where  $N_k(x)$  are the indexes of the  $k$  nearest observations to  $x$  in  $\mathcal{X}$ . Eq.(22) merely states that we predict  $y$  to have the average value of the  $y_i$ 's in its neighbourhood.

For a categorical  $y$ , we would replace the averaging with a majority vote of the classes in its neighbourhood.

**Regularization** The regularization of the KNN is performed by choice of  $k$ .

**Universal Approximator** KNN is a universal approximator, in that it can recover approximate any  $f$  relating  $x$  to  $y$ .

---

<sup>3</sup>Well, bar CARTs and Smoothing Splines

<sup>4</sup>Why is this the case? Well, because non parametric methods replace the idea of optimization in a parameter space, with the idea of similarity in neighbourhoods.

**Sample Complexity** While KNN is a universal approximator, it is notoriously known to require many observations to recover even simple relations. With realistic sample sizes, the performance of KNN is typically dominated by other methods.

### 3.2 Kernel Smoothing

Not to be confused with the Kernel Trick in Kernel SVM, Kernel PCA, and Kernel Regression. Kernel *smoothing* is essentially a generalization of a moving average. The *Nadaraya-Watson* weighted average is defined as

$$\hat{f}(x) := \frac{\sum_i \mathcal{K}_\lambda(x, x_i) y_i}{\sum_i \mathcal{K}_\lambda(x, x_i)} \quad (23)$$

Nadaraya-  
Watson

where  $\mathcal{K}_\lambda$  is the kernel function, which is merely a measure of the distance between the evaluation point  $x$  and the data points  $\{x_i\}_{i=1}^n$  and weights the contribution of each  $\{y_i\}_{i=1}^n$  to the value at  $x$ . The denominator merely ensures the weights sum to 1. Regularization is controlled by the  $\lambda$  parameter.

The moving average in a window of width  $\lambda$  is an instance where  $\mathcal{K}_\lambda(x, x_i)$  is fixed in the window. The Band-Pass filter, popular in Electrical Engineering, is an instance of the Nadaraya-Watson smoother, with a Gaussian kernel [TODO: verify].

Band  
Pass

**Metric Spaces** Clearly, as  $\mathcal{K}_\lambda(x, x_i)$  measures distances, it only makes sense if there indeed exists a notion of distance between the  $x$ 's, which is saying that the feature space  $\mathcal{X}$  is a metric space. This can be far from trivial when dealing with categorical predictors such as countries, genomes, gender, etc. Have no worries however, as even with these type of variables, one can define some notion of distance (not claiming it will prove useful in application).

**Relation to KNN** There is an intimate relation between KNN and Kernel Smoothing. Essentially, both predict the value of  $y$  at some  $x$  by averaging their neighbourhood. Averaging can be weighted or not, in both cases. The important distinction is how neighbourhoods are defined. In KNN, the neighbourhood of  $x$  is the  $k$  nearest data points. The radius of the neighbourhood thus varies, while the number of data points does not. In Kernel Smoothing, the neighbourhood of  $x$  is all the data points in the support of the kernel. The radius of the neighbourhood is now fixed, while the number of data points can vary.

### 3.3 Local Likelihood and Local ERM

Although presented as two competing concepts, they can augment each other. We have already seen that Kernel Smoothing plays a part within some ERM algorithms such as GAM (§2.8) and PPR (§2.9).

Another way to marry the two ideas, is by performing EMR only locally, in each neighbourhood of data points defined by a kernel. This idea is very powerful and leads to, e.g., the LOESS (§3.4) and *Local Likelihood* methods.

### 3.4 Local Regression (LOESS)

[TODO]

## 4 Statistical Decision Theory

This section follows the spirit of Section 7 in Hastie et al. [2003], up to some changes in notation.

In Section 2, we gave an intuitive argument for which without some inductive bias, learning will return models with poor performance on new data. In this section we learn how to quantify the performance of a model. In particular, when given new data. This allows us to select among competing candidate models. It will also allow us to choose the value of the regularization parameter of each method.

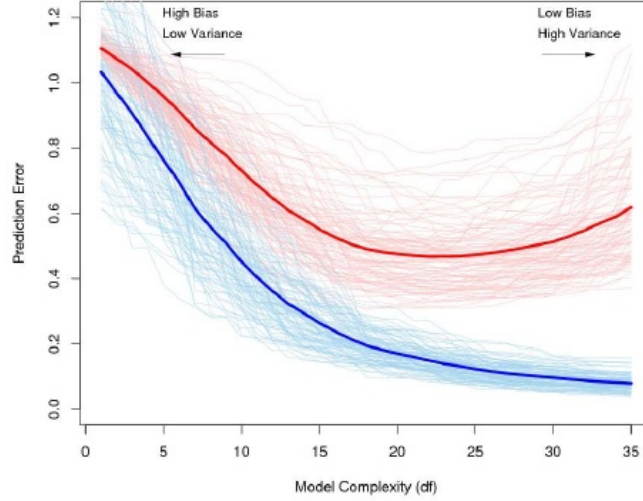
Figure 1 demonstrate the prediction error (red curve) of some model as the model complexity increases. As can be seen, the prediction error decreases as the model becomes more complex, but saturates at some point. This is because the reduction in the bias is smaller than the increase in variance of learning very complex models. This is the celebrated bias-variance tradeoff.

Once we are able to estimate the prediction error from our data, we will seek for a model which minimizes this error.

Before we proceed, we now need to distinguish between several types of prediction errors. The population *risk* of a model parametrized by  $\theta$ , was previously defined as the average loss over all possible data instances, and denoted by  $R(\theta)$  (§1.4). The empirical risk was defined as the average loss over the observed data points, and denoted by  $\mathbb{R}(\theta)$ . We now update these

Bias  
Variance  
Trade-off

## Bias/variance tradeoff



T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer series in statistics. Springer, New York, 2001.

15

Figure 1: Overfitting: Prediction error on new data (red curve) versus the empirical prediction error (light blue). The empirical prediction error will always decrease as more complicated models are fit (moving right). The prediction error on new data, however, will not always decrease and will typically show a local minima.

definitions to deal with the the  $f(x)$  notation of the previous section.

$$R(f) := E_{Y,X} [l(Y, f(X))], \quad (24)$$

$$\mathbb{R}(f) := \mathbb{E}[l(y, f(x))] = \frac{1}{n} \sum_i l(y_i, f(x_i)), \quad (25)$$

$$\bar{R}(f) := \frac{1}{n} \sum_i E_Y [l(Y, f(x_i))], \quad (26)$$

$$R(\hat{f}_n) := E_{\hat{f}_n} \left[ E_{Y,X} [l(Y, \hat{f}_n(X)) | \hat{f}_n] \right]. \quad (27)$$

Eq.(24) is merely a reformulation of  $R(\theta)$  from Section 1.4. It captures the expected loss, a given predictor,  $f(X)$ , will incur on average when given new  $X$ 's and  $Y$ 's. This will be the magnitude which will tell us which models

perform well, and which do not. It is known as the *test error* or also as *prediction error*.

Test  
Error

Eq.(25) is the reformulation of empirical risk,  $\mathbb{R}(\theta)$ , we have been optimizing in Section 2. We referred to it as the *empirical risk*, but it is also known as the *train error*.

Train  
Error

Eq.(26) is the average risk at the observed  $x$ 's, when given new  $Y$ 's<sup>5</sup>. This is the *in sample error*.

In  
Sample  
Error

Eq.(27) is called the *expected prediction error*, i.e., the expected loss when  $f$  is also re-learned. Put differently: How much would we err when: (1) we are given  $n$  new examples  $\mathcal{S}_1$ ; (2) re-learn  $\hat{f}_n$  on  $\mathcal{S}_1$ ; (3) compute the risk of  $\hat{f}_n$  (in the population, not in  $\mathcal{S}_1$ ). We emphasize this by writing  $\hat{f}_n$  instead of  $f$ .  $R(\hat{f}_n)$  is thus not a property of a particular predictor  $f$ , but rather of a whole learning algorithm on random samples of size  $n$ . It could have also been written as  $R(\text{algorithm})$ , although I have not seen this notation in use.

Ex-  
pected  
Predic-  
tion  
Error

We would like to compare the performance of models based on  $R(f)$ , as this will give us an idea on the quality of the prediction on new data. Alas, computing  $R(f)$  requires the distribution of  $y$  and  $x$ , while we only have access to the  $n$  observed samples. Can the empirical risk  $\mathbb{R}(f)$  estimate the unknown risk  $R(f)$ ? Figure 1 suggests it cannot since  $\mathbb{R}(f)$  underestimates  $R(f)$ . Why is this? At an intuitive level: this is because with ERM we learn the  $f$  with smallest error in each sample. It is thus the same as estimating the expected height in a population, by using the minimum in each sample; we will clearly be underestimating the expectation. Then again, there is the hope that we may take this minimum and debias it. This is the goal in the next sections.

Before proceeding, we distinguish between two similar tasks:

**Model Selection** This is the task of selecting between several candidate models.

**Model Assessment** This is the task of assessing the prediction error (i.e., the expected loss, the risk) of a given model.

## 4.1 Train, Validate, Test

If data is abundant, a trivial, assumption free way to estimate  $R(f)$ <sup>6</sup>, is to split the data into 3 sets. A *training set*, used to learn several competing

<sup>5</sup>This magnitude should not be unfamiliar: e.g., inference in ANOVA is performed conditional on the  $x$ 's, which typically stem from a designed experiment.

<sup>6</sup>Think: why  $R(f)$  is being estimated, and not  $R(\hat{f}_n)$  nor  $\bar{R}(f)$ ?

models. A *validation set*, used to check the performance of the learned models and choose the best performer using some comparison measure. A *test set*, used to estimate the risk, as the empirical risk  $\mathbb{R}(f)$  will be unbiased to the population risk  $R(f)$ .

If there is not enough data for this scheme, keep reading...

## 4.2 Unbiased Estimators of the Risk

Under appropriate assumptions, the bias in  $\mathbb{R}(f)$  when estimating  $\bar{R}(f)$ <sup>7</sup> can be computed analytically, and accounted for. The bias  $\bar{R}(f) - \mathbb{R}(f)$  is called the *optimism* of the algorithm. Akaike's Information Criterion (AIC), the finite sample Corrected AIC (AICc), Mallow's Cp (Cp), the Bayesian Information Criterion (BIC, aka SBC, aka SBIC), the Minimum Description Length (MDL), Vapnic's Structural Risk Minimization (SRM), the Deviance Information Criterion (DIC), and the Hannan-Quinn Information Criterion (HQC), all try to estimate  $\bar{R}(f)$  by correcting for the optimism under different assumptions.

Opti-  
mism

Cp,  
AIC,  
BIC,  
MDL,  
SRM

The differences, pros, and cons, of each will not be discussed herein. Just remember what they mean when you see them in your favourite software (R!). They all have in common that you will want the model with the smallest criterion. But be careful- as they are used for model selection, they are indifferent to scaling, and thus should be not interpreted as the expected prediction error.

**Remark 13.** Not all model selection criteria estimate  $\bar{R}(f)$ . The Focused Information Criterion (FIC), for example, does not.

**Further Reading** For a brief review of AIC, BIC, MDL and SRM see Chapter 7 in [Hastie et al., 2003]. For a more rigorous derivation, see Claeskens and Hjort [2008].

## 4.3 Jackknifing

If concerned with over fitting, here is a simple algorithm to estimate the prediction error:

---

<sup>7</sup>In this case, note that it is  $\bar{R}(f)$  being estimated, and not  $R(f)$  nor  $R(\hat{f}_n)$ .

---

**Algorithm 1** Jackknife

---

```
for  $i \in 1, \dots, n$  do
     $\hat{f}^{(i)} \leftarrow$  the learned model with all but the  $i$ 'th observation.
     $l^{(i)} \leftarrow$  the loss of  $\hat{f}^{(i)}$  on the  $i$ 'th observation.
end for
return the average loss over  $l^{(i)}$ .
```

---

This process is called the *Jackknife*, or *Leave-One-Out-Cross-Validation*. This algorithm return an estimator of  $R(\hat{f}_n)$ . This might be quite surprising: every split uses almost an identical sample, so why would it not estimate  $R(f)$ ? See Section 7.12 in Hastie et al. [2003] for details..

But wait! We might be able to stabilize the variability of the estimated error in every split, if instead of leaving only a single observation aside, we leave some more. This lead to way to *K-Fold Cross Validation* in the next section.

## 4.4 Cross Validation

---

**Algorithm 2** Cross Validation

---

```
Split the data into  $K$  parts ("folds").
for  $k \in 1, \dots, K$  do
     $\hat{f}^{(k)} \leftarrow$  the learned model with all except the observations in the  $k$ 'th fold.
     $l^{(k)} \leftarrow$  the loss average of  $\hat{f}^{(k)}$  on the observations in the  $k$ 'th fold.
end for
return the average over  $l^{(k)}$ .
```

---

This simple algorithm estimates  $R(\hat{f}_n)$  without any assumption on the data generating process, and less data than would be required for a “train-validate-test” scheme. Well, as it actually serves for model selection, it should be seen as a “train-validate” scheme, without the “test” part. It is thus *not* an unbiased estimate of  $R(\hat{f}_n)$ . See Section 7.12 in Hastie et al. [2003] for details.

But wait again! The Cross Validation scheme resamples the data *without replacement* to estimate  $R(\hat{f}_n)$ . Could we have sampled it *with* replacement? Yes. This is the idea underlying the *Bootstrapping* scheme.

## 4.5 Bootstrapping

Here is the simplest version of Bootstrap validation:

---

**Algorithm 3** Bootstrap

---

```
for  $b^* \in 1, \dots, B$  do
     $\mathcal{S}^{b^*} \leftarrow n$  randomly selected observations, with replacement, from the
    original data.
     $\hat{f}^{b^*} \leftarrow$  the model learned with  $\mathcal{S}^{b^*}$ .
     $l^{b^*} \leftarrow$  the average loss of  $\hat{f}^{b^*}$  on the observations in the original data.
end for
return the over of  $l^{b^*}$ .
```

---

This algorithm is not a good estimator of  $R(\hat{f}_n)$  as observations play a role both in learning and in validating. Several corrections are available. For details see Section 7.11 in Hastie et al. [2003].

The Bootstrap is a very general scheme, which can be used not only for model validation, but for assessing many of its statistical properties. It is possibly best known when used for hypothesis testing. For more on the Bootstrap, see Efron and Tibshirani [1994].

## 5 Unsupervised Learning

Unlike Supervised Learning, in Unsupervised Learning there is not outcome variable. There is thus no notion of “right” and “wrong”. We merely want to learn the joint distribution of the data,  $X \in \mathcal{X}$ , and represent it in some way we can understand. Describing the data via it’s joint distribution would be nice, but would require many samples if  $X$  has a high dimension (by high we mean  $p > 3!$ ).

For low dimensional data, aim at learning the joint distribution. For higher dimensions we need to set more modest goals:

**Detect high density regions** Find feature combinations which tend to concentrate, hopefully, because they belong to homogenous and interpretable subgroups. This goal is discussed in the present section.

**Find low dimensional representations** Find a low dimensional representation of the joint distribution. This allows the learning from realistic sample sizes, and analysis by, e.g., interpretable parameters and visualization. This goal is discussed in Section 6.

### 5.1 Density Estimation

### 5.2 Agglomerative Clustering

[TODO]



## **5.3 K Means Clustering**

[TODO]

## **5.4 Mixtures**

[TODO]

### **5.4.1 Association Rules**

[TODO]

# **6 Dimensionality Reduction**

[TODO]

## **6.1 In Supervised Learning**

[TODO]

### **6.1.1 Model Selection**

[TODO]

### **6.1.2 LASSO**

[TODO]

## **6.2 In unsupervised Learning**

[TODO]

### **6.2.1 PCA**

[TODO]

### **6.2.2 Multidimensional Scaling (MDS)**

[TODO]

### **6.2.3 Self Organizing Maps**

[TODO]

#### **6.2.4 Principal Curves**

[TODO]

#### **6.2.5 ICA**

[TODO]

#### **6.2.6 kPCA**

[TODO]

#### **6.2.7 ISomap**

[TODO]

#### **6.2.8 LLE**

[TODO]

### **7 Latent Space Models**

[TODO]

#### **7.1 Mixtures**

[TODO]

#### **7.2 Regression Switching**

[TODO]

#### **7.3 Factor Analysis**

[TODO]

#### **7.4 Hidden Markov**

[TODO]

## 7.5 Collaborative Filtering

[TODO]

## List of Algorithms

1	Jackknife . . . . .	23
2	Cross Validation . . . . .	23
3	Bootstrap . . . . .	24

## References

- G. Claeskens and N. L. Hjort. *Model Selection and Model Averaging*. Cambridge University Press, Cambridge ; New York, 1 edition edition, July 2008. ISBN 9780521852258.
- B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, New York, softcover reprint of the original 1st ed. 1993 edition edition, May 1994. ISBN 9780412042317.
- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, July 2003. ISBN 0387952845.
- G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, 1st ed. 2013. corr. 4th printing 2014 edition edition, Aug. 2013. ISBN 9781461471370.
- P. McCullagh and J. A. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC, Boca Raton, 2 edition edition, Aug. 1989. ISBN 9780412317606.
- J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, Jan. 2015. ISSN 08936080. doi: 10.1016/j.neunet.2014.09.003. URL <http://arxiv.org/abs/1404.7828>. arXiv: 1404.7828.
- S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, May 2014. ISBN 9781107057135.
- V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, 1 edition edition, Sept. 1998. ISBN 9780471030034.
- L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer, New York, 2004. ISBN 0387402721 9780387402727.