Class Notes (experimental)

Jonathan Rosenblatt

April 24, 2015

Contents

Chapter 1

Introduction

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], also freely available online. All books are very well written and strongly recommended.

The notation conventions used in this text have been collected in Appendix D.

Chapter 2

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 what follows. These concepts and examples can be found in many introductory books to statistics. I particularly recommend Wasserman [2004] or Abramovich and Ritov [2013].

2.1   Moment matching

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

$$\mathbf{E}\left[g(X)\right]$$

by

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

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

Example 1 (Exponential Rate). Estimate λ in $\mathbf{x}_i \sim exp(\lambda)$, $i = 1, \ldots, n$, i.i.d. $\mathbf{E}\left[x\right] = 1/\lambda. \Rightarrow \widehat{\lambda} = 1/\mathbb{E}\left[x\right]$.

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

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

2.2 Quantile matching

The fundamental idea: match empirical quantiles to theoretical. Denoting by $F_{\mathbf{x}}(t)$ the CDF of $\mathbf{x}$, then $F_{\mathbf{x}}^{-1}(\alpha)$ is the α quantile of $\mathbf{x}$. Also denoting by $\mathbb{F}_x(t)$ the Empirical CDF of $x_1, \ldots, x_n$, then $\mathbb{F}_x^{-1}(\alpha)$ is the α quantile of $x_1, \ldots, x_n$. The quantile matching method thus implies estimating

$$F_{\mathbf{x}}^{-1}(\alpha)$$

by

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

Example 3 (Exponential rate). Estimate λ in $\mathbf{x}_i \sim exp(\lambda)$, $i = 1, \ldots, n$, i.i.d.

$$F_{\mathbf{x}}(t) = 1 - \exp(-\lambda t) = \alpha \Rightarrow$$
$$F_{\mathbf{x}}^{-1}(\alpha) = \frac{-\log(1 - \alpha)}{\lambda} \Rightarrow$$
$$F_{\mathbf{x}}^{-1}(0.5) = \frac{-\log(0.5)}{\lambda} \Rightarrow$$
$$\widehat{\lambda} = \frac{-\log(0.5)}{\mathbb{F}_x^{-1}(0.5)}.$$

2.3 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 $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim P_\theta$, with density (or probability) $p_\theta(x_1, \ldots, x_n)$. Denote the likelihood, as a function of θ: $\mathcal{L}(\theta) : p_\theta(x_1, \ldots, x_n)$. Then

$$\widehat{\theta}_{ML} := argmax_\theta \left\{ \mathcal{L}(\theta) \right\}.$$

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 λ in $X_i \sim exp(\lambda)$, $i = 1, \ldots, n$, 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 $\widehat{\lambda}_{ML} = 1/\mathbb{E}[X]$.

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: $P(\mathbf{x}_{t+1} = 1 | x_t = 0) = p_1$ and $P(\mathbf{x}_{t+1} = 1 | X_t = 1) = p_2$. The likelihood:

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

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

Remark 1 (Confession). 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 Bernoulli 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 (Autoregression of order 1 (AR(1))). Estimate the drift parameter a, in a discrete time Gaussian process where: $\mathbf{x}_{t+1} = a\mathbf{x}_t + \varepsilon; \varepsilon \sim \mathcal{N}(0, \sigma^2) \Rightarrow \mathbf{x}_{t+1} | \mathbf{x}_t \sim \mathcal{N}(ax_t, \sigma^2)$.

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

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

Moving to the likelihood:

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

Taking the log and differentiating with respect to a and equating 0 we get $\widehat{a}_{ML} = \frac{\sum x_{t+1} x_t}{\sum x_t^2}$.

We again see the power of the ML device. Could we have arrived to this estimator by intuiton 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 β in $Y \sim \mathcal{N}\left(X\beta, \sigma^2 I\right)$, 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:

$$\mathcal{L}(\beta) = p_{\beta,\sigma^2}(y) = (2\pi)^{-n/2}|\sigma^2 I|^{-1/2}\exp\left(-\frac{1}{2\sigma^2}\|y-X\beta\|_2^2\right)$$

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

$$\widehat{\beta}_{ML} = (X'X)^{-1}X'y. \tag{2.1}$$

2.4 M-Estimation and Empirical Risk Minimization

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

Definition 1 (Loss Function). The penalty for predicting θ when observing x:

$$l(x;\theta). \tag{2.2}$$

Definition 2 (Risk Function). The expected loss:

$$R(\theta) := \mathbf{E}\left[l(\mathbf{x};\theta)\right]. \tag{2.3}$$

Then the best prediction, θ^*, being the minimizer of the expected risk is

$$\theta^* := argmin_\theta\left\{R(\theta)\right\}. \tag{2.4}$$

As we do not know the distribution of $\mathbf{x}$, we cannot solve Eq.(2.4), so we minimize the *empirical* risk.

Definition 3 (Empirical Risk). The average loss in the sample:

$$\mathbb{R}(\theta) := \mathbb{E}\left[l(x;\theta)\right] = \frac{1}{n}\sum_i l(x_i, \theta). \qquad (2.5)$$

A prediction that can actually be computed with data is thus the empirical minimizer $\widehat{\theta}$:

$$\widehat{\theta} := argmin_\theta \left\{\mathbb{R}(\theta)\right\}. \qquad (2.6)$$

Example 8 (Squared Loss). Let $l(x;\theta) = (x-\theta)^2$. Then $R(\theta) = \mathbb{E}\left[(\mathbf{x}-\theta)^2\right] = (\mathbb{E}\left[\mathbf{x}\right]-\theta)^2 + Var[\mathbf{x}]$. Clearly $Var[\mathbf{x}]$ does not depend on θ so that $R(\theta)$ is minimized by $\theta^* = \mathbb{E}\left[\mathbf{x}\right]$. **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 (Ordinary Least Squares (OLS)). Define the loss $l(y, x; \beta) := \frac{1}{2}(y-x\beta)^2$. Computing the risk, $\mathbb{E}\left[\frac{1}{2}\|\mathbf{y} - \mathbf{x}\beta\|_2^2\right]$ will require dealing with the joint distribution of $(\mathbf{x}, \mathbf{y})$. We don't really care about that right now. We merely want to see that the empirical risk minimizer, is actually the classical OLS problem. And well, it is (by definition actually):

$$\mathbb{R}(\beta) = \sum_{i=1}^n \frac{1}{2}(y_i - x_i\beta)^2 = \frac{1}{2}\|y - x\beta\|_2^2.$$

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

$$\widehat{\beta}_{OLS} = (X'X)^{-1}X'y. \qquad (2.7)$$

2.5   Notes

2.5.1   Maximum Likelihood

Maximum likelihood estimators are a particular instance of M-estimators, if we set the loss function to be the negative log likelihood of the (true) sampling distribution.

2.5.2   Choosing the Loss Function

While by far the most popular, we do not automatically revert to minimizing a squared error loss. There are several considerations when choosing the loss function. Most ERM learning methods can be applied with different loss functions.

The first consideration is computational complexity: if you choose a loss function that leads to a non-convex empirical risk, you are in trouble. There are no guarantees you will be able to find the risk minimizer in finite computing time.

The second consideration is the nature of the outcome y. Some loss functions are more appropriate to continuous y's and some for categorical y's. Typical loss functions for continuous y's are the squared loss, absolute loss, and hinge loss. Typical loss functions for categorical y's are the Binomial likelihood loss (also known as the Cross Entropy, or Deviance), and the hinge loss.

A third consideration, which is rarely given the importance it should get, is "What is the meaning of θ^*"? Or, "What are we actually estimating"? As we have seen in Example 8, the squared loss implies we are aiming to estimate the population mean. What are we estimating when we use the hinge loss? The binomial loss? We will not discuss these matters, as we are discussing methods where these choices have already been made for us. When the day you start thinking of your own learning algorithms, you will need to give some thought to this question.

Chapter 3

Supervised Learning

3.1 Supervised Learning By Empirical Risk Minimization (EMR)

3.1.1 Empirical Risk Minimization and Inductive Bias

In Supervised Learning, or *learning with teacher*, 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 know as *features*, or *predictors*, are assumed to belong to some *feature space* $\mathcal{X}$.

Feature
Space

Unlike classical statistics, we don't care to explain the causal process relating x and y, we just want good predictions. The implied ERM problem is

$$\widehat{f}(x) := argmin_f \left\{ \mathbb{R}(f(x)) \right\} = argmin_f \left\{ \frac{1}{n} \sum_i l(y_i - f(x_i)) \right\}. \qquad (3.1)$$

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

Overfitting

We need to make sure that we do not learn overly complex predictors, which generalize poorly to new data. 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:

Inductive
Bias

Constrained Hypothesis Classes 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. Hypothe-
sis

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

Non ERM Approaches Many learning problems can be cast as ERM problems, but another way to introduce bias is by learning f in a non-optimal manner. Inductive bias is thus introduced by the (non) optimization algorithm Learning algorithms that cannot be cast as ERMs include: Nearest Neighbour (§3.2.1), Kernel Regression(§3.2.2), Boosting (§3.2.4).

We now proceed to show that many supervised learning algorithms are in fact ERMs with some type of inductive bias: constrained hypothesis class and\or or regularization and\or suboptimality.

3.1.2   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 β as they are isomorphic, as will be the case in all hypothesis classes that admit a finite dimensional parametrization.

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 Model
Selection methods to choose the regularization level, in this case, the variables in the OLS, in Section 4.

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

Remark 2 (No Sampling Distribution)**.** We emphasize, that unlike the classical statistical literature, we never assumed any sampling distribution. This

is why we will not be doing hypothesis testing on the model parameters. ERM can be thus seen as *exploratory* statistics.

In particular, we are not discussing hypothesis testing and confidence intervals for the OLS problem, for this same reason: because we are not assuming the data generating distribution. This, in contrast to the classical statistical literature where the linear model is both the data generating process, and the assume hypothesis class.

In statistical terminology, we may be fitting *misspecified models*, we simply do not care about it, as long as predictions are good.

Remark 3 (OLS Extensions)**.** Most if not all extensions of OLS, such as Generalized Least Squared (GLS) and Generalized Linear Models (§3.1.5) are ERM problems.

3.1.3   Ridge Regression

Consider the Ridge regression problem:

$$argmin_\beta \left\{ \frac{1}{n} \sum_i (y_i - x_i\beta)^2 + \frac{\lambda}{2}\|\beta\|_2^2 \right\} \qquad (3.2)$$

$$\widehat{\beta}_{Ridge} = (X'X + \lambda I)^{-1} X'y \qquad (3.3)$$

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 favours linear functions with small coefficients.

The regularization of β can have several interpretations and justifications:

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

A Subjective Bayesian View If we believe that β should be small; say our beliefs can be quantified by $\beta \sim \mathcal{N}(0, 1/\lambda I)$, then the Ridge solution is actually the mean of our posterior beliefs on $\beta|y$.

Whatever justification you prefer, it can be easily shown that $\frac{\partial R(\lambda, \beta)}{\partial \lambda}$ at $\lambda = 0$ is negative, thus, predictions only improve by introducing some small regularization (how small is small? See §4).

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

3.1.4 LASSO

Consider the LASSO problem:

$$argmin_\beta \left\{ \frac{1}{n} \sum_i (y_i - x_i\beta)^2 + \lambda \|\beta\|_1 \right\} \qquad (3.4)$$

As can be seen, just like in Ridge regression, $\mathcal{F}$ is restricted to linear functions of x. The regularization however differs. Instead of l_2 norm penalty, we use an l_1 norm penalty. Eq.(3.4) does not have a closed form solution for $\widehat{\beta}_{LASSO}$ but the LARS algorithm, a quadratic programming algorithm, solves it efficiently.

The LASSO has gained much popularity as it has the property that $\widehat{\beta}_{LASSO}$ 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 Xs where $p > n$, and making predictions computationally efficient.

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

$$\widehat{\beta}_{j,LASSO} = sign(\widehat{\beta}_{j,OLS}) \left[|\widehat{\beta}_{j,OLS}| - \frac{\lambda}{2} \right]_+ . \qquad (3.5)$$

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

3.1.5 Logistic Regression

The logistic regression is the first categorical prediction problem we consider. 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:

$$p(x) := P(\mathbf{y} = 1|x) = \Psi(x\beta) \qquad (3.6)$$

$$\Psi(t) = \frac{1}{1 + e^{-t}} \qquad (3.7)$$

The hypothesis class is thus $\mathcal{F} = \{f : f(x) = \Psi(x\beta)\}$.

In the $\{0, 1\}$ encoding, the loss is the negative log likelihood, i.e.:

$$l(y, x, \beta) = -\log \left[p(x)^y (1 - p(x))^{1-y} \right] = -\log \left[\Psi(x\beta)^y (1 - \Psi(x\beta))^{1-y} \right]. \qquad (3.8)$$

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

$$l(y, x, \beta) = -\log[1 + \exp(-yf(x))]. \qquad (3.9)$$

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 $\mathbb{L} \subset \mathcal{X}$ 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 4 (Generalized Linear Models (GLM)). Logistic regression is a particular instance of the very developped 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

3.1.6 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 $\mathbb{L} = \left\{ x : x\widehat{\beta}_{OLS} = 0.5 \right\}$.

Remark 5. We can interpret $\widehat{y}$ as the probability of an event, but there a slight technical difficulty as $\widehat{y}$ might actually be smaller than 0 or larger than 1.

3.1.7 Linear Support Vector Machines (SVM)

The literature typically presents the SVM method using the geometrical arguments that originally motivated the problem. For our purposes, we present the ERM formulation of the problem, followed by the original arguments. Assuming a linear hypothesis class, $f(x) = \beta_0 + \beta x$, and $\mathcal{G} = \{-1, 1\}$ encoding, the SVM ERM problem is

$$\min_{\beta, \beta_0} \left\{ \sum_i [1 - y_i f(x_i)]_+ + \frac{\lambda}{2} \|\beta\|_2^2 \right\}. \qquad (3.10)$$

Eq.(3.10) thus reveals that the linear SVM is actually an ERM problem, over a linear hypothesis class, with what is called the *hinge* loss function and l_2 regularization of β.

Geometrical Motivation for SVM

For completeness, we present Vapnik's original geometric motivation [Vapnik, 1998] for the formulation of the SVM problem.

We do not assume anything on the data, and seek for a hyperplane $\mathbb{L}$ that seperates two classes. 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 $\min_{x_0 \in \mathbb{L}} \{\|x - x_0\|_2\}$. Note $\beta^* := \beta/\|\beta\|_2$ 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 $\|\beta^*\|_2 \|x - x_0\|_2 = \|\beta^*(x - x_0)\|_2$. Now recalling that $\|\beta^*\|_2 = 1$ and $\beta^* x_0 = -\beta_0/\|\beta\|_2$ we have $\|x - x_0\|_2 = \frac{1}{\|\beta\|_2}(x\beta + \beta_0) = \frac{1}{\|\beta\|_2} f(x)$. $\qquad\square$

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

$$\max_{\beta, \beta_0} \{M \quad s.t. \quad \forall i : y_i f(x_i) \geq M, \quad \|\beta\|_2 = 1\} \tag{3.11}$$

If the data is not separable by a plane, which is typically the case, 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.(3.11) now becomes [Hastie et al., 2003, Eq.(12.25)]

$$\max_{\beta, \beta_0} \left\{M \quad s.t. \quad \forall i : y_i f(x_i) \geq M(1 - \xi_i), \|\beta\|_2 = 1, \sum_i \xi_i \leq C, \forall i : \xi_i \geq 0 \right\} \tag{3.12}$$

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

Remark 6 (Name Origins). SVM takes its name from the fact that $\widehat{\beta}_{SVM} = \sum_i \widehat{\alpha}_i y_i x_i$. The explicit form of $\widehat{\alpha}_i$ can be found in [Hastie et al., 2003, Section

12.2.1]. For our purpose, it suffices to note that $\widehat{\alpha}_i$ will be 0 for all data points far away from $\mathbb{L}$. The data points for which $\widehat{\alpha}_i > 0$ are the *support vectors*, which give the method its name.

Remark 7 (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.

3.1.8 Generalized Additive Models (GAMs)

A way to allow for a more broad hypothesis calss $\mathcal{F}$, 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

$$argmin_{f \in \mathcal{F}} \left\{ \frac{1}{n} \sum_i (y_i - f(x_i))^2 \right\} \tag{3.13}$$

where f is as defined above.

Remark 8 (Not a pure ERM)**.** The learning of $\{f_j\}_{j=1}^{p}$ is in fact not performed by optimization, but rather by kernel smoothing (§ 3.2.2). The solution to Eq.(3.13) is not a pure ERM problem, but a hybrid between ERM and kernel smoothing.

3.1.9 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

$$f(x) = \sum_{m=1}^{M} g_m(w_m x) \tag{3.14}$$

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.(3.13), with the appropriate f.

Remark 9 (Not a pure ERM)**.** Just like the GAM problem, in the PPR problem $\{g_m\}_{m=1}^M$ are learned by Kernel Regresson (§3.2.2). Solving the PPR problem is thus a hybrid of ERM and Kernel Regresson algorithms.

Remark 10 (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*.

3.1.10 Neural Networks (NNETs)

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.(3.14), 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.

$$g_m(x) := \beta_m \sigma(x\alpha_m)$$

where only $\{\beta_m, \alpha_m\}_{m=1}^M$ 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.

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 they strike 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 Like any other ERM problem, we are free to choose the appropriate loss function. 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.

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 σ, 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.

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].

3.1.11   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*.

The hypothesis class of CARTs includes functions of the form

$$f(x) = \sum_{m=1}^{M} c_m I_{\{x \in R_m\}} \qquad (3.15)$$

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*.

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 is a greedy fashion, by splitting on each variable x_j at a time. The problem presented in Eq. 3.15 is actually a decision *list*. It is the nature of the optimization that makes the solution a decision *tree*.

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 (knows as the deviance, or cross-entropy), or a first order approximation of the latter known as the *Gini Index.*

Universal Approximator Like many other hypothesis classes, CART's $\mathcal{F}$ can approximate any function, with large enough M.

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

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.

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

3.1.12 Smoothing Splines

The GAM (§3.1.8) 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*

$$argmin_f \left\{ \frac{1}{n} \sum_i (y_i - f(x_i))^2 + \lambda \|\nabla^2 f\|_2^2 \right\}. \tag{3.16}$$

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, $\left\{ \widehat{f}(x_i) \right\}_{i=1}^{n}$ are linear in $\{y_i\}_{i=1}^{n}$. A fact that greatly facilitates the analysis of the statistical properties of these models.

20

In the presence of $p > 1$ predictors, Eq.(3.16) can be generalized as

$$argmin_f \left\{ \frac{1}{n} \sum_i (y_i - f(x_i))^2 + \lambda J(\nabla^2 f) \right\}. \qquad (3.17)$$

where J is some matrix norm. A natural extension is the squared Frobenius norm[1], which returns a *thin plate spline* as the risk minimizer.

3.2 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 between 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[2], 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*.

3.2.1 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[3].

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

$$\widehat{y}(x) = \frac{1}{k} \sum_{i \in N_k(x)y_i} \qquad (3.18)$$

where $N_k(x)$ are the indexes of the k nearest observations to x in $\mathcal{X}$. Eq.(3.18) merely states that we predict y to have the average value of the y_i's in its neighbourhood.

---

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

[2]Well, bar CARTs and Smoothing Splines

[3]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.

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.

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.

3.2.2 Kernel Regression

Not to be confused with the Kernel Trick in Kernel SVM, Kernel PCA (§5.5.1), and Kernel Density Estimation (§5.3.2).

Kernel *Regression* is essentially a generalization of a moving average. It is also known as a *scan statistic*, or *searchlight statistic*.

The *Nadaraya-Watson* weighted average encompasses most if not all kernel smoothers and is defined as

$$\widehat{f}(x) := \frac{\sum_i \mathcal{K}_\lambda(x, x_i) y_i}{\sum_i \mathcal{K}_\lambda(x, x_i)} \tag{3.19}$$

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 some λ parameter.

The moving average in a window of width λ 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.

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 Regression. 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 Regression, 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.

3.2.3 Local Likelihood and Local ERM

Although presented as two competing concepts, they can augment each other. We have already seen that Kernel Regression plays a part within some ERM algorithms such as GAM (§3.1.8) and PPR (§3.1.9).

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 (§3.2.3) and *Local Likelihood* methods.

Local Regression (LOESS)

While not the original historical motivation, we can think of LOESS as a kernel ERM problem. I.e., minimizing the squared loss, in a sliding window, over some very simple hypothesis class, with weighted observations.

It is more flexible than Kernel Regresson. An intuition to this can be gained by thinking of Kernel Regresson as fitting a local regression with an intercept only, while LOESS also allows local polynomials.

It has several attractive statistical properties. In particular, it can also be used for estimation gradients more accurately than by using simple differences. This flexibility however comes at a slight computational cost. It is thus rare to see high dimensional ($p \geq 3$) LOESS fits, even though the theory can easily be generalized to higher dimensions.

3.2.4 Boosting

Boosting is not an ERM problem, but rather an (meta-)optimization scheme due to Schapire [1990]. A *weak learner* is a learning algorithm slightly better than random guessing. The fundamental idea in Boosting is that a *strong learner* is obtained by starting with some weak learner, then training a series of these on the data, while re-weighting the data points inversely to the error,

and then taking a weighted average of the series of learners as a (strong) predictor.

The ADABoost algorithm [Freund and Schapire, 1997] is an instance of Boosted when applied to CARTs.

It was initially quite surprising and magical that this heuristic algorithm can produce the great predictors it does. It was later shown that ADABoost can be seen as a greedy, self regularizing, optimization algorithm to solving an ERM with log-likelihood loss [Friedman et al., 2000] over the CART hypothesis class.

For more on Boosting and ADABoost see Chapter 10 in Hastie et al. [2003].

3.3 Dimensionality Reduction In Supervised Learning

We start with finding a low representation of x that predicts y. This practice has several different motivations. As a regularization device avoiding overfitting, and as a computational device reducing memory and time complexity. For a general discussion of dimensionality reduction, see Appendix C.

3.3.1 Variable Selection

A trivial way of reducing the dimension of x is by dropping some of it components. Clearly, with p variables, there are 2^p possible variable combinations. Searching over all possible variable combinations is called *subset selection*, but is rarely performed in practice due to computational complexity. It more practical to perform a *forward search*, i.e., a greedy search where we insert on variable at a time. The search stops when the next variable does

not improve the model's accuracy. While tempting, measuring the accuracy using the empirical risk will certainly lead to overfitting (see Figure 4.1). We can thus drive the search with a CV scheme, but the computational burden might be overwhelming. Using some estimator of R auch as AIC, BIC,... is a reasonable and practical way to guide the search.

Algorithm 1 Forward Search

Pick your favourite model selection criterion $\widehat{R}(x)$.
while $\widehat{R}(x)$ keeps getting smaller **do**
    $\widehat{f} \leftarrow$ the model with the smallest $\widehat{R}(x)$ after adding a *single* variable.
end while
return $\widehat{R}(x)$.

Remark 11 (Hypothesis Testing Driven Variable Selection)**.** For a statistician , it is probably very natural to guide a forward search with hypothesis testing. I.e., at each iteration, insert the variable with the most significance. Note however, that when interested in good predictions, we might actually gain accuracy by dropping a significant variable. This is a classical bias-variance tradeoff: dropping a true predictor adds bias, but reduces the variance as we have one less parameter to estimate. If the true effect is small, we can gain from this tradeoff. This intuition is actually used as a model selection criterion [Foster and Stine, 2004]. The bottom line is however, that a **significant variable, does not make it a good predictor.**

3.3.2 LASSO

We presented the LASSO in Section 3.1.4 as a regularized ERM problem. We claimed, without proof, that the l_1 regularization in fact performs soft coefficient thresholding. The estimated $\widehat{\beta}_{LASSO}$ has thus many entries set to zero, acting as a model selection device and reducing the dimension of the problem.

3.3.3 Canonical Correlation Analysis (CCA)

[TODO]

3.3.4 Partial Least Squares (PLS)

[TODO]

3.4 Generative Models In Supervised Learning

For the mere purpose of making a prediction, we do not need to learn the full data generating distribution $P(y, x)$. Knowing this distribution, however,

does permit to make predictions, via Bayes Theorem: $P(y|x) = \frac{P(y,x)}{\int P(y,x)dy}$. Several supervised (and unsupervised) methods make use of this relation to make predictions. These are known as *generative models* . Since the generative distribution serves both in supervised and unsupervised learning, we have dedicated it a section of its own (§B).

3.4.1 Fisher's Linear Discriminant Analysis (LDA)

The fundamental idea behind *Fisher's LDA* (sometimes, just LDA) is that for dichotomous y's: $P(x|y)$ is multivariate Gaussian, with the same covariance for all y. Estimating $P(x|y)$ thus amounts to the estimation of the mean and covariance, which are fairly simple problems that do not require too much data (relatively).

The decision boundaries of this method turn out to be linear in $\mathcal{X}$. Denoting the distance from the class centers by $\widehat{\delta} := (\widehat{\mu}_1 - \widehat{\mu}_2)$, class thus 2 will be predicted if

$$x'\widehat{\Sigma}^{-1}\widehat{\delta} > \frac{1}{2}\widehat{\delta}'\widehat{\Sigma}^{-1}\widehat{\delta} \qquad (3.20)$$

and class 1 otherwise.

Remark 12 (LDA and OLS classification)**.** Interestingly, the two-class LDA, is the same as a regression classifier from Section 3.1.6 [Hastie et al., 2003, Eq. 4.11].

3.4.2 Fisher's Quadratic Discriminant Analysis (QDA)

If you are uncomfortable with the fixed covariance assumption of LDA, one can relax thus assumption. This leads to QDA, which is more flexible than LDA, requiring slightly more data, but permitting the learning of a class of quadratic classifiers, and not only linear.

3.4.3 Naïve Bayes

In LDA (§3.4.1) we dealt with the problem of estimating high dimensional distributions by adding the class-wise Gaussianity assumption. As Gaussians as fairly simple to learn from data, if there is any truth to this assumption, our life improved. Another life-simplifying assumption, replacing the Gaussianity assumption, is that $P(x|y)$ is the product of the margins, i.e., the the features are independent within each class: $P(x|y) = \prod_{j=1}^{p} P(x_j|y)$ This greatly simplifies things as estimating the univariate marginal distributions

$P(x_j|y)$ is a fairly simple problem, which can be done non-parametrically by Kernel smoothing for continuous predictors (§3.2.2), and simple relative frequencies for discrete predictors.

3.5 Ensembles

This section follows the lines of Chapter 8 in Hastie et al. [2003].

Our inductive bias constrains the hypothesis class so that we do not overfit the data. It is overly optimistic to think that nature generates samples in accord with our own inductive bias. Ensemble learning is a "meta" class of algorithms, in which we do not confine ourselves to selecting a single f but rather to selecting (finitely) many, possibly from different classes, and them aggregating them into a single predictor. This is a very flexible framework that might actually return classifiers that do not belong to any of underlying hypothesis classes. As such, it is an *improper learning* algorithm..

Improper
Learning

One might consider many combinations of hypothesis classes, learning methods and aggregation schemes. Several celebrated algorithms fall in this class.

Remark 13. As opposed to a *strong learner*, a *weak learner* is a learning algorithm that is only partially optimized for the observed data. Some of the above algorithms are motivated by the idea of combining many weak learners from the same $\mathcal{F}$ to return a strong learner. Such is the Bagging algorithm. It has been noted, however, that it is preferable[4] to combine strong learners from different hypothesis classes, then to combine weak learners from the same hypothesis class Gashler et al. [2008].

Weak/
Strong
Learner

3.5.1 Committee Methods

Considering a set of M candidate learning algorithms, a *committee method* is simply an average of the M predictors:

Algorithm 2 Commitee Methods

For M candidate learning algorithms.
for $m \in 1, \ldots, M$ **do**
$\quad \widehat{f}^m(x) \leftarrow$ the predictor learned with the m'th algorithm.
end for
$\bar{f}(x) \leftarrow$ the average of $\widehat{f}^m(x)$.
return $\bar{f}(x)$

---

[4]Statistically, not necessarily computationally.

This concept can be generalized by considering different aggregation schemes. The following (meta-)algorithms are merely different aggregation schemes. Most, if not all, can be seen as minimizers of some posterior expected loss. I.e., they minimize some risk with respect to the posterior probabilities of $\left\{\widehat{f}^m(x)\right\}_{m=1}^M$.

3.5.2 Bayesian Model Averaging

The idea of *Bayesian model averaging* is that of using the posterior mean as a predictor. Computing the posterior mean can be quite a tedious task. When dealing with hypotheses from the same parametric class, a more practical approach is that of *inverse BIC weighting*. This is justified by that fact that the BIC (§4.2) is approximately proportional to the posterior probability of $\widehat{f}^m(x)$.

Algorithm 3 Model Averaging

    For M candidate learning algorithms.
    for $m \in 1, \ldots, M$ **do**
        $\widehat{f}^m(x) \leftarrow$ the predictor learned with the m'th algorithm.
        $BIC^m \leftarrow$ the BIC of $\widehat{f}^m(x)$.
    end for
    $\bar{f}(x) \leftarrow$ the average of $\widehat{f}^m(x)$ inversely weighted by BIC^m.
    return $\bar{f}(x)$

BIC is used as weights as it approximates the posterior

3.5.3 Stacking

A simple and powerful (meta-)algorithm by which you train several predictors, and then use their predictions as features for a new predictor. A built-in Jackknifing (or Cross Validation) ensures over-fit models are not overweighted. Here is stacking example for a continuous y:

Algorithm 4 Stacking

For M candidate learning algorithms.

for $i \in 1, \ldots, n$ **do**

    for $m \in 1, \ldots M$ **do**

        $\widehat{f}_m^{(i)}(x) \leftarrow$ the predictor learned with the m'th algorithm, and without the i'th observation.

    end for

end for

$\bar{f}(x_i) \leftarrow$ the OLS prediction using $\left\{ \widehat{f}_m^{(i)}(x_i) \right\}_{m=1}^{M}$ as features.

return $\bar{f}(x)$

3.5.4 Bootstrap Averaging (Bagging)

Generates bootstrap samples and averages the learned predictors (see Algorithm 5). While seemingly completely frequentist, the Bootstrap average can be seen as an estimator of the posterior mean for some non-parametric prior.

A *random forrest* is such an algorithm, when CARTs are fitted and averaged.

Random
Forrest

Algorithm 5 Bagging

Choose the number Bootstraps B.

for $b^* \in 1, \ldots, B$ **do**

    Generate a bootstrap sample of size n: $\mathcal{S}_{b^*}$.

    Learn $\widehat{f}(x)_{b^*}$ from $\mathcal{S}_{b^*}$.

end for

$\bar{f}(x) \leftarrow$ the average of $\widehat{f}(x)_{b^*}$.

return $\bar{f}(x)$

3.5.5 Boosting

The Boosting algorithm in Section 3.2.4 might have the flavour of an Ensemble method, but it should be seen as a regularization scheme. Moving on...

Chapter 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 3.1, 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 4.1 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.

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 θ, was previously defined as the average loss over all possible data instances, and denoted by $R(\theta)$ (§2.4). 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 tradeoff
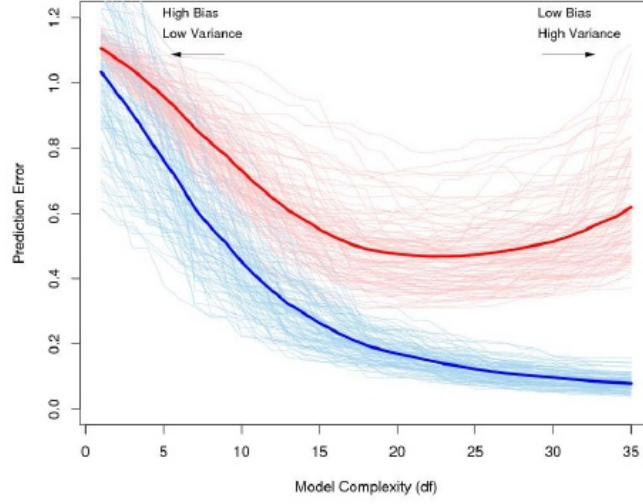
T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer series in statistics. Springer, New York, 2001.

Figure 4.1: 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.

$$R(f) := \mathbf{E_{Y,X}} \left[\mathbf{l}(\mathbf{Y}, \mathbf{f}(\mathbf{X})) \right], \tag{4.1}$$

$$\mathbb{R}(f) := \mathbb{E} \left[l(y, f(x)) \right] = \frac{1}{n} \sum_i l(y_i, f(x_i)), \tag{4.2}$$

$$\bar{R}(f) := \frac{1}{n} \sum_i \mathbf{E_Y} \left[\mathbf{l}(\mathbf{Y}, \mathbf{f}(\mathbf{x_i})) \right], \tag{4.3}$$

$$R(\hat{f}_n) := \mathbf{E_{\widehat{f_n}}} \left[\mathbf{E_{Y,X}} \left[\mathbf{l}(\mathbf{Y}, \widehat{\mathbf{f_n}}(\mathbf{X})) | \widehat{\mathbf{f_n}} \right] \right]. \tag{4.4}$$

Eq.(4.1) is merely a reformulation of $R(\theta)$ from Section 2.4. 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

31

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

Eq.(4.2) is the reformulation of empirical risk, $\mathbb{R}(\theta)$, we have been optimizing in Section 3.1. We referred to it as the *empirical risk*, but it is also known as the *train error*.

Eq.(4.3) is the average risk at the observed x's, when given new Y's [1]. This is the *in sample error*.

Eq.(4.4) 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 $\widehat{f}_n$ on $\mathcal{S}_1$; (3) compute the risk of $\widehat{f}_n$ (in the population, not in $\mathcal{S}_1$. We emphasize this by writing $\widehat{f}_n$ 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 $R(algorithm)$, although I have not seen this notation in use.

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 4.1 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 $R(f)^2$, is to split the data into 3 sets. A *training set*, used to learn several competing

---

[1]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.

[2]Think: why $R(f)$ is being estimated, and not $R(\hat{f}_n)$ nor $\bar{R}(f)$?

models. A *validation set*, used 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 $\bar{R}(f)^3$ 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 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.

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 14. 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:

---

^{3}In this case, note that it is $\bar{R}(f)$ being estimated, and not $R(f)$ nor $R(\hat{f}_n)$.

Algorithm 6 Jackknife
---
    for $i \in 1, \dots, n$ **do**

        $\widehat{f}^{(i)} \leftarrow$ the learned model with all but the i'th observation.

        $l^{(i)} \leftarrow$ the loss of $\widehat{f}^{(i)}$ 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 7 Cross Validation

    Split the data into K parts ("folds").

    for $k \in 1, \dots, K$ **do**

        $\widehat{f}^{(k)} \leftarrow$ the learned model with all *except* the observations in the k'th fold.

        $l^{(k)} \leftarrow$ the loss average of $\widehat{f}^{(k)}$ 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 8 Bootstrap

---
    for $b^* \in 1, \ldots, B$ **do**

        $\mathcal{S}^{b^*} \leftarrow n$ randomly selected observations, with replacement, from the original data.

        $\widehat{f}^{b^*} \leftarrow$ the model learned with $\mathcal{S}^{b^*}$.

        $l^{b^*} \leftarrow$ the average loss of $\widehat{f}^{b^*}$ 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].

Chapter 5

Unsupervised Learning

Unlike Supervised Learning, in Unsupervised Learning there is no 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. Well, maybe "merely" is not the right word, as learning the joint distribution of the data means that instead of learning the relation between a set x and another, y, we now try to learn the relation between all pairs of variables in x, which is clearly more challenging.

Describing the data via it's joint distribution is the pinnacle, but would require many samples if x has a high dimension (by high we mean $p > 3$!). For higher dimensions we need to set more modest goals, which vary according to the purpose of the analysis.

The different goals of unsupervised learning can be

1. **Density Estimation**: Estimate $P(x)$.

2. **High density regions**: Find feature combinations which tend to concentrate, hopefully, because the belong to homogenous and interpretable subgroups.

3. **Low dimensional representation**: 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. Since this also serves in supervised learning, it is discussed with more generality in Appendix C.

4. **Cluster**: Assign observation to homogenous groups. In particular, the particular features of an observation are no longer of interest, once it has been assigned to a cluster.

Relation to Supervised Classification   Learning the joint data distribution, i.e., density estimation, is not only a goal for itself, but can serve for classification. The full data generating distribution, in the context of *supervised* learning is called the *generative model.* See Appendix B for the usage of the estimated generative model for classification.

5.1   Cluster Analysis

In cluster analysis, we aim at assigning observations to (hopefully) homogenous and meaningful clusters.

Groups identified are called *clusters*, so that these methods are known as *cluster analysis*, or *data segmentation* methods.

Cluster analysis is typically easier than learning a joint distribution (§5.3) or even detecting high density regions (§5.2). We will see that for cluster analysis, we don't need the actual features X; as it will turn out, many methods only require some notion of distance between the points, and not their actual features. For this reason, clustering is intimately related to graphs and *networks*[1], which capture only the similarity or dissimilarity of the observations, and not their actual features.

5.1.1   K-Means Clustering

The idea behind K-means clustering is to find a representative point for each of K clusters, and assign each (unlabeled) data point to one of these clusters. As each cluster has a representative point, this is also a *prototype method.*. The clusters are defined so that they minimize the average distance between all points to the center of the cluster.

In K-means, the clusters are first defined, and then similarities computed. This is thus a *top-down* method.

Algorithm 9 K-Means

Choose the number of clusters K.
Arbitrarily assign points to clusters.
while Clusters keep changing **do**
   Compute the cluster centers as the average of their points.
   Assign each point to its closest cluster center (in Euclidean distance).
end while
return Cluster assignments and means.

---

[1]A graph describes a yes/no relation. A network is a weighted graph, which not only describes the existence of a relation, but also its strength.

5.1.2   K-Medoids Clustering

If a Euclidean distance is inappropriate for a particular $\mathcal{X}$, or that robustness to corrupt observations is required, or that we wish to constrain the cluster centers to be actual observations, then the *K-Medoids* algorithm is an adaptation of K-means that allows this.

Algorithm 10 K-Medoids

    Choose a similarity metric $s(x_i, x_j)$.
    Choose the number of clusters K.
    Arbitrarily assign points to clusters.
    while Clusters keep changing **do**
        Within each cluster, set the center as the data point that minimizes the sum of distances to other points in the cluster.
        Assign each point to its closest cluster center (in $s(x_i, x_j)$ distance).
    end while
    return Cluster assignments and centers.

See Section 14.3.10 in Hastie et al. [2003].

5.1.3   Hierarchical Clustering

[TODO]

5.1.4   Spectral Clustering

[TODO]

5.1.5   Self Organizing Maps (SOM)

SOMs are aimed at learning low dimensional representations of data useful for clustering, using the original features X. As such, they take the form of K-means clustering (§5.1.1), but not in the original space $\mathcal{X}$ but rather in a low dimensional representation $\mathcal{M}$.

Mathematics of SOMs   [TODO]
For more on SOMs see Section 14.4 in Hastie et al. [2003].

5.2 Detect High Density Regions

When data data is high dimensional so that we cannot perform density estimation, or when we are merely interested in grouping observations, we can still detect high density regions where observations "clump" together. Hopefully, into meaningful homogenous groups. The following methods are thus also known a *bump hunting* or *mode finding*. They are all different heuristic algorithms that return high density regions.

Bump Hunting

Once high density regions have been detected, they could be assigned to clusters. Then again, assigning to clusters is typically an easier problem which can be solved directly (see §5.1).

5.2.1 Association Rules

Association rules, or *market basket analysis*, or *affinity analysis*, can be seen as approximating the joint distribution with a region-wise constant function (Eq. 3.15). Put differently, we want to capture high density regions of the joint distribution of x with by approximating it with a decision-list (not tree). Learning a decision-list is a computationally impractical problem in general. Association rules are thus typically learned over binary feature spaces $\mathcal{X}$, using heuristic optimization schemes.

Market Basket Analysis

This type of problems typically occurs in sales analysis, where vendors seek for combinations of products that tend to sell jointly (so that can design the store better, or discount product bundles).

The *Aprioiri* algorithm Agraval and Srikant [1994], is an example of such a heuristic search for high density combinations.

Apriori Algorithm

5.3 Density Estimation

We now aim at the possibly hardest target in unsupervised learning- estimating $P(x)$. As previously stated, learning P is of interest for itself, but can also serve for classification (§3.4), for detecting high density regions (§5.2), and for clustering (§5.1).

5.3.1 Parametric Density Estimation

If a parametric generative model can be assumed, this collapses to the parameter estimation problem in the classical statistical literature (§2). Maximum Likelihood estimation being a particularly attractive approach, but certainly not the only one. If a parametric model cannot be assumed, we fall into the

realm of non-parametric methods. As we saw for supervised learning, these typically rely on pooling information from neighbourhoods of $\mathcal{X}$.

5.3.2 Kernel Density Estimation

Much like the Kernel Regression regression, a naïve estimator is the moving average. A natural generalization, in the spirit of the Nadaraya-Watson smoother (Eq.(3.19)) is the *Parzen* estimate:

$$\widehat{p}(x) := \frac{1}{n\lambda} \sum_{i=1}^{n} \mathcal{K}_\lambda(x, x_i). \tag{5.1}$$

Remark 15. If you have been convinced by the use of KNN (§3.2.1) for regression (or classification), there is no reason not to use it for density estimation. It will keep offering the same pros and cons as in KNN regression (or classification).

As previously stated, these methods may fail when x are high dimensional. We thus recur to other methods.

5.3.3 Graphical Models

[TODO]

5.4 Dimensionality Reduction to Linear Subspaces

The idea of ERM and Inductive Bias also applies to unsupervised learning. If $f(x)$ is low dimensional representation of x, mapping it to a "simple" manifold $\mathcal{M}$, we would seek some f that does not incur too much loss, on average. I.e., we seek to minimize $R(f)$.

As usual, we do not have access to the data generating process of x, so we typically content ourselves with the empirical risk minimization. In the context of unsupervised learing, the empirical risk is known as the *reconstruction error*.

The building blocks of a supervised learning problem are: (i) the loss function l, (ii) the hypothesis class $\mathcal{F}$, (iii) a regularization level, and (iv) the optimization scheme. In unsupervised learning, we have similar building blocks. We still have some loss function, regularized or not, penalizing for the poor reconstruction of the data. We can still solve an ERM problem,

or replace the optimization by some other learning algorithm. The main difference is in that instead of some low dimensional mapping $f : x \mapsto \widehat{y}$, we learn a mapping $f : x \mapsto \widehat{x}$.

For a general discussion of the idea of dimensionality reduction see Appendix C.

5.4.1 Principal Components Analysis (PCA)

PCA is such a basic technique, it has been rediscovered and renamed independently in many fields. It can be found under the names of *discrete Karhunen–Loève Transform; Hotteling Transform; Proper Orthogonal Decomposition (POD); Eckart–Young Theorem; Schmidt–Mirsky Theorem; Empirical Orthogonal Functions; Empirical Eigenfunction Decomposition; Empirical Component Analysis; Quasi-Harmonic Modes; Spectral Decomposition; Empirical Modal Analysis;* and possibly more[2]. The many names are quite interesting as they offer an insight into the different problems that led to its (re)discovery.

Starting with an example, consider human height and weight data. While clearly two dimensional data, you don't really need both to understand how "big" are the people in the data. This is because, height and weight vary mostly along a single dimension, which can be interpreted as the "bigness" of an individual. This is why, physicians use the Body Mass Index (BMI) as an indicator of size, instead of a two-dimensional measurement. Assume you now wish to give each individual a size score, that is a linear combination of height and weight: PCA does just that. It returns the linear combination that has the most variability, i.e., the combination which best distinguishes between individuals.

Terminology PCA has received much attention. As such, it has rich underlying theory and terminology. Here are some terms needed to understand PCA outputs:

Principal Components The linear combinations of the features, which best separate between observations. In our example- the "bigness" index of each individual. The first components captures the most variance, the second components, the second most-variance, etc. In terms of $\mathcal{M}$, the principal components are an orthogonal basis for $\mathcal{M}$.

Scores Synonymous to Principal Components.

---

[2]Wikipedia: `http://en.wikipedia.org/wiki/Principal_component_analysis`

Loadings The weights of each data point in each principal component. In our example, the importance of the height and weight in constructing the "bigness" index.

Intuition Notice we have currently offered two motivations for PCA: (i) Find linear combinations that best distinguish between observations, i.e., maximize variance. (ii) Find the linear subspace the bets approximates the data. The reason these two problems are equivalent, is due to the use of the squares error. Informally speaking, the data has some total variance. This variance can be decomposed into the part captured in $\mathcal{M}$, and the part not captured[3]. Since the variance in the data consists of sums of squares, minimizing the distance from X to $\mathcal{M}$, is the same as maximizing the variance of $X \hookrightarrow \mathcal{M}$, since their sum is fixed.

Mathematics of PCA

PCA seeks to represent x in the simplest manifold possible: a linear subspace. Thus, $\mathcal{M}$ is simply a rank q linear subspace in $\mathbb{R}^p$. A low dimensional representation of the points in the data consists of decoding their low dimensional encoding. Since we are dealing with linear spaces, encoding and decoding can be represented with matrices. We thus denote the encoding operator by $E : \mathcal{X} \mapsto \mathbb{R}^q$, a $q \times p$ matrix. We also denote the decoding operator by $D : \mathbb{R}^q \mapsto \mathcal{X}$, a $p \times q$ matrix. Our predictor function (f) in this case is $f(x_i) = DEx_i$.

The ERM problem:

$$argmin_{D,E} \left\{ \frac{1}{n} \sum_i (x_i - DEx_i)^2 \right\}. \tag{5.2}$$

Without proof, it turns out that Eq.(5.2) has a closed form solution. This solution is typically presented using the Singular Value Decomposition (SVD) of X we now define. SVD

Definition 4 (SVD). Any $n \times p$ matrix X, can be decomposed into $X = UDV'$ where U is an $n \times p$ orthogonal matrix ($U'U = I_p$); D is a $p \times p$ diagonal matrix with diagonal elements $d_1 \geq d_2 \geq \cdots \geq d_p$; V is a $p \times p$ orthogonal matrix ($V'V = I_p$).

The solution to Eq.(5.2) is

$$\widehat{E} = V_q' \tag{5.3}$$

$$\widehat{D} = V_q, \tag{5.4}$$

---

[3]Analogous to $SST = SSR + SSE$ in linear regression.

where V_q are the first q columns of V in the Singular Value Decomposition of X.

5.4.2 Sparse Principal Component Analysis (sPCA)

[TODO]

5.4.3 Multidimensional Scaling (MDS)

In MDS we take a distance network between observations, and try to embed it in a lower dimensional linear subspace, while preserving the original distances. Unless the data really is of a low dimension, the embedding may distort the original distances, and may even change the ordering of the observations. The good news is we will be able to visualize and\or cluster them in their new simplified representation. If employed for the purpose of clustering, it is very similar in nature to SOMs (§5.1.5), but differs in that it requires only the similarity network $\mathfrak{S}$, and not the actual features X.

The embedding is merely the assigning of each point to a location in some lower dimensional space. The assignment is driven by a *stress function* which penalizes for the average distortion in pair-wise distances.

Mathematics of MDS Starting with either a dissimilarity network $\mathfrak{D} = (d_{i,j})$, or a similarity network $\mathfrak{S} = (s_{i,j})$. Similarities can be though of as correlations, and dissimilarities as Euclidean distances (which are indeed the typical measures in use). Define $z_i \in \mathbb{R}^q$ the location of point i in the target linear space of rank q. The z_i's are set to minimize the *stress* function. Typical stress functions include:

Classical Scaling Using the centred inner product (i.e. empirical covariance) as the similarity measure and squared error for stress: $s_{i,j} := \langle x_i - \bar{x}, x_j - \bar{x} \rangle$ and the new location are given by

$$argmin_{z_1,...,z_n} \left\{ \sum_{i,j=1}^{n} (s_{i,j} - \langle z_i - \bar{z}, z_j - \bar{z} \rangle)^2 \right\}. \tag{5.5}$$

Kruskal-Shepard Scaling Using squared error for stress:

Sammon Mapping Using the proportion of squared error for stress:

Remark 16 (MDS and PCA)**.** content...

For more on MDS see Section 14.8 in Hastie et al. [2003].

5.5 Dimensionality Reduction to Non Linear Subspaces

5.5.1 Kernel Principal Component Analysis (kPCA)

Back to the motivating example from the PCA section (§5.4.1): assume we want to construct a "bigness" score, that best separates between individuals, but we no longer constrain it to be a linear function of the height and weight. We could thus try to solve

$$argmax_g \left\{ \mathbb{C}ov \left[g(X) \right] \right\} \tag{5.6}$$

where $\mathbb{C}ov \left[g(X) \right]$ is the empirical covariance of g applied on each individual, i.e., row-wise on X.

Alas, just like in the supervised learning problem, without any constraints on g, we might overfit and\or not be able to compute g as optimization is done in a infinite dimensional space. We also need to recall that if we allow for a q dimensional score, then we have to solve Eq.(5.6) q times. We thus have two matters to attend: (i) We need to constrain g so that it does not overfit. (ii) We need the problem to be computable. This is precisely what kPCA does. It turns out, that for a particular class of functions g, the optimal g's take a simple form. The classes of such g's are known as Reproducing Kernel Hilbert Spaces. They are presented in Appendix A.

The fundamental observation allowing the application of RKHS theory, is that the scores in the PCA problem, actually depend only on the similarities between individuals, as measured by their empirical correlation. In kPCA we replace the measure of similarity with some other measure called the *kernel*, chosen from a particular class, and then proceed as if we were solving a PCA problem.

Mathematics of kPCA   [TODO]

5.5.2 Principal Curves

[TODO]

5.5.3 ISOmap

[TODO]

5.5.4 LLE

[TODO]

5.5.5 Random Projections

[TODO: Johnson-Lindenstrauss Lemma]

5.5.6 Compressed Sensing

[TODO]

5.5.7 Information Bottleneck

[TODO]

Remark 17 (Information Bottleneck and ICA). [TODO]

5.6 Latent Space Generative Models

We have already met generative models for supervised learning (§3.4), and
for unsupervised learning (§5.3). We now discuss a class of generative mod-
els, that can be seen as a dimensionality device. The following models and
methods assume that the data generating process is governed by some low
dimensional unobservable *state*, also known as *latent variable*, or *hidden vari-
able*. The latent variables define a *state*. This is why these models are also
known as *state space* models. A term coined by Kalman [1960].

State-
Space
Models

This specification allows us to: (i) *Interpret* the data generating process
via its states. (ii) *Formulate* the density estimation problem as a low dimen-
sional problem, we can actually hope to solve, even when the data itself is
very high dimensional.

The fundamental idea is that while the data generating distribution may
have a complicated form, when we condition on the unobserved variable, the
distribution greatly simplifies.

5.6.1 Mixtures

[TODO]

5.6.2 Regression Switching

[TODO]

5.6.3 Factor Analysis (FA)

Factor Analysis is solved very similarly to PCA (§5.4.1), so that the two are often confused and interchanged. FA, however, stems from a rather different motivation than PCA. In can be seen as a descriptive method using some linear algebra properties. FA is a generative method, estimable using maximum likelihood. In some cases, the two methods can collapse to the same problem. We already know this can happen as we have seen that the Linear Regression problem with Gaussian disturbances can collapse to an OLS problem.

In FA we assume that the observed X's depend linearly on a set of q independent latent (i.e. unobservable) attributes we denote with S. The generative model is thus

$$X = AS + \varepsilon \tag{5.7}$$

Assuming a generative distribution on S and ε, we may try to estimate AS by maximum likelihood. Sadly, recovering the particular latent attributes S from $\widehat{AS}$ is still impossible as there are infinitely many such solutions. To see this, consider an orthogonal *rotation* matrix R ($R'R = I$). For each such R: $AS = AR'RS = A^*S^*$.

The choice of R changes the interpretation of the latent attributes. This is why many researchers find FA an unsatisfactory inference tool.

PCA Terminology

- Factors: The unobserved attributes S.

- Loadings: The A matrix; the contribution of each attribute to the observed X.

- Rotation: An orthogonal re-combination of the latent attributes S and loadings, which changes the interpretation of the result.

5.6.4 Independent Component Analysis (ICA)

ICA aims at overcoming the non-uniquness of the latent attributes in FA (§5.6.3). The fundamental observation is that the non-uniqueness (i.e., non identifiability of S) stems from the fact that the independent Gaussian distribution is invariant under rotation. I.e. for $\mathbf{S} \sim \mathcal{N}(0, \sigma^2 I)$ is the same as that of $R\mathbf{S}$. By replacing the Gaussianity assumption on the distribution of $\mathbf{S}$, by some other multivariate distribution, coordinate-wise independent, we can identify S. This is precisely what ICA does.

Remark 18 (Blind Source Separation). ICA is a popular technique in signal processing, where S is actually the signal (e.g. sound) produced by several different sources. Recovering S is thus recovering the original signals mixing in the recorded X. This is known as *blind source separation.*

Remark 19 (ICA and FA). The solutions to the ICA problem can ultimately be seen as a solution to the FA problem with a particular rotation R. Put differently, by the formulation of the ICA problem, implies a specific rotation so that S is identifiable.

Mathematics of ICA

[TODO]

For more on ICA, see Hyvärinen and Oja [2000].

5.6.5 Hidden Markov Models (HMM)

[TODO]

5.6.6 Collaborative Filtering

[TODO]

5.7 Random Graphs

We have already seen that for the purpose of clustering, we only need some proximity graph between observations, and not the whole features. Random Graphs are a class of *generative models*, but unlike other generative methods, the model does not specify a distribution over the features, but rather a distribution over the proximity measures. For this, it is intimately related to *random graph* and *random matrix* theory.

5.7.1 Erdős Rènyi

[TODO]

5.7.2 Exchangeable Graph Model

[TODO]

5.7.3 $p1$ Graph Model

[TODO]

5.7.4 $p2$ Graph Model

[TODO]

5.7.5 Stochastic Block Graph Model

[TODO]

5.7.6 Latent Space Graph Model

[TODO]

5.7.7 Exponential Random Graphs (ERGMs)

[TODO]

Remark 20 (Relation to Spectral Clustering). [TODO]

Appendix A

The Kernel Trick and Reproducing Kernel Hilbert Spaces (RKHS)

In the context of supervised learning the *kernel trick* is a mathematical device that allows to learn very complicated predictors (f) in a computationally efficient manner. More generally, in the context of unsupervised learning, the kernel tricks allow to learn complicated non-linear mappings of the original features (and not only predictor functions).

Not all predictors and not all problem admit this trick. Then again, many do. Methods for which it applies include: SVM's (§3.1.7), principal components analysis (§5.4.1), canonical correlation analysis (§3.3.3), ridge regression (§3.1.3), spectral clustering (§5.1.4), Gaussian processes[1], and more[2].

We now give an exposition of the method in the context of supervised learning.

Think of smoothing splines (§3.1.12); It was quite magical that without constraining the hypothesis class $\mathcal{F}$, the ERM problem in Eq.(3.16) has a finite dimensional closed form solution. The property of an infinite dimensional problem having a solution in a finite dimensional space is known as the *kernel property*. We with to generalize this observation and ask- which problems have the kernel property? Stating the general optimization problem:

Kernel Property

$$argmin_f \left\{ \frac{1}{n} \sum_i l(y_i, f(x_i)) + \lambda J(f) \right\} \qquad (A.1)$$

---

[1]See the Bayesian interpretation below to see why they apply to Gaussian Processes.

[2]This partial list is taken from Wikipedia: `http://en.wikipedia.org/wiki/Kernel_method`

The problem is then- what type of penalties $J(f)$ will return simple solutions to Eq.(A.1). The answer is: function that belong to *Reproducing Kernel Hilbert Space* function spaces. RKHS's are denoted by $\mathcal{H}_\mathcal{K}$. They include many functions, but they are a rather "small" subset of the space of all possible functions. These spaces, and the functions therein, are defined by another function called a *Kernel* denoted by $\mathcal{K}$. Choosing a particular kernel defines the space and the functions therein. Choosing a particular kernel, also defines the form of J in Eq.(A.1). Put differently: for any choice of a kernel $\mathcal{K}$, there is a particular $J(f)$ for which the solution of Eq.(A.1) will be a function in $\mathcal{H}_\mathcal{K}$ and will be easily computable.

A.1 Mathematics of RKHS

We now show how choosing a kernel $\mathcal{K}$ defines a space $\mathcal{H}_\mathcal{K}$, and a penalty $J(f)$.

A kernel is a non-negative symmetric function of two arguments: $\mathcal{K}(x, y) :$ $\mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}_+$. By fixing y, $\mathcal{K}(x, y)$ is a function with a single argument $x \mapsto \mathcal{K}(x, y)$. $\mathcal{H}_\mathcal{K}$ is merely the space of functions of x, spanned at given y's:

$$f(x) : \sum_m \alpha_m \mathcal{K}(x, y_m) \tag{A.2}$$

From linear algebra, you may know that positive definite matrices be diagonalized. This analogy carries to $\mathcal{K}$, which admits an eigen-expansion:

$$\mathcal{K}(x, y) = \sum_{i=1}^{\infty} \gamma_i \phi(x) \phi(y) \tag{A.3}$$

Using Eqs.(A.3) and (A.2) we can thus expand elements f of $\mathcal{H}_\mathcal{K}$:

$$f(x) = \sum_{i=1}^{\infty} c_i \phi(x) \tag{A.4}$$

where $c_i = \gamma_i \sum_m \alpha_m \phi(y)$. We also define a norm $\|f\|_{\mathcal{H}_\mathcal{K}}^2$ in this space, which is induced by $\mathcal{K}$:

$$\|f\|_{\mathcal{H}_\mathcal{K}}^2 := \sum_{i=1}^{\infty} \frac{c_i^2}{\gamma_i} \tag{A.5}$$

The penalty $J(f)$ in Eq.(A.1), is simply be $\|f\|_{\mathcal{H}_\mathcal{K}}^2$. The f's that solve Eq.(A.1) are guaranteed to have a simple form. They reside in an n dimen-

sional linear function space [Wahba, 1990]:

$$f(x) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(x, x_i) \tag{A.6}$$

The functions $\mathcal{K}(x, x_i)$ can be seen as a basis to the solution space. The good news continue! Being only n dimensional, the norms of these f's, do not require integration but rather only finite summation:

$$\|f\|_{\mathcal{H}_{\mathcal{K}}}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{K}(x_i, x_j) \alpha_i \alpha_j := \alpha' K \alpha. \tag{A.7}$$

Adding the above results, we can restate Eq.(A.1) and say that when fixing $\mathcal{K}$ and using the appropriate J, we only need to solve:

$$argmin_\alpha \left\{ \frac{1}{n} \sum_i l(y_i - K_i \alpha) + \lambda \alpha' K \alpha \right\} \tag{A.8}$$

which is a quadratic programming problem over an n dimensional linear space, easily solvable with numeric routines.

A.1.1 The Bayesian View of RKHS

Just as the ridge regression (§3.1.3) has a Bayesian interpretation, so does the kernel trick. Informally, the functions solving Eq.(A.1) can be seen as the posterior mode if our prior beliefs postulate that the function we are trying to recover is a Gaussian zero-mean process with covariance given by $\mathcal{K}$. This view suggests the intuition that the regularization introduced by $J(f)$ shrinks the estimated f towards a smoother function. At an extreme, where $\lambda \to \infty$, we will recover a constant function, since the the mode of our Gaussian process prior is at the origin of $\mathcal{H}_{\mathcal{K}}$.

Appendix B

Generative Models

By *generative model* we mean that we specify the whole data distribution. This is particularly relevant to supervised learning where many methods only assume the distribution of $P(y|x)$ without stating the distribution of $P(x)$. Assuming only $P(y|x)$ is known as a *discriminative model*, or *discriminative analysis*. In a generative model, in contrast, we assume the whole $P(y, x)$.

For the mere purpose of making a prediction, we do not need to learn $P(y, x)$. Knowing this distribution, however, does permit to make predictions, via Bayes Theorem: $P(y|x) = \frac{P(y,x)}{\int P(y,x)dy}$. Generative models make use of this relation to make predictions.

To gain some intuition, consider a supervised learning problem where the data has an equal number of samples per class. Learning the distribution of x withing each class, allows to a simple classification of a given x to the class with highest probability. LDA (§3.4.1), QDA (§3.4.1), and Naïve Bayes (§3.4.3) follow this exact same rational.

Appendix C

Dimensionality Reduction

Dimensionality reduction is a useful concept in both supervised and unsupervised learning.

The fundamental idea behind dimensionality reduction is that while $\mathcal{X}$ may be high dimensional, thus $P(x)$ hard to learn, there is hope that $\mathbf{x}$ does not really vary in the whole space. If the mass of $P(x)$ is concentrated around some low dimensional manifold $\mathcal{M}$, then the original problem might be approximated to learning the distribution of the projection $P(X \hookrightarrow \mathcal{M})$ on $\mathcal{M}$. If $\mathcal{M}$ is fairly low dimensional, we may hope to visualize and understand $P(X \hookrightarrow \mathcal{M})$ with fairly simple tools. Dimensionality reduction also reduces the memory required to represent the data. It is thus intimately related to *lossy compression* in information theory.

Lossy Compression

A similar reasoning justifies dimensionality reduction in supervised learning. While $P(x)$ might vary in the whole $\mathcal{X}$, but there might be only few directions which carry information on y. Learning $P(y|x)$ can thus be well approximated by $P(y|x \hookrightarrow \mathcal{M})$.

Remark 21. The subspace $\mathcal{M}$, which approximates $P(x)$ may differ than the one that approximates $P(y|x)$. Despite this, it is still quite common for a supervised learning problem to be preceded by an unsupervised dimensionality reduction stage.

Appendix D

Notation

In this text we use the following notation conventions:

x A scalar or vector.

$\mathbf{x}$ A (possibly) vector valued random variable.

X A matrix.

$\mathbf{x}$ A matrix valued random variable (a random matrix).

X' The matrix transpose of X.

$\|x\|_2$ The l_2 norm of x: $\sqrt{\sum_j x_j^2}$.

$\|x\|_1$ The l_1 norm of x: $\sum_j |x_j|$

$\langle x, y \rangle$ The scalar product of two vectors x and y.

$\mathcal{S}$ A data sample.

$\mathbf{E}\left[\mathbf{x}\right]$ The expectation of $\mathbf{x}$.

$\mathbb{E}\left[x\right]$ The empirical expectation (average) of the vector x.

$\mathbf{C}ov\left[\mathbf{x}\right]$ The covariance matrix of $\mathbf{x}$: $\mathbf{E}\left[(\mathbf{x} - \mathbf{E}\left[\mathbf{x}\right])(\mathbf{x} - \mathbf{E}\left[\mathbf{x}\right])'\right]$.

$\mathbb{C}ov\left[x\right]$ The empirical covariance matrix of x: $\mathbb{E}\left[(x - \mathbb{E}\left[x\right])(x - \mathbb{E}\left[x\right])'\right]$.

$F_{\mathbf{x}}(t)$ The CDF of $\mathbf{x}$ at t.

$F_{\mathbf{x}}^{-1}(\alpha)$ The inverse CDF at α (the quantile function).

$\mathbb{F}_x(t)$ The empirical CDF of data vector x.

$\mathbb{F}_x^{-1}(\alpha)$ The empirical α quantile of the data vector x.

$\mathbf{x} \sim P$ The random variable $\mathbf{x}$ is P distributed.

$p(x)$ The density function of P at x.

$\mathcal{N}(\mu, \sigma^2)$ The univariate Gaussian distribution with mean μ and variance σ^2.

$\mathcal{N}(\mu, \Sigma)$ The multivariate Gaussian distribution with mean vector μ and covariance matrix Σ.

$\mathcal{L}(\theta)$ The likelihood function at θ.

$L(\theta)$ The log likelihood function at θ.

$l(x, \theta)$ The loss function of θ at x.

$R(\theta)$ The risk at θ.

$\mathbb{R}(\theta)$ The empirical risk at θ.

$f(x)$ A prediction (hypothesis) at x.

$\mathcal{F}$ The class of all hypotheses f.

$\mathbb{L}$ A hyperplane.

$\mathcal{G}$ A set of categories.

$[t]_+$ The positive part of t.

$\mathcal{K}(x, y)$ A kernel function evaluated at (x, y).

$I_{\{A\}}$ The indicator function of the set A.

$\mathcal{M}$ A manifold.

$\hookrightarrow$ A projection operator.

s_{ij} A similarity measure between observations i and j.

d_{ij} A dissimilarity measure between observations i and j.

$\mathfrak{S}$ A weighted graph (network),i.e. matrix, of similarities between observations.

$\mathfrak{D}$ A weighted graph (network),i.e. matrix, of dissimilarities between observations.

List of Algorithms

Bibliography

F. Abramovich and Y. Ritov. *Statistical theory: a concise introduction*. 2013. ISBN 9781439851845 1439851840.

R. Agraval and R. Srikant. 'Fast Algorithms for Mining Association Rules in Large Data Bases. In *20th Inter-national Conference on Very Large Databases, Santiagom*, 1994.

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.

D. P. Foster and R. A. Stine. Variable Selection in Data Mining. *Journal of the American Statistical Association*, 99(466):303–313, June 2004. ISSN 0162-1459. doi: 10.1198/016214504000000287. URL `http://dx.doi.org/10.1198/016214504000000287`.

Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997. ISSN 0022-0000. doi: 10.1006/jcss.1997.1504. URL `http://www.sciencedirect.com/science/article/pii/S002200009791504X`.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, Apr. 2000. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1016218223. URL `http://projecteuclid.org/euclid.aos/1016218223`.

M. Gashler, C. Giraud-Carrier, and T. Martinez. Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous. In *Seventh Inter-*

national Conference on Machine Learning and Applications, 2008. ICMLA '08, pages 900–905, Dec. 2008. doi: 10.1109/ICMLA.2008.154.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, July 2003. ISBN 0387952845.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4–5):411–430, June 2000. ISSN 0893-6080. doi: 10.1016/S0893-6080(00)00026-5. URL `http://www.sciencedirect.com/science/article/pii/S0893608000000265`.

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.

R. E. Kalman. *Contributions to the Theory of Optimal Control.* 1960.

P. McCullagh and J. A. Nelder. *Generalized Linear Models, Second Edition.* Chapman and Hall/CRC, Boca Raton, 2 edition edition, Aug. 1989. ISBN 9780412317606.

R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00116037. URL `http://link.springer.com/article/10.1007/BF00116037`.

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.

G. Wahba. *Spline Models for Observational Data.* SIAM, 1990. ISBN 9781611970128.

L. Wasserman. *All of statistics: a concise course in statistical inference.* Springer, New York, 2004. ISBN 0387402721 9780387402727.