Class Notes (experimental)

Jonathan Rosenblatt

April 14, 2015

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 A.

Contents

1 Estimation

In this section, we present several estimation principles. Their properties are not discussed, as the section is merely a reminder and a preparation for 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].

1.1 Moment matching

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

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

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. $\mathbf{E}\left[\mathbf{x}\right] = 1/\lambda$. $\Rightarrow \hat{\lambda} = 1/\mathbb{E}\left[x\right]$.

Example 2 (Linear Regression)**.** Estimate β in $\mathbf{y} \sim \mathcal{N}\left(X\beta, \sigma^2 I\right)$, a p dimensional random vector. $\mathbf{E}\left[\mathbf{y}\right] = X\beta$ 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: $\mathbf{E}\left[\mathbf{X}'\mathbf{y}\right] = X'X\beta$ which I match to $\mathbb{E}\left[X'y\right] = X'y$:

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

1.2 Quantile matching

The fundamental idea: match empirical quantiles to theoretical. Denoting by $F_{\mathbf{x}}(t)$ 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.

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

1.3 Maximum Likelihood

The fundamental idea is that if the data generating process (i.e., the *sampling distribution*) can be assumed, then the observations are probably some high probability instance of this process, and not a low probability event: Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim P_\theta$, 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

$$\hat{\theta}_{ML} := argmax_\theta\{\mathcal{L}(\theta)\}.$$

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

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

Example 4 (Exponential rate)**.** Estimate 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 $\hat{\lambda}_{ML} = 1/\mathbb{E}[X]$.

Example 5 (Discrete time Markov Chain)**.** Estimate the transition probabilities, p_1 and p_2 in a two state, $\{0, 1\}$, discrete time, Markov chain where: $P(\mathbf{x}_{t+1} = 1 | x_t = 0) = p_1$ 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 $\hat{\mathbf{p}}_1 = \frac{n_{01}}{n_{01}+n_{00}}$, and that $\hat{\mathbf{p}}_2 = \frac{n_{11}}{n_{11}+n_{10}}$.

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: $\mathbf{x}_{t+1} = \mathbf{x}_t + \varepsilon; \varepsilon \sim \mathcal{N}(0, \sigma^2) \Rightarrow \mathbf{x}_{t+1}|\mathbf{x}_t \sim \mathcal{N}(ax_t, \sigma^2)$.

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:

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

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

We again see the power of the ML device. Could we have arrive to this estimator by 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 $Y \sim \mathcal{N}(X\beta, \sigma^2 I)$, a p dimensional random vector. Recalling the multivariate Gaussian PDF:

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

So in the regression setup:

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

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

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

1.4 M-Estimation and Empirical Risk Minimization

M-Estimation, 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 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.

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

Risk

$$\theta^* := argmin_\theta \{R(\theta)\}. \tag{2}$$

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

Empirical Risk

$$\hat{\theta} := argmin_\theta \{\mathbb{R}(\theta)\}. \tag{3}$$

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

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

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

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

Generative Model

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

[1] A Generative Model is a supervised learning problem where the we use the assumed distribution of the Xs and not only $Y|X$. The latter are know as Discriminative Models.

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

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

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

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

1.5 Notes

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

2 From Estimation to Supervised Learning

2.1 Empirical Risk Minimization (ERM) and Inductive Bias

In Supervised Learning, or *learning with teacher* problems where we want to extract the relation $y = \mathbf{f}(x)$ 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 (§6) The attributes, also know as features, or predictors, are assumed to belong to some *feature space* $\mathcal{X}$.

Feature Space

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

$$\hat{\mathbf{f}}(x) = argmin_{\mathbf{f}} \left\{ \sum_i l(y_i - \mathbf{f}(x_i)) \right\}. \tag{5}$$

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

Overfitting

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

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

Inductive Bias

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

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

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

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

2.2 Ordinary Least Squares (OLS)

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

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

Model Selection

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 is a useful and fundamental one that will revisit us throughout the course, even if we typically don't call it by its name.

Basis Augmentation

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

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

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

2.3 Ridge Regression

Consider the Ridge regression problem:

$$argmin_\beta \left\{ \frac{1}{n} \sum_i (y_i - x_i\beta)^2 + \frac{\lambda}{2}\|\beta\|^2 \right\} \tag{6}$$

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

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

The regularization of 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 believes on $\beta|y$.

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

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

2.4 LASSO

Consider the LASSO problem:

$$argmin_\beta \left\{ \frac{1}{n} \sum_i (y_i - x_i\beta)^2 + \lambda\|\beta\|_1^2 \right\} \tag{8}$$

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

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

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

Sparsity

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

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

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

Soft Thresholding

2.5 Logistic Regression

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

Categorical Prediction

$$P(y = 1|x) = \Psi(x\beta) \tag{10}$$

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

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

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

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

$$l(y, x, \beta) = -\log\left[1 + \exp(-yf(x))\right]. \tag{13}$$

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

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

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

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

Remark 6 (GLMs)**.** Logistic regression is a particular instance of the very 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

2.6 Regression Classifier

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

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

2.7 Linear Support Vector Machines (SVM)

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

Problem Setup Encode y as $\mathcal{G} = \{-1, 1\}$. Define a plane $\mathbb{L} = \{x : \mathbf{f}(x) = 0\}$, and assume a linear hypothesis class, $\mathbf{f}(x) = x\beta + \beta_0$. 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 $\mathbf{f}(x) = x\beta + \beta_0$ is not only the value of our classifier, but it is actually proportional to the signed distance of x from $\mathbb{L}$.

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

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

Using this fact, then $y_i\mathbf{f}(x_i)$ 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

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

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

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

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

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

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

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

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

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

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

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

2.8 Generalized Additive Models (GAMs)

A way to allow for a more broad hypothesis 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 $\mathbf{f}(x) = \beta_0 + \sum_{j=1}^{p} f_j(x_j)$. 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

$$argmin_{\beta_0, \mathbf{f} \in \mathcal{F}} \left\{ \frac{1}{n} \sum_i (y_i - \mathbf{f}(x_i))^2 \right\} \tag{17}$$

where $\mathbf{f}$ is as defined above.

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

2.9 Projection Pursuit Regression (PPR)

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

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

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

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

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

Universal Approximator

2.10 Neural Networks (NNETs)

2.10.1 Single Hidden Layer

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

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

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

Activation Function

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

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

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

Deviance & Cross Entropy

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.

Deep Neural Networks

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

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

2.11 Classification and Regression Trees (CARTs)

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

Decision Tree

The hypothesis class of CARTs includes functions of the form

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

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

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

Tree Depth

As $\mathbf{f}(x)$ 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. 19 is actually a decision *list*. It is the nature of the optimization that makes the solution a decision *tree*.

Decision List

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

Impurity Measure

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*.

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.

Bagging

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

2.12 Smoothing Splines

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

$$argmin_{\mathbf{f}} \left\{ \frac{1}{n} \sum_i (y_i - \mathbf{f}(x_i))^2 + \lambda \|\nabla^2 f\|_2^2 \right\}. \tag{20}$$

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

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

$$argmin_{\mathbf{f}} \left\{ \frac{1}{n} \sum_i (y_i - \mathbf{f}(x_i))^2 + \lambda J(\nabla^2 f) \right\}. \tag{21}$$

where J is some matrix norm. A natural extension is the squared Frobenius norm[2], which returns a *thin plate spline* as the risk minimizer.

Thin Plate Spline

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

3 Non ERM Supervised Learning

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

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

Non Parametric

3.1 k-Nearest Neighbour (KNN)

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

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

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

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

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

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

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

[3]Well, bar CARTs and Smoothing Splines

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

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

3.2 Kernel Smoothing

Not to be confused with the Kernel Trick in Kernel SVM, Kernel PCA, and Kernel Regression.

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

Scan Statistic

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

Nadaraya-Watson

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

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

Band Pass

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

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

The radius of the neighbourhood is now fixed, while the number of data points can vary.

3.3 Local Likelihood and Local ERM

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

Another way to marry the two ideas, is by performing EMR only locally, in each neighbourhood of data points defined by a kernel. This idea is very powerful and leads to, e.g., the LOESS (§3.3.1) and *Local Likelihood* methods.

3.3.1 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 smoothing. An intuition to this can be gained by thinking of kernel smoothing 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.

3.4 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.

ADABoost

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 solv-

ing 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].

4 Statistical Decision Theory

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

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

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

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 $R(\theta)$ (§1.4). The empirical risk was defined as the average loss over the observed data points, and denoted by $\mathbb{R}(\theta)$. We now update these definitions to deal with the the $\mathbf{f}(x)$ notation of the previous section.

$$R(\mathbf{f}) := \mathbf{E}_{\mathbf{Y},\mathbf{X}}\left[\mathbf{l}(\mathbf{Y}, \mathbf{f}(\mathbf{X}))\right], \tag{24}$$

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

$$\bar{R}(\mathbf{f}) := \frac{1}{n}\sum_i \mathbf{E}_{\mathbf{Y}}\left[\mathbf{l}(\mathbf{Y}, \mathbf{f}(\mathbf{x_i}))\right], \tag{26}$$

$$R(\hat{\mathbf{f}}_n) := \mathbf{E}_{\hat{\mathbf{f}}_\mathbf{n}}\left[\mathbf{E}_{\mathbf{Y},\mathbf{X}}\left[\mathbf{l}(\mathbf{Y}, \hat{\mathbf{f}}_\mathbf{n}(\mathbf{X}))|\hat{\mathbf{f}}_\mathbf{n}\right]\right]. \tag{27}$$

Eq.(24) is merely a reformulation of $R(\theta)$ from Section 1.4. It captures the expected loss, a given predictor, $\mathbf{f}(X)$, will incur on average when given new

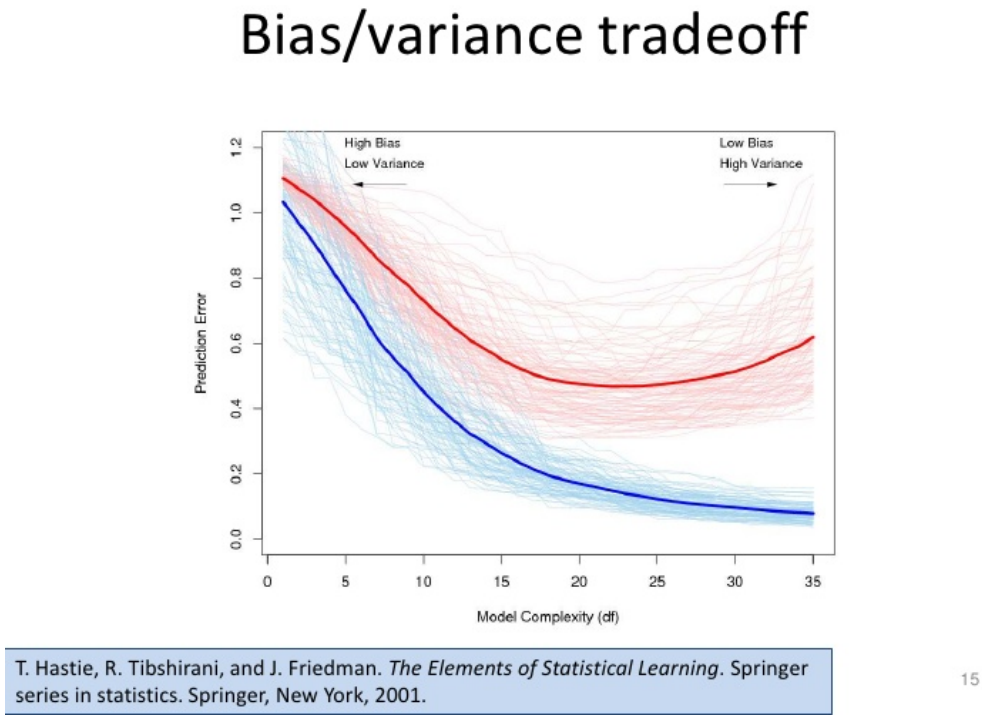

Figure 1: Overfitting: Prediction error on new data (red curve) versus the empirical prediction error (light blue). The empirical prediction error will always decrease as more complicated models are fit (moving right). The prediction error on new data, however, will not always decrease and will typically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

X's and Y's. This will be the magnitude which will tell us which models perform well, and which do not. It is known as the *test error* or also as *prediction error*.

Test Error

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

Train Error

Eq.(26) is the average risk at the observed x's, when given new Y's [5]. This is the *in sample error*.

In Sample Error

Eq.(27) is called the *expected prediction error*, i.e., the expected loss when $\mathbf{f}$ is also re-learned. Put differently: How much would we err when:(1) we are given n new examples $\mathcal{S}_1$; (2) re-learn $\hat{\mathbf{f}}_n$ on $\mathcal{S}_1$; (3) compute the risk of

[5]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.

$\hat{\mathbf{f}}_n$ (in the population, not in $\mathcal{S}_1$. We emphasize this by writing $\hat{\mathbf{f}}_n$ instead of $\mathbf{f}$. $R(\hat{\mathbf{f}}_n)$ is thus not a property of a particular predictor $\mathbf{f}$, 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.

Expected Prediction Error

We would like to compare the performance of models based on $R(\mathbf{f})$, as this will give us an idea on the quality of the prediction on new data. Alas, computing $R(\mathbf{f})$ requires the distribution of y and x, while we only have access to the n observed samples. Can the empirical risk $\mathbb{R}(\mathbf{f})$ estimate the unknown risk $R(\mathbf{f})$? Figure 1 suggests it cannot since $\mathbb{R}(\mathbf{f})$ underestimates $R(\mathbf{f})$. Why is this? At an intuitive level: this is because with ERM we learn the $\mathbf{f}$ 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 $R(\mathbf{f})$[6], is to split the data into 3 sets. A *training set*, used to learn several competing 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 $\mathbb{R}(\mathbf{f})$ will be unbiased to the population risk $R(\mathbf{f})$.

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

4.2 Unbiased Estimators of the Risk

Under appropriate assumptions, the bias in $\mathbb{R}(\mathbf{f})$ when estimating $\bar{R}(\mathbf{f})$[7] can be computed analytically, and accounted for. The bias $\bar{R}(\mathbf{f}) - \mathbb{R}(\mathbf{f})$ is called the *optimism* of the algorithm. Akaike's Information Criterion (AIC), the

Optimism

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

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

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 $\bar{R}(\mathbf{f})$ by correcting for the optimism under different assumptions.

Cp, AIC, BIC, MDL, SRM

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

Remark 13. Not all model selection criteria estimate $\bar{R}(\mathbf{f})$. 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:

Algorithm 1 Jackknife

for $i \in 1, \ldots, n$ **do**
  $\hat{\mathbf{f}}^{(i)} \leftarrow$ the learned model with all but the i'th observation.
  $l^{(i)} \leftarrow$ the loss of $\hat{\mathbf{f}}^{(i)}$ 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 $R(\hat{\mathbf{f}}_n)$. This might be quite surprising: every split uses almost an identical sample, so why would it not estimate $R(\mathbf{f})$? See Section 7.12 in Hastie et al. [2003] for details..

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

4.4 Cross Validation

Algorithm 2 Cross Validation

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

This simple algorithm estimates $R(\hat{\mathbf{f}}_n)$ 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 $R(\hat{\mathbf{f}}_n)$. 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 $R(\hat{\mathbf{f}}_n)$. Could we have sampled it *with* replacement? Yes. This is the idea underlying the *Bootstrapping* scheme.

4.5 Bootstrapping

Here is the simplest version of Bootstrap validation:

Algorithm 3 Bootstrap

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

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

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

5 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 $\mathbf{f}$ 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 14. 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[8] 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

5.1 Committee Methods

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

Algorithm 4 Commitee Methods

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

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.

[8] Statistically, not necessarily computationally.

I.e., they minimize some risk with respect to the posterior probabilities of $\{\hat{\mathbf{f}}^m(x)\}_{m=1}^M$.

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 $\{\hat{\mathbf{f}}^m(x)$.

Algorithm 5 Model Averaging

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

BIC is used as weights as it approximates the posterior

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 6 Stacking

For M candidate learning algorithms.
for $i \in 1, \ldots, n$ **do**
  for $m \in 1, \ldots M$ **do**
    $\hat{\mathbf{f}}_m^{(i)}(x) \leftarrow$ the predictor learned with the m'th algorithm, and without the i'th observation.
  end for
end for
$\bar{\mathbf{f}}(x_i) \leftarrow$ the OLS prediction using $\{\hat{\mathbf{f}}_m^{(i)}(x_i)\}_{m=1}^M$ as features.
return $\bar{\mathbf{f}}(x)$

5.4 Bootstrap Averaging (Bagging)

Generates bootstrap samples and averages the learned predictors (see Algorithm 7). 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 7 Bagging

Choose the number Bootstraps B.
for $b^* \in 1, \ldots, B$ **do**
  Generate a bootstrap sample of size n: $\mathcal{S}_{b^*}$.
  Learn $\hat{\mathbf{f}}(x)_{b^*}$ from $\mathcal{S}_{b^*}$.
end for
$\bar{\mathbf{f}}(x) \leftarrow$ the average of $\hat{\mathbf{f}}(x)_{b^*}$.
return $\bar{\mathbf{f}}(x)$

5.5 Boosting

The Boosting algorithm in Section 3.4 might have the flavour of an Ensemble method, but it should be seen as a regularization scheme. Moving on...

6 Unsupervised Learning

Unlike Supervised Learning, in Unsupervised Learning there is not outcome variable. There is thus no notion of "right" and "wrong". We merely want to learn the joint distribution of the data, $X \in \mathcal{X}$, and represent it in some way we can understand. 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.

The different goals of unsupervised learning are

Density Estimation Estimate the joint density of x over $\mathcal{X}$.

Detect high density regions Find feature combinations which tend to concentrate, hopefully, because the belong to homogenous and interpretable subgroups.

Find low dimensional representations Find a low dimensional representation of the joint distribution. This allows the learning from realistic sample sizes, and analysis by, e.g., interpretable parameters and visualization. This goal is discussed in Section 8.

Relation to Supervised Classification Any supervised classification problem can be solved, at least in theory, by solving an unsupervised. I.e., estimate the joint distribution within each class, and then classify new observations to the highest density class, via Bayes rule. This is known as a *generative* model, as we try to learn the full data generating distribution, unlike the *discriminative* models described above, where the distribution of the features x is of no interest.

6.1 Density Estimation

6.1.1 Parametric Density Estimation

Density estimation deals with the learning of the data generating distribution. If a parametric generative model can be assumed, this collapses to an unsupervised ERM problem. Maximum Likelihood estimation being a particularly attractive approach. If a parametric model cannot be assumed, we fall into the real of non-parametric methods. As we saw for Supervised learning, these typically rely on pooling information from neighbourhoods of $\mathcal{X}$.

6.1.2 Kernel Density Estimation

Much like the Kernel Smoothing regression, a naïve estimator is the moving average. A natural generalization, in the spirit of the Nadaraya-Watson smoother (Eq.(23)) is the *Parzen* estimate:

Parzen Estimate

$$\hat{\mathbf{p}}(x) := \frac{1}{n\lambda} \sum_{i=1}^{n} \mathcal{K}_\lambda(x, x_i). \tag{28}$$

Remark 15. If you have been convinced by the use of KNN (§3.1) 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.

6.2 Detect High Density Regions- Cluster Analysis

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. Either defined as groups of observations, or as regions in the feature space $\mathcal{X}$.

Bump Hunting

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

Since these methods are aim at detecting data clustering, they clearly imply the availability of some measure of distance, or similarity, between data points.

6.2.1 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.

Prototype Methods

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

Top Down Clustering

Algorithm 8 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.

K-Medoids 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 9 K-Medoids

Choose a similarity metric $d(x_i, x_j)$.
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 $d(x_i, x_j)$ distance).
end while
return Cluster assignments and centers.

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

6.2.2 Hierarchical Clustering

[TODO]

6.2.3 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. 19). 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

7 Generative Models

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 gen-

erative distribution serves both in supervised and unsupervised learning, we have dedicated it a section of its own.

7.1 Density Estimation

Clearly, the trivial generative model is the density estimation, discussed in Section 6.1.

7.2 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 $\hat{\delta} := (\hat{\mu}_1 - \hat{\mu}_2)$, class thus 2 will be predicted if

$$x'\hat{\mathbf{\Sigma}}^{-1}\hat{\delta} > \frac{1}{2}\hat{\delta}'\hat{\mathbf{\Sigma}}^{-1}\hat{\delta} \tag{29}$$

and class 1 otherwise.

Relation to OLS classification Interestingly, the two-class LDA, is the same as a regression classifier from Section 2.6 [Hastie et al., 2003, Eq. 4.11].

7.3 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.

7.4 Naïve Bayes

In LDA 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), and simple relative frequencies for discrete predictors.

8 Dimensionality Reduction

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 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 now visualize and analyze $P(X \hookrightarrow \mathcal{M})$ 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 16. 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 preceeded by an unsupervised dimensionality reduction stage.

8.1 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.

8.1.1 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 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.

Subset Selection

Algorithm 10 Forward Search

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

Remark 17 (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 classica 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.**

8.1.2 LASSO

We presented the LASSO in Section 2.4 as a regularized ERM problem. We claimed, without proof, that the l_1 regularization in fact performs soft coefficient thresholding. The estimated $\hat{\beta}_{LASSO}$ has thus many entries set to zero, acting as a model selection device.

8.1.3 Partial Least Squares (PLS) and Canonical Correlation Analysis (CCA)

[TODO]

8.2 Dimensionality Reduction in Unsupervised Learning

The idea of ERM also applies to unsupervised learning. If $\mathbf{f}(x)$ is low dimensional representation of x, mapping it to $\mathcal{M}$, we would seek some $\mathbf{f}$ that does not incur too much loss, on average. I.e., we seek to minimize $R(\mathbf{f}(x))$. 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 know as the *reconstruction error*.

Reconstruction Error

8.2.1 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; quasiharmonic modes; spectral decomposition; empirical modal analysis;* and possibly more[9]. The many names are quite interesting as they offer an insight into the different problems that led to the (re)discovery of PCA.

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: the "bigness" of an individuals. This is why, doctors 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 index, 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.

More generally, PCA seeks to represent x in the simplest manifold possible: a linear subspace. Thus, $\mathcal{M}$ is simply a rank q linear subspace in $\mathbb{R}^p$: $\mathbf{f}(x) := H_q x$ for H_q a $q \times p$ matrix with rank q. The ERM problem is then:

$$\hat{\mathbf{H}}_q := argmin_{H_q} \left\{ \frac{1}{n} \sum_i (x_i - H_q x_i)^2 \right\}. \tag{30}$$

Without going into the optimization details, it turns out that Eq.(30) has a closed form solution. This solution is typically presented using the Singular

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

Value Decomposition (SVD) of X we now define. SVD

Definition 1 (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.(30) is

$$\hat{\mathbf{H}}_q = V_q' V_q, \tag{31}$$

where V_q are the first q columns of V.

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

Loadings The weights of each data point in each component, i.e., the entries of the H_q matrix.

Scores Also known as *component scores*, or *factor scores*, or simply *factors*. The q values of the transformed variables.

8.2.2 Random Projections

[TODO: Johnson-Lindenstrauss Lemma]

8.2.3 Compressed Sensing

[TODO]

8.2.4 Multidimensional Scaling (MDS)

[TODO]

8.2.5 Self Organizing Maps

[TODO]

8.2.6 Principal Curves

[TODO]

8.2.7 ICA

[TODO]

8.2.8 kPCA

[TODO]

8.2.9 ISOmap

[TODO]

8.2.10 LLE

[TODO]

8.2.11 Information Bottleneck

[TODO]

9 Latent Space Models

[TODO]

9.1 Mixtures

[TODO]

9.2 Regression Switching

[TODO]

9.3 Factor Analysis

[TODO]

9.4 Hidden Markov

[TODO]

9.5 Collaborative Filtering

[TODO]

A 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: $\sum_j |x_j|$

$\mathcal{S}$ A data sample.

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

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

$F_{\mathbf{x}}(t)$ 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.

$\mathbf{f}(x)$ A prediction (hypothesis) at x.

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

$\mathbb{L}$ 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.

List of Algorithms

References

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 International 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.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, 1st ed. 2013. corr. 4th printing 2014 edition edition, Aug. 2013. ISBN 9781461471370.

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

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.

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