An overview of statistical learning methods

Konan Hara

August 1, 2019

Contents

1 Introduction

Statistical learning methods aim to predict the outcome for new observation using the data at hand. To understand the relationship between regression methods (e.g., linear regression, logistic regression, and alike) and statistical learning methods, I state the learning problem summarized in Vapnik (1999).

Some generic notations are necessary to be defined before the explanation. Denote f as a generic notation for a probability density function, e.g., $f_X(X = x_0)$. When f indicates the whole distribution like $f_X(X)$, and the subscript of f and the object in the parenthesis are the same, I omit the subscript, i.e., $f_X(X) = f(X)$. The expectation of a function $g(X)$ taken over a distribution $f(X)$ is defined as

$$E_{f(X)}[g(X)] \equiv \int g(X)f(X)dX.$$

I use $E_X[g(X)]$ (the expectation of $g(X)$ taken over a distribution of X) or $E[g(X)]$ (the expectation of $g(X)$ taken over a distribution) when the distribution taking over the expectation is clear from the context. Similarly, let $\mathrm{Var}_{f(X)}[\cdot]$ indicate that the expectation in the variance formula is taken over a distribution $f(X)$, and $\mathrm{Var}_X[\cdot]$ and $\mathrm{Var}[\cdot]$ be the short form of it.

The model of learning or estimating the underlying function of an outcome from a sample dataset, which is drawn from the target population, is described by five components. At first, a set of random vectors $X \in \mathcal{X} \subset \mathbb{R}^p$ which are drawn independently from the p-dimensional input distribution of the target population, $f(X)$. I implicitly include a constant as the first element of X for notational simplicity.

Second, a *supervisor* that returns an output $Y \in \mathcal{Y} \subset \mathbb{R}$ for every input vector X according to the conditional distribution of the output of the target population $f(Y|X)$. Whether the output type is continuous or discrete affects the representation of the function which we seek to learn. This distinction in output type has led to a naming convention for the prediction tasks: *regression* when we predict continuous outputs, and *classification* when we predict discrete outputs. For K-class classification ($K \geq 2$), I use a set $\{0, 1, \ldots, k, \ldots, K-1\}$ as a notation for K classes except when stated otherwise.

Third, a sample dataset $\mathcal{T} \equiv \{(x_1, y_1), \ldots, (x_N, y_N)\}$ which is assumed to be N independent identically distributed random observations drawn from the joint distribution of X and Y, $f(X, Y) = f(X)f(Y|X)$.

Fourth, a *learning machine* which is capable of implementing a set of candidate functions. Here, it is convenient to consider regression and classification separately. For regression, we want to find a function $a : \mathcal{X} \to \mathcal{Y}$ such that $a(X)$ approximates Y. Thus, a set of candidate functions suitable for regression is $\{a(X; \theta) : \theta \in \Theta\}$, where functions are characterized by a vector of parameters θ in a parameter space Θ. For instance, candidate functions can be specified as a simple linear in parameters model with an arbitrary p-dimensional parameter space: $\{X^T\theta : \theta \in \Theta \subset \mathbb{R}^p\}$.

By contrast, for K-class classification, although $a(X)$ is an interest as well, frequently, a function $b : \mathcal{X} \to \mathbb{R}^K$ such that $b(X) = (b_0(X), b_1(X), \ldots, b_k(X), \ldots, b_{K-1}(X))^T$ is a vector of scores of the propensity for the assignment to classes attracts more attention than it. The domain of the function is typically a subset of $\mathbb{R}^K$. For example, researchers often seek $b : \mathcal{X} \to [0, 1]^K$ such that $\sum_{k=0}^{K-1} b_k(X) = 1$ and $b_k(X)$ approximates the conditional probability of the assignment to class k, $\Pr(Y = k|X)$, because it is convenient for the interpretation of the results.

Another particular example is a prediction function for two-class classification. With only two classes, a score of the propensity for the assignment to either of the classes is sufficient for the prediction purpose. Consequently, one of the elements of $b(X)$, say, $b_1 : \mathcal{X} \to \mathbb{R}$ such that $b_1(X)$ is a score of the propensity for the assignment to class 1, is often a primary interest of researchers.

Therefore, a set of candidate functions suitable for classification can be either of $\{a(X; \theta) : \theta \in \Theta\}$ or $\{b(X; \theta) : \theta \in \Theta\}$. Note that even when the objective of classification is to find $a(X)$, this is usually carried out taking

$$a(X) = \arg\max_k b_k(X)$$

after the estimation of $b(X)$. For brevity of explanation, the response functions $a(X; \theta)$ and $b(X; \theta)$ will be collectively referred to as $\Psi_\theta(X)$ when the distinction between them is unneces-

sary.

The final component is a measure of the loss or discrepancy, a loss function $L(Y, \Psi_\theta(X))$, between the response Y of the supervisor and the response $\Psi_\theta(X)$ provided by the learning machine for a given X.

The problem of learning is that of choosing the function $\Psi_\theta(X)$ which is the best available approximation to the supervisor's response in terms of the loss function $L(Y, \Psi_\theta(X))$ from the given set of functions $\{\Psi_\theta; \theta \in \Theta\}$ based on the sample dataset $\mathcal{T}$. Consider the expected value of the loss, given by the *risk functional* (Vapnik 1999):

$$R(\theta) \equiv E_{X,Y}[L(Y, \Psi_\theta(X))].$$

The popular loss function is a squared error loss for regression and a negative log-likelihood or *log-loss* for classification. For sample (x_i, y_i), the squared error loss is defined as

$$L(y_i, a(x_i; \theta)) = \{y_i - a(x_i; \theta)\}^2,$$

and the log-loss as

$$L(y_i, b(x_i; \theta)) = -\log b_{y_i}(x_i; \theta),$$

where the log-loss implicitly assumes $\sum_{k=0}^{K-1} b_k(X) = 1$. The risk functional with the squared error loss is called *expected prediction error* (EPE):

$$\text{EPE}(\theta) \equiv E_{X,Y}[\{Y - a(X; \theta)\}^2].$$

Now, the goal of the statistical learning can be summarized as to find the function $\Psi_{\theta_0}, \theta_0 \in \Theta$ which minimizes the risk functional $R(\theta)$ over the class of functions $\{\Psi_\theta : \theta \in \Theta\}$ when the only available information is the sample dataset $\mathcal{T}$.

Regression methods can be interpreted as methods of finding θ that minimize the empirical risk functional $R(\theta)$:

$$\underset{\theta}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, \Psi_\theta(x_i)).$$

Linear regression or ordinary least squares (OLS) specifies the output function $a(X;\theta)$ as linear in parameters and the loss function as squared error loss:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i^T\theta)^2.$$

Binary logistic regression specifies the output function $b(X;\theta)$ in the logit form and the loss function as the log-loss:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\{-\log b_{y_i}(x_i;\theta)\},$$

where

$$b_{y_i}(x_i;\theta) = \frac{y_i \exp x_i^T\theta}{1+\exp x_i^T\theta} + \frac{1-y_i}{1+\exp x_i^T\theta}.$$

Regression methods are known to provide $\sqrt{N}$-consistent estimators under certain regularity conditions. There are many admirable textbook treatments with regard to regression methods and asymmptotic analysis of them (e.g., Wooldridge (2010) and Greene (2012)).

In the case of a fixed number of inputs p with sample size $N \to \infty$ or sufficiently large relative to p, regression methods work perfectly. Regression methods provide a consistent estimator with a reasonable sampling variance, and consequently, provide a useful predictive value at target value $X = x_0$ by just plugging in the values in the estimated function. However, when p is large relative to N including the case of $p > N$, it is well known that regression methods often work poorly in the interpretation of the estimated parameters and the accuracy of prediction on future data (Zou and Hastie 2005). Since the regularity condition no longer holds when $p > N$, regression methods fail to maintain the consistency, and hence, any interpretation is left on the estimator. Moreover, even if $p \leq N$, one can hardly acquire an adequately small sampling variance to gain some meaningful insights from the results when p is large relative to N.

In contrast to regression methods, statistical learning methods pursue to minimize the risk functional $R(\theta)$ more directly using the *hyperparameter* rather than relying on the sample analogue of the risk functional. The hyperparameter aids the model to attain the minimum risk functional in the following way. Statistical learning methods randomly divide the sample dataset into two parts: a *training set* and a *validation set*. For each candidate value of the

hyperparameter, an estimation of the parameter of the model is conducted with the training set, and an estimator for the risk functional of the estimated model is computed by the average loss of the model in the validation set. The hyperparameter is subsequently tuned to be the value that minimizes the estimator of the risk functional.

A method called *cross-validation* (CV) is also used to estimate the risk functional. Although CV is computationally much harder than the calculation of the average loss in the validation set, CV is a more efficient way of estimating the risk functional.

For two-class classification, Bradley (1997) proposed a method based on one minus the area under the receiver operating characteristic curve (AUC) instead of the risk functional for the hyperparameter tuning. As the calculation of the AUC only requires a ranked list of samples of their propensity for the assignment to classes, the method is robust to the monotonic transformation of the functional form of the prediction function. Although the one minus AUC approach is not covered by the risk functional approach of Vapnik (1999), the notion of the risk functional approach is later refined to include such approach (Chen et al. 2009). There, the risk functional approach of Vapnik (1999) is subsumed as the *pointwise approach*, and the one minus AUC approach is categorized as the *listwise approach*. However, I continue to assume the word "risk functional" to mean the risk functional of Vapnik (1999) because the formal definition of the refinement involves complicated and confusing notions regarding the distribution of the expected value of the loss to be taken. All of the following statement regarding the risk functional can be extended to the one minus AUC approach without any modification.

A typical hyperparameter of statistical learning methods is the coefficient for the *regularization* term. Suppose there are a dataset generated from an underlying function with some form of error and a set of functions that are candidates for the best underlying function approximation. The regularization principle, which is first introduced by Tikhonov (1963), imposes some form of smoothness constraints on the candidate functions to find the best approximating function given the dataset. Adapting the regularization technique to the statistical learning setting yields a general class of regularization problems:

$$\min_{\theta} \left\{ \sum_{i=1}^{N} L(y_i, \Psi_\theta(x_i)) + \lambda J(\theta) \right\}, \tag{1}$$

where $\lambda \geq 0$ is a regularization coefficient and $J(\cdot)$ is the regularization term. Commonly used regularization term is an *L_2-penalty*

$$J(\theta) = \|\theta\|_2^2 = \sum_{j=1}^{p} \theta_j^2,$$

and an *L_1-penalty*

$$J(\theta) = \|\theta\|_1 = \sum_{j=1}^{p} |\theta_j|.$$

This formulation emcompasses most of statistical learning methods covered here.

Although it is better to tune the hyperparameter to minimize the estimator of the risk functional, it is often prespecified based on existing knowledge to avoid high computational burden. As which of the whole sample dataset and the training set should be used for estimation of parameters depends on whether the hyperparameter is prespecified or not, I do not distinguish between the sample dataset and the training set in the following exposition of the estimation detail of statistical learning methods. There, I use the word "dataset" with a standard sample size notation N.

The distinction between the regression and the statistical learning is further clarified through the following example. Suppose the sample dataset $\mathcal{T}$ arises from a linear model

$$Y = X^T\theta_0 + \epsilon \tag{2}$$

with $\theta_0 \in \Theta \subset \mathbb{R}^p$, $E(\epsilon|X) = 0$, and $\mathrm{Var}(\epsilon|X) = \sigma^2 < \infty$. Let the estimator from a linear regression be $\hat{\theta}$. Then, the EPE at the target input x_0 can be decomposed into three elements:

$$\begin{aligned}
\mathrm{EPE}(\theta|X = x_0) &= E_{\mathcal{T}}\left[E_{Y|X=x_0}[\{Y - x_0^T\hat{\theta}\}^2|X = x_0]\right] \\
&= E_{Y|X=x_0}[\{Y - x_0^T\theta_0\}^2|X = x_0] \\
&\quad +\{x_0^T\theta_0 - E_{\mathcal{T}}[x_0^T\hat{\theta}]\}^2 \\
&\quad +E_{\mathcal{T}}[\{E_{\mathcal{T}}[x_0^T\hat{\theta}] - x_0^T\hat{\theta}\}^2] \\
&= \sigma^2 + \mathrm{Bias}^2[x_0^T\hat{\theta}] + \mathrm{Var}_{\mathcal{T}}[x_0^T\hat{\theta}],
\end{aligned}$$

where $E_{\mathcal{T}}[\cdot]$ indicates that the expectation is taken over the sampling distribution of the sample dataset $\mathcal{T}$. Since the first term σ^2 cannot be reduced by devising the estimation method, minimizing the EPE is the same as minimizing the sum of squared bias and the sampling variance. Under the linear model (2), the unbiasedness of OLS assures the unbiasedness of the resulting prediction as well:

$$E_{\mathcal{T}}[x_0^T\hat{\theta}] = x_0^T E_{\mathcal{T}}[\hat{\theta}] = x_0^T\theta_0 \Leftrightarrow \text{Bias}[x_0^T\hat{\theta}] = 0.$$

Thus, we now know that the linear regression produces a least bias estimator for the prediction. But still, some methods may achieve their better prediction performance through a bias-variance trade-off. Statistical learning methods are such methods that aim to minimize EPE by the reduction of sampling variance along with paying the cost of some bias.

The remaining of this section is largely based on Hastie, Tibshirani, and Friedman (2009) and overviews popular statistical learning methods: (1) Discriminant analysis; (2) Generalized additive model; (3) k-nearest neighbor; (4) Support vector machine; (5) Penalized regression; (6) Tree-based model; (7) Neural network.

2 Discriminant analysis

The *linear discriminant analysis* (LDA), which is originally proposed by Fisher (1936), aims to discriminate between two or more classes with a linear discriminant function that maximizes the ratio of the between-class variance to the within-class variance. This subsection only deals with K-class classification, on which the discriminant analysis mainly focuses.

Denote the between-class variance covariance matrix (VCM) of the input vector X as $B = \text{Var}_Y(E[X|Y])$ and the class k's within-class VCM of X as $W_k = \text{Var}(X|Y = k)$. Now, consider a linear discriminant function $Z = \nu^T X$, $\nu \in \mathbb{R}^p$. The between-class variance of Z is

$$\text{Var}_Y(E[Z|Y]) = \text{Var}_Y(\nu^T E[X|Y]) = \nu^T \text{Var}_Y(E[X|Y])\nu = \nu^T B\nu,$$

and the class k's within-class VCM of Z is

$$\text{Var}(Z|Y=k) = \nu^T \text{Var}(X|Y=k)\nu = \nu^T \text{Var}(X|Y=k)\nu = \nu^T W_k \nu.$$

The LDA assumes a common VCM, W, for all classes, $\forall k$, $W_k = W$.

Then, the Fisher's problem amounts to finding ν that maximizes the ratio of the between-class variance of Z to the within-class variance of Z:

$$\arg\max_{\nu} \frac{\nu^T B \nu}{\nu^T W \nu}.$$

The solution, ν_1, is shown to be $W^{-1/2}$ multiplied by the eigenvector of the largest eigenvalue of $W^{-1/2}BW^{-1/2}$. Similarly one can find the second best linear discriminant function $\nu_2^T X$ by finding ν orthogonal to ν_1 that maximizes $\nu^T B\nu / \nu^T W \nu$: the solution ν_2 is $W^{-1/2}$ multiplied by the eigenvector of the second largest eigenvalue of $W^{-1/2}BW^{-1/2}$. And this procedure can be continued L times to find a sequence of discriminant coordinates $\{\nu_l\}_{l=1}^{L}$. As we need at most $K-1$ discriminant functions to separate input space into K classes, $L \leq K-1$. In practice, discriminant coordinates are estimated using the estimator of B and W, e.g.,

$$\hat{B} = \frac{1}{N}\sum_{k=0}^{K-1} |\mathcal{S}_k|(\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T \quad \text{and} \quad \hat{W} = \frac{1}{N}\sum_{k=0}^{K-1}\sum_{i\in\mathcal{S}_k}(x_i - \bar{x}_k)(x_i - \bar{x}_k)^T,$$

where $\mathcal{S}_k = \{i : y_i = k\}$, $|\mathcal{S}_k|$ is the cardinality of $\mathcal{S}_k$, $\bar{x}_k = \sum_{i\in\mathcal{S}_k} x_i/|\mathcal{S}_k|$, and $\bar{x} = \sum_{i=1}^{N} x_i/N$.

As L is at most $K-1$, the LDA usually achieves a considerable dimension reduction compared to the p-dimensional input space and may help us to understand informative attributes of the data. Besides, the discriminant functions can be interpreted as a score of the propensity for a certain class assignment and this interpretation leads to a further generalization of the LDA.

Consider an optimal scoring problem that transforms class labels to scores which are optimally predicted by linear regression on X. Suppose $s : \mathcal{Y} \to \mathbb{R}^L$ is a function that assigns $L \leq K-1$ scores to the classes. Then the optimal scoring problem solves the following minimization problem:

$$\underset{s,\{\beta_l\}_{l=1}^{L}}{\text{argmin}} \sum_{l=1}^{L}\sum_{i=1}^{N}\{s_l(y_i) - x_i^T\beta_l\}^2,$$

where s_l is the lth component of s. It can be shown that the sequence of discriminant coordinates $\{\nu_l\}_{l=1}^{L}$ derived from the Fisher's problem is identical to the sequence $\{\beta_l\}_{l=1}^{L}$ up to a constant (Mardia, Kent, and Bibby 1979).

From this re-formalization of the LDA, one can think of a basis expansion and an application of regularization principle to generalize the LDA:

$$\underset{s,\{\beta_l\}_{l=1}^{L}}{\operatorname{argmin}} \sum_{l=1}^{L} \left[\sum_{i=1}^{N} \{s_l(y_i) - h(x_i)^T \beta_l\}^2 + \lambda J(\beta_l) \right],$$

where $h(\cdot)$ is a flexible transformation function. This generalization of the LDA is proposed by Hastie, Tibshirani, and Buja (1994) as the *flexible discriminant analysis* (FDA), and by Hastie, Buja, and Tibshirani (1995) as the *penalized discriminant analysis* (PDA). The FDA and PDA successfully incorporate nonlinearity while maintaining the dimension reduction aspect of the LDA.

Although the discriminant analysis is good at classification in a moderate number of inputs, they are poor at dealing with sparse high-dimensional inputs (i.e., most of the entries of the inputs are zero for each observation, and the number of the inputs is large relative to the sample size). Because the model underlying the discriminant analysis assumes that the within-class distribution of every class is nondegenerate for all inputs, inputs which have the same value for all samples in some class do not fit for the model. Consequently, computer programs that execute the discriminant analysis usually refuse a dataset with such inputs, and we need to discard those inputs before the analysis if we seek to apply the discriminant analysis to the dataset. One can think of an *unsupervised learning* method that extracts essential components of the inputs, e.g., *principal component analysis* (Mardia, Kent, and Bibby 1979), before the analysis to alleviate the problem. Nevertheless, there is no standard way of an input pre-processing in this direction for the discriminant analysis yet.

3 Generalized additive model

A linear model fails to capture nonlinear effects, and this fact distorts the ability to predict outcomes by it. Although one can think of adding nonlinear terms to the linear model by hand,

one never knows whether the additional terms are sufficient for the model or not. Hastie and Tibshirani (1986) proposed the *generalized additive model* (GAM), in which flexible nonlinear effects can be incorporated automatically.

For regression, a candidate function $a(X;\theta)$ in the GAM is specified as

$$\text{link}[a(X;\theta)] = \sum_{j=1}^{p} \alpha_j(X_j;\theta_j),$$

where $\text{link}[\cdot]$ is a *link function*, X_j is the jth input, and θ_j is the jth set of parameters. When the link is the *identity link*, i.e., $\text{link}[a] = a$, and the loss function is the squared error loss, the model is an extension of the linear regression, and it is called the *additive linear regression model*.

For two-class classification, candidate functions $b(X;\theta)$ in the GAM are specified as

$$\text{link}[b_1(X;\theta)] = \sum_{j=1}^{p} \alpha_j(X_j;\theta_j).$$

This concept can be extended to K-class classification. When the link is the *logit link*, $\text{link}[b] = \text{logit}(b)$, and the loss function is the log-loss, the model is an extension of the logistic regression, and it is called the *additive logistic regression model*.

Each function $\alpha_j(X_j;\theta_j)$ is fitted by a *scatterplot smoother* (e.g., a *cubic smoothing spline* or *kernel smoother*) using the *backfitting algorithm*, where the degrees of freedom for the smoothers is a hyperparameter of the model. A detailed description of estimation methods is covered in Hastie and Tibshirani (1990).

The additive linear regression model with a cubic smoothing spline is shown to be the minimizer of

$$\sum_{i=1}^{N}\{y_i - \sum_{j=1}^{p} \alpha_j(x_{ij};\theta_j)\}^2 + \sum_{j=1}^{p} \lambda_j \int \{\alpha_j''(t_j)\}^2 dt_j,$$

where x_{ij} is the jth input of the ith sample (Hastie, Tibshirani, and Friedman 2009). Thus, the GAM can be interpreted as a regularization problem (1). The GAM is highly flexible in incorporating nonlinearity of a moderate number of inputs, however, the difficulty in the hyperparameter tuning and the need of input pre-processing if the number of inputs is large

narrow the area of suitable application of the model. Although a progress has been made in the smoothing parameter tuning and the automatic variable selection in a sparse high-dimensional setting recently (Lin and Zhang 2006; Ravikumar et al. 2009), these potentially innovative methods are still computationally prohibitive for large scale data.

4 k-nearest neighbor

The *k-nearest neighbor* (kNN) classifier appears as a natural estimator for the *nonparametric discriminatory analysis* (Fix and Hodges 1951). As the kNN is mainly applied to classification, this subsection concentrates on classification. To avoid a notational confusion caused by the usage of the same letter 'k' in K-class classification and the k-nearest neighbor, let K-class be J-class instead in this subsection: $\mathcal{Y} = \{0, 1, \ldots, j, \ldots, J-1\}$. k is a hyperparameter of the model.

Let the distance of a query point x_0 and a sample input x_i can be measured by a designated distance metric $D(x_0, x_i)$. In the kNN, we first find a set of indices of k training samples, $\mathcal{S}_{k(x_0)} \subset \{1, 2, \ldots, N\}$, which are the k closest neighbors to the query point. Then, we predict a class probability of the query point from the frequency of the class of the k-nearest neighbors (*voting* estimator, Fix and Hodges (1951)),

$$b_j(x_0; \theta) = \frac{1}{k} \sum_{i \in \mathcal{S}_{k(x_0)}} I(y_i = j),$$

or the inverse distance weighted frequency of the class of the k-nearest neighbors (*inverse distance weighting* (IDW) estimator, Shepard (1968)),

$$b_j(x_0; \theta) = \frac{\sum_{i \in \mathcal{S}_{k(x_0)}} w(x_0, x_i) I(y_i = j)}{\sum_{i \in \mathcal{S}_{k(x_0)}} w(x_0, x_i)}, \quad \text{where} \quad w(x_0, x_i) = \frac{1}{D(x_0, x_i)}.$$

Typically, the Euclidean (L_2) distance is used as the distance metric, i.e., $D(x_0, x_i) = \|x_0 - x_i\|_2$. Besides, inputs may be better to be standardized to have mean zero and variance one when units of the inputs are different from each other. The distance metric can be a more general distance measure like the *L_p-distance* or the *Mahalanobis distance*, and in the case of

categorical inputs, a distance measure like the *Hamming distance* may be more suitable for a distance metric. Designing the distance metric in the kNN is difficult as an appropriate distance metric depends on the setting.

Although the kNN creates a highly flexible nonparametric estimator for classification, the lack of the interpretability of the method discourages the use of the method in biomedical and clinical research except research related to the field of image recognition, where the kNN had established an era by the invention of the *tangent distance* (Simard, LeCun, and Denker 1992).

5 Support vector machine

The *support vector machine* (SVM) is developed as an extension of the *optimal separating hyperplane* (Boser, Guyon, and Vapnik 1992; Cortes and Vapnik 1995). The idea is to map the inputs into a high-dimensional input space through some prespecified nonlinear mapping and to construct an optimal separating hyperplane in the enlarged space. Although the SVM can be extended to K-class classification or regression, the main focus of the model is on two-class classification, with which I deal here. In the SVM literature, output space is usually defined as $\mathcal{Y} = \{-1, 1\}$, and I follow this convention.

First, consider an optimal separating hyperplane between two perfectly separable classes (Rosenblatt 1958). Define a hyperplane L by $\{X : X^T\theta = 0, \|\theta\|_2 = 1\}$, where $\|\theta\|_2 = 1$ is for a normalization, and a classification rule by

$$a(X; \theta) = \text{sign}[X^T\theta].$$

As the signed distance from a point X to the hyperplane L is $X^T\theta$, the optimal separating hyperplane that creates the biggest margin between the training points for two classes satisfies

$$\underset{\{\theta:\|\theta\|_2=1\}}{\arg\max}\ M \ \text{ s.t. } \forall i,\ \ y_i(x_i^T\theta) \geq M,$$

which is equivalent to

$$\underset{\theta}{\text{argmin}}\ \|\theta\|_2 \ \text{ s.t. } \forall i,\ \ y_i(x_i^T\theta) \geq 1.$$

The latter formulation is more convenient to solve as the norm constraint on θ is dropped and M is eliminated. Now, the margin is formally defined as the area that satisfies $|y_i(x_i^T\theta)| \leq M$ under the normalization constraint $\|\theta\|_2 = 1$ and $|y_i(x_i^T\theta)| \leq 1$ otherwise.

Next, suppose that the classes are linearly nonseparable. The optimal separating hyperplane is redefined as the hyperplane that maximizes the margin, but allows for some points to be on the wrong side of the boundary of the margin subject to a given upper bound of the total proportional amount of the slack, C:

$$\underset{\{\theta:\|\theta\|_2=1\}}{\arg\max}\ M \ \text{ s.t. } \begin{cases} \forall i,\ \ y_i(x_i^T\theta) \geq M(1-\xi_i) \ \text{ and } \ \xi_i \geq 0, \\ \sum_{i=1}^N \xi_i \leq C \end{cases},$$

which is equivalent to

$$\underset{\theta}{\operatorname{argmin}} \|\theta\|_2 \ \text{ s.t. } \begin{cases} \forall i,\ \ y_i(x_i^T\theta) \geq 1-\xi_i \ \text{ and } \ \xi_i \geq 0, \\ \sum_{i=1}^N \xi_i \leq C \end{cases}. \tag{3}$$

Moreover, the solution of (3) can be shown to be equivalent to that of

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^N \max(1 - y_i(x_i^T\theta), 0) + \lambda\|\theta\|_2^2,$$

where λ is the hyperparameter reflecting the cost of the slack. The loss function $L(Y, X^T\theta) = \max(1 - Y(X^T\theta), 0)$ is known as the *hinge loss*. It is clear that the problem also forms a class of regularization problems (1). Given the hyperparameter λ, the solution is

$$\hat{\theta}_\lambda = \frac{1}{2\lambda}\sum_{i=1}^N \hat{\alpha}_i y_i x_i,$$

where

$$\hat{\alpha}_i = \begin{cases} 0 & \text{if } y_i(x_i^T\hat{\theta}_\lambda) > 1 \text{ (samples correctly classified and outside the margin),} \\ [0,1] & \text{if } y_i(x_i^T\hat{\theta}_\lambda) = 1 \text{ (samples sitting on the boundary of the margin),} \\ 1 & \text{if } y_i(x_i^T\hat{\theta}_\lambda) < 1 \text{ (samples inside the margin or wrongly classified).} \end{cases}$$

Therefore, the estimated optimal separating hyperplane at λ can be written as

$$\begin{aligned} & X^T\hat{\theta}_\lambda = 0 \\ \Leftrightarrow \quad & \frac{1}{2\lambda}\sum_{i=1}^{N}\hat{\alpha}_i y_i X^T x_i = 0. \end{aligned}$$

From the formula, you can see that the samples correctly classified and outside the margin do not contribute to the prediction rule.

Finally, to enlarge the input space, substitute the transformed input vectors $h(x_i)$ for the raw input vectors x_i:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \max(1 - y_i(h(x_i)^T\theta), 0) + \lambda\|\theta\|_2^2.$$

Then, the estimated generalized hyperplane at λ is

$$\begin{aligned} & h(X)^T\hat{\theta}_\lambda = 0 \\ \Leftrightarrow \quad & \frac{1}{2\lambda}\sum_{i=1}^{N}\hat{\alpha}_i y_i h(X)^T h(x_i) = 0 \\ \Leftrightarrow \quad & \frac{1}{2\lambda}\sum_{i=1}^{N}\hat{\alpha}_i y_i K(X, x_i) = 0, \end{aligned}$$

where $K(X, X') \equiv h(X)^T h(X')$ is a type of function known as the *kernel function*, and the resulting classification rule is

$$a(X; \hat{\theta}_\lambda) = \text{sign}[\frac{1}{2\lambda}\sum_{i=1}^{N}\hat{\alpha}_i y_i K(X, x_i)].$$

It is known that the sufficient knowledge for the estimation of the SVM is the kernel function, and the input transforming function $h(X)$ is not necessarily required. The popular choices for

the kernel function in the SVM literature are

$$\begin{aligned} \text{Linear: } K(X, X') &= X^T X', \\ d\text{th-degree polynomial: } K(X, X') &= (1 + X^T X')^d, \\ \text{Radial basis: } K(X, X') &= \exp(-\gamma \|X - X'\|_2^2), \\ \text{Sigmoid: } K(X, X') &= \tanh(\kappa_1 X^T X' + \kappa_2). \end{aligned}$$

There are two features of the hinge loss that make the SVM robust to outliers. First, under the hinge loss, samples correctly classified and outside the margin do not contribute to the prediction rule. Second, the hinge loss gives a linear penalty rather than a quadratic penalty to samples inside the margin or wrongly classified. Nonetheless, a squared hinge loss that gives a quadratic penalty to the samples may improve the performance when the effect of the outliers is negligible.

The SVM is extremely computationally intensive for the large sample size. Researchers have developed an efficient algorithm that is computationally feasible for the linear kernel SVM (Rong-En et al. 2008). However, it is still challenging to employ other kernel functions in the large sample size setting.

6 Penalized regression

As a natural consequence of the development of regularization techniques, researchers have applied the regularization principle to the linear regression and created a *penalized least squares* estimator:

$$\underset{\theta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - x_i^T \theta)^2 + \lambda J(\theta) \right\}.$$

Penalized least squares methods using the L_2-penalty and the L_1-penalty as the regularization term are called *ridge regression* (Hoerl and Kennard 1970) and the *lasso* (Tibshirani 1996), respectively.

When the inputs are mutually independent and standardized to have mean zero and variance one, the estimator of the coefficient of the jth input of the ridge $\hat{\theta}_j^{Ridge}$ and that of the lasso $\hat{\theta}_j^{Lasso}$

can be expressed by the corresponding OLS estimator $\hat{\theta}_j^{OLS}$ and the regularization coefficient λ:

$$\begin{cases} \hat{\theta}_j^{Ridge} = \frac{\hat{\theta}_j^{OLS}}{1+\lambda}; \\ \hat{\theta}_j^{Lasso} = \text{sign}(\hat{\theta}_j^{OLS})\{\max(|\hat{\theta}_j^{OLS}| - \lambda, 0)\}. \end{cases}$$

The ridge regression does a proportional shrinkage, while the lasso shifts each OLS estimator by a constant factor λ, truncating at zero. The ridge regression cannot produce a parsimonious model as it keeps all inputs in the model in the same way as the linear regression. On the other hand, the lasso does both continuous shrinkage and automatic variable selection to obtain a parsimonious model. The automatic variable selection is an attractive feature because researchers prefer a simpler model which puts more light on the relationship between the output and inputs.

Although the estimator from the lasso appears to be interpretable like the linear regression, the estimator is neither unbiased nor consistent, and it is not possible to interpret it as the OLS estimator. Therefore, while the regression can make a valid inference for the effect of an input, such inference is not possible in the standard penalized regression framework. Recently, some researchers are pursuing valid inference methods for the effect of input in large p setting based on the penalized regression framework (Belloni, Chernozhukov, and Hansen 2011; Belloni and Chernozhukov 2013; Belloni, Chernozhukov, and Hansen 2014; Raskutti, Wainwright, and Yu 2011).

Zou and Hastie (2005) proposed a compromise between the ridge and the lasso, so called *elastic-net*. The elastic-net uses a linear combination of the L_2-penalty and the L_1-penalty as the regularization term:

$$J(\theta) = \alpha\|\theta\|_2^2 + (1-\alpha)\|\theta\|_1,$$

where $\alpha \in [0, 1]$ is an additional hyperparameter which determines the degree of the compromise. They argue that the prediction performance of the elastic-net is expected to be better than that of the lasso if there is a group of variables among which the pairwise correlations are very high.

A *penalized logistic regression* estimator arises as an extension of penalized least squares

methods to the logistic regression framework:

$$\underset{\theta}{\operatorname{argmin}} \left[\sum_{i=1}^{N} \{-\log b_{y_i}(x_i;\theta)\} + \lambda J(\theta) \right],$$

where

$$b_{y_i}(x_i;\theta) = \frac{y_i \exp x_i^T\theta}{1+\exp x_i^T\theta} + \frac{1-y_i}{1+\exp x_i^T\theta}.$$

The regularization term $J(\cdot)$ can be either of the L_2-penalty (Zhu and Hastie 2004), the L_1-penalty (Shevade and Keerthi 2003), or the elastic-net penalty (Waldron et al. 2011).

7 Tree-based model

Morgan and Sonquist (1963) proposed a simple *tree-based model* that tries to automatically select inputs that are crucial to predict an outcome and flexibly incorporate nonlinearity and interactions of them. The idea is to split the input space into subgroups that can be expressed as a leaf of a decision tree, and then assign a simple predictive value to each subgroup. Subgroups are called *leaves* and *nodes* in the context of the tree-based model.

7.1 Classification and regression tree

A popular estimation method for a simple tree-based model called *classification and regression tree* (CART) is introduced by Breiman et al. (1984). Consider the case of regression first. The input space is split into M subgroups $\{R_m\}_{m=1}^M$ and a constant response c_m is assigned to each subgroup as a predictive value for the subgroup inputs:

$$a(X;\theta) = \sum_{m=1}^{M} c_m I(X \in R_m),$$

where $\theta = \left\{\{c_m\}_{m=1}^M, \{R_m\}_{m=1}^M\right\}$, M is a hyperparameter, and each subgroup R_m is defined by a combination of simple decision rules. For example, a subgroup R_m can be defined as a set of inputs that the jth input X_j is over s and the j'th input $X_{j'}$ is over s': $R_m = \{X : X_j > s \quad \text{and} \quad X_{j'} > s'\}$.

The estimation proceeds as follows. Define a pair of half-spaces into which the input space is divided by a hyperplane $X_j = s$:

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j > s\}.$$

Then we seek a best splitting hyperplane that minimizes the total loss of the dataset given the constant response rule:

$$\underset{j,s}{\operatorname{argmin}} \left\{ \min_{c_1} \sum_{x_i \in R_1(j,s)} L(y_i, c_1) + \min_{c_2} \sum_{x_i \in R_2(j,s)} L(y_i, c_2) \right\}.$$

A typical loss function is the squared error loss for regression. Having found the hyperplane, we partition the dataset into two regions by the hyperplane and repeat this splitting process on each of the two regions. The process is repeated on all of the resulting regions to grow a tree. A tree is stopped to grow when a designated minimum node size (e.g., ten) is reached. The final tree τ_0 is pruned using a cost-complexity criterion to find an optimal tree.

To state the cost-complexity criterion of the pruning procedure, some notations need to be defined: a subtree τ is defined to be any tree that can be obtained by pruning τ_0; let $|\tau|$ denote the number of leaves in the subtree τ; let $\hat{c}_m$ be the solution that minimizes the loss in the mth node:

$$\underset{c_m}{\operatorname{argmin}} \sum_{x_i \in R_m} L(y_i, c_m).$$

The cost-complexity criterion is defined using these notations with the regularization principle (1):

$$C_\lambda(\tau) = \sum_{m=1}^{|\tau|} \sum_{x_i \in R_m} L(y_i, \hat{c}_m) + \lambda |\tau|,$$

where λ is a hyperparameter reflecting the number of subgroups M. Though the loss function in the criterion can be a different loss function from that in the tree growing procedure, usually it is the squared error loss as well. For a given value of λ, one can show that there is a unique subtree τ_λ that minimizes the criterion. In practice, we use the *weakest link pruning* to produce a sequence of subtrees. In the weakest link pruning, we successively collapse the internal node that produces the smallest per-leaf increase in a total loss until a single-node tree is produced.

The subtree that minimizes the criterion in the sequence is known to be the optimal subtree τ_λ.

For K-class classification, the form of candidate functions becomes to

$$b(X;\theta) = \sum_{m=1}^{M} p_m I(X \in R_m),$$

where $\theta = \left\{\{p_m\}_{m=1}^{M}, \{R_m\}_{m=1}^{M}\right\}$ and $p_m = (p_{m0}, p_{m1}, \ldots, p_{mk}, \ldots, p_{mK-1})^T$ is a vector of the conditional probability of the assignment to the K classes in node m. Practically, the estimator of p_{mk}, $\hat{p}_{mk}$, is the proportion of the class k observations in node m. We succesively seek a hyperplane that minimizes the total loss of the dataset given the estimator of p_m, $\hat{p}_m = (\hat{p}_{m0}, \hat{p}_{m1}, \ldots, \hat{p}_{mk}, \ldots, \hat{p}_{mK-1})^T$, in the tree growing procedure. For example, the objective function in the first splitting process is

$$\underset{j,s}{\operatorname{argmin}} \left\{ \sum_{x_i \in R_1(j,s)} L(y_i, \hat{p}_1) + \sum_{x_i \in R_2(j,s)} L(y_i, \hat{p}_2) \right\}.$$

A typical loss function is the log-loss for classification. The pruning procedure is conducted with a loss function that suits classification as well: typically the log-loss again. Consequently, the cost-complexity criterion is slightly changed from that of the procedure for regression,

$$C_\lambda(\tau) = \sum_{m=1}^{|\tau|} \sum_{x_i \in R_m} L(y_i, \hat{p}_m) + \lambda|\tau|,$$

but the remaining part of the procedure is the same.

7.2 Random forest

A problem with the simple tree-based model is their high variance of the estimated prediction function. The hierarchical nature of the procedure (i.e., the effect of an error in the top split is propagated down to all of the splits below it) and the lack of smoothness of the prediction surface cause the high variance. *Bootstrap aggregation* or *bagging* (Breiman 1996) is a technique that reduces the variance of an estimated prediction function. Bagging averages predictions that are estimated over a collection of bootstrap samples which are generated from the dataset. If the

correlation of pairs of bagged predictions is not perfect, the variance of the bagging estimate is guaranteed to be smaller than that of the initial estimate. Bagging introduces randomness to the predictions by the use of bootstrap samples and creates a set of predictions that are not perfectly correlated with each other. As bagging works especially well for high-variance and low-bias estimation methods, the power of the tree-based model is highly enhanced by using it.

The larger the correlation of pairs of bagged trees is, the more the benefit of the aggregation is limited. The *random forest* (Breiman 2001) aims to improve the variance reduction property of bagging by lowering the correlation between the trees. The random forest augments randomness of the trees by randomly selecting $\xi \leq p$ of the inputs as candidates for splitting inputs for each split in the tree growing procedure. The number of inputs selected for each split ξ is a hyperparameter for the random forest. Breiman (2001) recommends the default value of $\lfloor p/3 \rfloor$ for regression and $\lfloor \sqrt{p} \rfloor$ for classification for the choice of ξ.

Although the hyperparameter for the tree size was the number of leaves in the CART, the minimum node size or the depth of the tree is commonly used in the random forest. The hyperparameter is usually not tuned via the risk functional estimation to avoid a high computational burden, and it is preset based on existing knowledge. The results are known to be fairly insensitive to particular choices of the hyperparameter (Segal 2004), and Hastie, Tibshirani, and Friedman (2009) evaluates that tuning it does not worth the cost.

7.3 Importance sampled learning ensemble

The idea in the *gradient boosting machine* (GBM, Friedman (2001)) and the *stochastic gradient boosting machine* (SGBM, Friedman (2002)) is to *de-correlate* the trees more efficiently than bagging. As these methods are subsumed under the framework of the *importance sampled learning ensemble* (ISLE, Friedman and Popescu (2003)), I introduce the framework here. Regression and classification are explained together.

The procedure of the ISLE is two folds: a preparation of a finite dictionary of trees on which the subsequent aggregation is based using the ISLE generator; an aggregation of the trees in light of the regularization principle. Denote a prediction function of a tree with parameter γ as $\Psi^{\tau}_{\gamma}(X)$. The generator is designed to create trees that have a small correlation with each other.

First, let

$$\zeta_0(X) = \begin{cases} \underset{c}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, c) & \text{if regression,} \\ \frac{1}{N}\sum_{i=1}^{N}(I(y_i = 0), I(y_i = 1), \ldots, I(y_i = K-1))^T & \text{if } K\text{-class classification.} \end{cases}$$

Then, iterate the next procedure for $d = 1, \ldots, D$. Draw a subsample of $N\eta$ ($\eta \in (0, 1]$) of the dataset, typically with replacement if $\eta = 1$ and without replacement otherwise. Given $\zeta_{d-1}(X)$ and the subsample, we estimate a parameter γ_d of a tree $\Psi^{\tau}_{\gamma_d}(X)$ that predicts an output $Y - \zeta_{d-1}(X)$ from inputs X. The tree is appended to the dictionary as the dth basis function. At the end of the round, update $\zeta_d(X)$ in the following manner:

$$\zeta_d(X) = \zeta_{d-1}(X) + \nu\Psi^{\tau}_{\gamma_d}(X),$$

where ν is a hyperparameter for the *learning rate* of the ISLE generator.

ISLE generator efficiently assembles sufficiently different or de-correlated basis trees via the randomness of subsampling and the learning of $\zeta_d(X)$. The more the subsamples are different from each other, the more the correlation among simple trees is reduced, and the larger the learning rate, the more the procedure avoids a tree similar to those found before.

The aggregation of the trees in the dictionary is accomplished by the regularization technique (1):

$$\min_{\{\beta_d\}_{d=0}^{D}} \sum_{i=1}^{N} L[y_i, \beta_0 + \sum_{d=1}^{D} \beta_d \Psi^{\tau}_{\gamma_d}(X)] + \lambda J(\beta).$$

The aggregation technique is called the *post-processing*, and Friedman and Popescu (2003) recommends the use of the L_1-penalty in it. Finally, we get a prediction function $\Psi_\theta(X)$ in the following form:

$$\Psi_\theta(X) = \beta_0 + \sum_{d=1}^{D} \beta_d \Psi^{\tau}_{\gamma_d}(X),$$

where $\theta = \{\{\beta_d\}_{d=0}^{D}, \{\gamma_d\}_{d=1}^{D}\}$.

There are five hyperparameters in the ISLE: a hyperparameter for the tree size, the number of trees to be bagged D, the subsampling ratio for each tree η, the learning rate ν, and the regularization coefficient for the post-processing λ. The tree size is often calibrated by the depth

of the tree. Hastie, Tibshirani, and Friedman (2009) suggests the depth to be six and states that the tuning of the hyperparameter for the depth seldom provides a significant improvement over just using six as the value of it. The number of trees to be bagged D is usually chosen to minimize the estimator of the risk functional.

The basis function generating process of the ISLE is identical to that of the GBM and that of the SGBM if the subsampling ratio η is one and otherwise, respectively. The difference between the ISLE and the boosting machines are in the way that they assemble the basis functions. The ISLE uses the regularization principle more directly than the boosting machines to overcome the disadvantage that they overfit with a growing number of trees to be bagged. Friedman (2002) recommends to set the subsampling ratio to be no more than a half and to be smaller for large sample size: $\eta \leq 1/2$ and $\eta \sim 1/\sqrt{N}$.

The learning rate smaller than one provides regularization through shrinkage, and the recommended value is 0.05 and 0.1 for the GBM and the SGBM, respectively (Friedman 2001, 2002). Lastly, the regularization coefficient for the post-processing λ is chosen to minimize the estimator of the risk functional.

8 Neural network

The *neural network* was first developed as a model for the human brain (McCulloch and Pitts 1943) and became famous as the basis of deep learning (Hinton, Osindero, and Teh 2006). The modern formulation of a neural network with a *back-propagation algorithm* made the neural network to be a computationally feasible method (Rumelhart, Hinton, and Williams 1986a, 1986b). The neural network attempts to capture nonlinear and interactive effects of inputs on output through the *hidden layer*. As the parameter and the hyperparameter of the model become highly complicated with multiple hidden layers, I concentrate on a single hidden layer neural network here.

A hidden layer consists of derived features or hidden units $Z = (Z_1, Z_2, \ldots, Z_M)$, where M is a hyperparameter of the model. A derived feature is created from a linear combination of the inputs and then the target function $\Psi_\theta(X)$ is modeled as a function of a linear combination of

the derived features: for regression, the model is represented as

$$\begin{cases} a(X;\theta) = \rho(Z^T\beta); \\ Z_m = \sigma(X^T\alpha_m) \text{ for } m = 1,\dots,M, \end{cases}$$

where $\theta = \{\{\alpha_m\}_{m=1}^M, \beta\}$, $\rho(\cdot)$ is an *output function*, and $\sigma(\cdot)$ is an *activation function*; for K-class classification,

$$\begin{cases} b_k(X;\theta) = \rho_k(V_0, V_1, \dots, V_{K-1}) \text{ for } k = 0,\dots,K-1; \\ V_k = Z^T\beta_k \text{ for } k = 0,\dots,K-1; \\ Z_m = \sigma(X^T\alpha_m) \text{ for } m = 1,\dots,M, \end{cases}$$

where $\theta = \{\{\alpha_m\}_{m=1}^M, \{\beta_k\}_{k=0}^{K-1}\}$, and $\{\rho_k(\cdot)\}_{k=0}^{K-1}$ are output functions.

The output function is usually the *identity function* for regression, $\rho(Z^T\beta) = Z^T\beta$; and the *softmax function* for classification, $\rho_k(V) = e^{V_k}/\sum_{l=0}^{K-1} e^{V_l}$. The activation function is usually chosen to be the *sigmoid*, $\sigma(X^T\alpha_m) = 1/(1+e^{-X^T\alpha_m})$.

The parameter θ is called *weights* in the context of the neural network, and we seek values for them that make the model fit the dataset well. Generally, neural networks have too many weights and overfit the data at the minimum of the empirical risk functional. Therefore, we apply regularization principle to the weight estimation problem as well. The objective function for the estimation of weights is in the form of the regularization problem (1). Typically, the loss function is the squared error loss and the log-loss for regression and classification, respectively, and the regularization is achieved via the L_2-penalty. The regularization technique of using the L_2-penalty in the weight estimation is called *weight decay*.

The number of hidden units M is typically in the range of $[5, 100]$, and the number tends to be large if the number of inputs and the sample size is large. Although one can search for the optimal number to minimize the estimator of the risk functional, it is most common to set a reasonably large number of hidden units and shrink weights with an appropriate regularization technique.

Note that as the neural network model is generally overparameterized, the identification of

the weights is poor. This feature hampers the interpretability of the model and makes the model difficult to apply to biomedical and clinical research except research related to image recognition. Therefore, the neural network is predominantly used for image recognition problems and mostly developing as a method of image recognition (e.g., deep learning models). Using multi-hidden layers with constraints such as *local connectivity* and *weight sharing* on the network, which allow for more complex connectivity but fewer weights, improved the performance of the neural network dramatically in the field of image recognition (LeCun 1989; LeCun et al. 1998). Nevertheless, a multiple hidden layer neural network that suits the situation other than image recognition is not yet sophisticated enough.

References

Belloni, Alexandre, and Victor Chernozhukov. 2013. "Least squares after model selection in high-dimensional sparse models." *Bernoulli* 19 (2): 521–547.

Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen. 2011. "Inference for High-Dimensional Sparse Econometric Models": 1–41. arXiv: `1201.0220`.

Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen. 2014. "Inference on Treatment Effects after Selection among High-Dimensional Controls." *The Review of Economic Studies* 81 (2): 608–650.

Boser, Bernhard E., Isabelle M Guyon, and Vladimir N. Vapnik. 1992. "A training algorithm for optimal margin classifiers." In *Proceedings of the fifth annual workshop on Computational learning theory,* 144–152. New York: ACM Press.

Bradley, Andrew P. 1997. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern Recognition* 30 (7): 1145–1159.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24:123–140.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (5): 5–32.

Breiman, Leo, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and regression trees.* London: Chapman & Hall/CRC.

Chen, Wei, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. 2009. "Ranking Measures and Loss Functions in Learning to Rank." In *NIPS '07: Proceedings of the 22nd International Conference on Neural Information Processing Systems,* 315–323. March.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-vector networks." *Machine Learning* 20 (3): 273–297.

Fisher, Ronald A. 1936. "The use of multiple measurements in taxonomic problems." *Annals of Eugenics* 7:179–188.

Fix, Evelyn, and J.L. Hodges. 1951. *Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties.* Technical report. U.S. Air Force, School of Aviation Medicine, Randolph Field.

Friedman, Jerome H. 2001. "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics* 29 (5): 1189–1232.

Friedman, Jerome H. 2002. "Stochastic gradient boosting." *Computational Statistics & Data Analysis* 38 (4): 367–378.

Friedman, Jerome H., and Bogdan E. Popescu. 2003. *Importance Sampled Learning Ensembles.* Technical report. Department of Statistics, Stanford University.

Greene, William H. 2012. *Econometric analysis.* 7th. Upper Saddle River: Prentice Hall.

Hastie, Trevor, Andreas Buja, and Robert Tibshirani. 1995. "Penalized Discriminant Analysis." *The Annals of Statistics* 23 (1): 73–102.

Hastie, Trevor, and Robert Tibshirani. 1986. "Generalized Additive Models." *Statistical Science* 1 (3): 297–310.

Hastie, Trevor, and Robert Tibshirani. 1990. *Generalized Additive Models.* London: Chapman & Hall/CRC.

Hastie, Trevor, Robert Tibshirani, and Andreas Buja. 1994. "Flexible Discriminant Analysis by Optimal Scoring." *Journal of the American Statistical Association* 89 (428): 1255–1270.

Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd. 745. New York: Springer.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. 2006. "A Fast Learning Algorithm for Deep Belief Nets." *Neural Computation* 18 (7): 1527–1554.

Hoerl, Arthur E., and Robert W. Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 12 (1): 55.

LeCun, Y. 1989. *Generalization and Network Design Strategies.* Technical report. Department of Computer Science, Univ. of Toronto.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86 (11): 2278–2324.

Lin, Yi, and Hao Helen Zhang. 2006. "Component selection and smoothing in multivariate nonparametric regression." *Annals of Statistics* 34 (5): 2272–2297.

Mardia, Kanti V., John T. Kent, and John M. Bibby. 1979. *Multivariate Analysis.* New York: Academic Press.

McCulloch, Warren S., and Walter Pitts. 1943. "A logical calculus of the ideas immanent in nervous activity." *The Bulletin of Mathematical Biophysics* 5 (4): 115–133.

Morgan, James N., and John A. Sonquist. 1963. "Problems in the Analysis of Survey Data, and a Proposal." *Journal of the American Statistical Association* 58 (302): 415–434.

Raskutti, Garvesh, Martin J. Wainwright, and Bin Yu. 2011. "Minimax Rates of Estimation for High-Dimensional Linear Regression Over Lq-Balls." *IEEE Transactions on Information Theory* 57 (10): 6976–6994.

Ravikumar, Pradeep, John Lafferty, Han Liu, and Larry Wasserman. 2009. "Sparse additive models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71 (5): 1009–1030.

Rong-En, Fan, Chang Kai-Wei, Hsieh Cho-Jui, Wang Xiang-Rui, and Lin Chih-Jen. 2008. "LIBLINEAR: A Library for Large Linear Classification." *Journal of Machine Learning Research* 9:1871–1874.

Rosenblatt, F. 1958. "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review* 65 (6): 386–408.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986a. "Learning Internal Representations by Error Propagation." Chap. 8 in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* edited by David E. Rumelhart and James L. McClelland, 1:318–362. Cambridge: MIT Press.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986b. "Learning representations by back-propagating errors." *Nature* 323 (6088): 533–536.

Segal, Mark R. 2004. "Machine learning benchmarks and random forest regression." *UCSF: Center for Bioinformatics and Molecular Biostatistics.*

Shepard, Donald. 1968. "A two-dimensional interpolation function for irregularly-spaced data." In *Proceedings of the 23rd ACM National Conference,* 517–524. New York, USA: ACM.

Shevade, S. K., and S. S. Keerthi. 2003. "A simple and efficient algorithm for gene selection using sparse logistic regression." *Bioinformatics* 19 (17): 2246–2253.

Simard, Patrice, Yann LeCun, and John S. Denker. 1992. "Efficient pattern recognition using a new transformation distance." In *NIPS'92: Proceedings of the 5th International Conference on Neural Information Processing Systems,* 50–58.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1): 267–288.

Tikhonov, Andrey N. 1963. "Solution of incorrectly formulated problems and the regularization method." *Soviet Math. Dokl.* 4:1035–1038.

Vapnik, Vladimir N. 1999. "An Overview of Statistical Learning Theory." *IEEE Transactions on Neural Networks* 10 (5): 988–999.

Waldron, Levi, Melania Pintilie, Ming-sound Tsao, Frances A Shepherd, Curtis Huttenhower, and Igor Jurisica. 2011. "Optimized application of penalized regression methods to diverse genomic data." *Bioinformatics* 27 (24): 3399–3406.

Wooldridge, Jeffrey M. 2010. *Econometric analysis of cross section and panel data.* 2nd. Cambridge: MIT Press.

Zhu, JI, and Trevor Hastie. 2004. "Classification of gene microarrays by penalized logistic regression." *Biostatistics* 5 (3): 427–443.

Zou, Hui, and Trevor Hastie. 2005. "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2): 301–320.