2. Training a Deep Neural Network

Recall our previous discussion that posed training a NN as an optimization problem. One seeks to find the weights $\boldsymbol{W}$ (and the intercepts $\boldsymbol{b}$) that minimimize a loss function between the output $\hat{Y}_i$ and the training sample Y_i.

A common loss function is quadratic loss, $L(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \sum (Y_i - \hat{Y}_i)^2$, but we shall see that it is advantageous to penalize this.

Assume the input vector $\boldsymbol{x}_i$ is d-dimensional, and that there are K nodes in the network. Thus $\boldsymbol{W}$ is a $K \times d$ matrix and $\boldsymbol{b}$ is vector of length K.

The architecture may be whatever we sensibly choose, and the activation functions are also quite flexible (but ReLU is the standard choice: it is almost linear, almost continuously differentiable, and it is often quick to train).

Recall our notation. At each node, the activation function $\sigma(\cdot)$ takes its inputs (initially a vector $\boldsymbol{x}_i$ for the ith observation, but in subsequent layers these are inputs from previous layers) and outputs $\sigma(\boldsymbol{w}_k' \boldsymbol{x}_{ik} + b_k)$ for $k = 1, \ldots, K$.

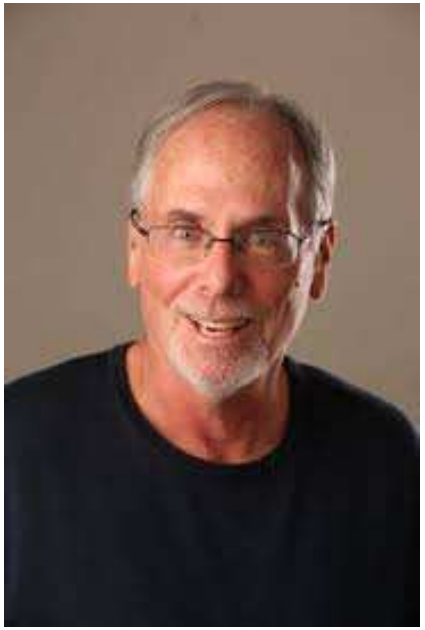
The activation function need not be monotone, but it is better if it is. During the training, backpropagation tells each node how much it should influence nodes in the next layer. If the activation function is not monotone, then increasing a node's weight can cause it to have less influence, which is the opposite of what is wanted. The result is chaotic training behavior, with the network being unlikely to converge to accuracy.

A key concept in effective training is regularization, which is closely related to parsimony and sparsity. One wants to have the simplest possible model that is adequate to one's purpose. This often implies that the model is parsimonious (containing only few terms) and this may be achieved by regularization (e.g., by forcing nodes with small weights to zero).

Sparsity is essentially Ockham's Razor, and is a key idea in all the inferential paradigms. It takes many forms.

- most of the explanatory variables in $p >> n$ regressions are irrelevant,
- a nonnegative matrix can be factored as a product of two matrices, each with low rank,
- the weights in a neural network take only a small number of distinct values,
- for an appropriate set of basis elements, such as wavelets or curvelets or treelets, the signal is approximately a linear combination of a small number of terms,
- a small number of latent factors explain most of the observed variation.

Alan Gelfand is an old-school Bayesian. When he analyzes data, he builds a model for the mechanism that generated the data, places a subjective prior over everything he doesn't know, and finds the posterior.



David Dunson is a new-school Bayesian. His prior is that the model is sparse. And this is often the only way forward in complex problems. This year he is leading the SAMSI program on Deep Learning, and he is focusing on Bayesian regularization.

Data science tries to find hidden structure in large, high-dimensional datasets. But as the Gelfand/Dunson distinction suggests, there can be significant variance in the interpretability of the results. Statisticians tend to favor interpretation, whereas computer scientists often prefer black box models with good accuracy and broad applicability.

As we saw in the first lecture, Deep learning can be used in applications, such as sketch-to-video animation, natural language processing, and control of dynamical systems. But, for exposition, it is helpful to focus upon two broad categories of activity: nonparametric regression and classification.

- Regression models, as used for climate change, wine price, cost of software development.
- Classification models, as used for fraudulent credit card transactions or credit risk assessment.

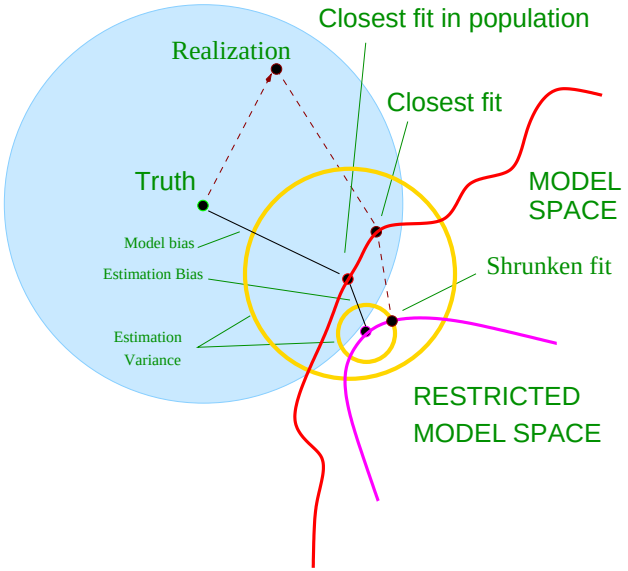


Figure 7.2: *Schematic of the behavior of bias and variance. The model space is the set of all possible predictions from the model, with the "closest fit" labeled with a black dot. The model bias from the truth is shown, along with the variance, indicated by the large yellow circle centered at the black dot labelled "closest fit in population". A shrunken or regularized fit is also shown, having additional estimation bias, but smaller prediction error due to its decreased variance.*

2.1 DL Regression

Nonparametric regression is a natural way to introduce the ideas of regularization and generalizability. The core model is

$$y = f(\boldsymbol{x}) + \epsilon$$

where $\boldsymbol{x} \in \mathbb{R}^d$ and f is unknown. The emphasis is upon estimating the function f.

In real problems one observes $\{y_i, \boldsymbol{x}_i\}$, $i = 1, \ldots, n$. We assume that:

- the $\boldsymbol{x}_i$ are measured without error
- the ϵ_i are i.i.d. with mean zero
- the variance of the ϵ_i values is an unknown constant σ.

These are the minor assumptions, and can be weakened in customary ways.

The main assumption regards the class of functions to which f belongs. Common assumptions include:

- f is in a Sobolev space (essentially, these are functions with bounded derivatives)
- f has a bounded number of discontinuities.

For now, we require only that f be vaguely smooth.

The problem of estimating f becomes vastly harder as d, the dimension of $\boldsymbol{x}$, increases. This is called the Curse of Dimensionality (COD). The term was coined by Richard Bellman in the context of approximation theory (*Adaptive Control Processes*, 1961, Princeton University Press).

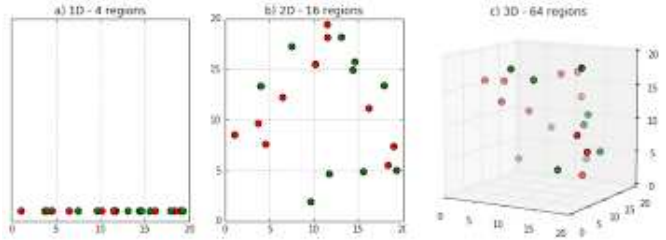
In order to minimize or evade the COD, data scientists have invented many computer-intensive techniques. Some of these include: CART, Projection Pursuit Regression, Random Forests, Support Vector Machines, and Neural Networks.

The COD applies to all multivariate analyses that choose not to impose strong modeling assumptions (e.g., that the relationship between $\boldsymbol{x}$ and $\mathbb{E}[Y]$ is linear, or that f belongs to a particular parametric family of curves).

Although we use regression as our example, the COD applies with equal force to classification (also to cluster analysis and multidimensional scaling).

In terms of the sample size n and dimension d, the COD has three nearly equivalent descriptions:

- For fixed n, as d increases, the sample size becomes too small.
- As d increases, the number of possible models explodes.
- For large d, most datasets are multicollinear (or concurve, which is a nonparametric generalization of multicollinearity).



This means that for large d, the amount of local information that is available to fit bumps and wiggles in f is too small.

To explain the model explosion description of the COD, suppose we restrict attention to just linear models of degree 2 or fewer. For $d = 1$ these are:

$$\mathbb{E}[Y] = \beta_0 \qquad \mathbb{E}[Y] = \beta_1 x_1 \qquad \mathbb{E}[Y] = \beta_2 x_1^2$$
$$\mathbb{E}[Y] = \beta_0 + \beta_1 x_1 \qquad \mathbb{E}[Y] = \beta_0 + \beta_2 x_1^2 \qquad \mathbb{E}[Y] = \beta_1 x_1 + \beta_2 x_1^2$$
$$\mathbb{E}[Y] = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$$

For $d = 2$ this set is extended to include expressions with the terms $\alpha_1 x_2$, $\alpha_2 x_2^2$, and $\gamma_{12} x_1 x_2$.

For general d, combinatorics shows that the number of possible models is

$$2^{\left[1+2d+\left(\begin{array}{c} d \\ 2 \end{array}\right)\right]} - 1.$$

This increases superexponentially in d, and there is not enough sample to enable the data to discriminate among these models.

To explain the multicollinearity description of the COD, recall that multicollinearity occurs when the explanatory values concentrate on an affine subspace in $\mathbb{R}^d$. And multicollinearity implies that the predictive value of the fitted model breaks down quickly as one moves away from the subspace in which the data concentrate.

For large d, the number of possible subspaces is enormous $(2^d - 2)$, and so the probability that a sample of fixed size n concentrates on an affine shift of one of them, just by chance, is large. (There are 2^d possible subspaces, and we exclude the full space and the degenerate space of dimension 0.)

Concurvity is the nonparametric analogue of multicollinearity, and it occurs when the data concentrate on some smooth manifold within $\mathbb{R}^d$. Since the number of smooth manifolds is larger than the number of affine shifts, the nonparametric version of the problem is worse.

In the early 1990s, several researchers obtained results that purport to evade the COD.

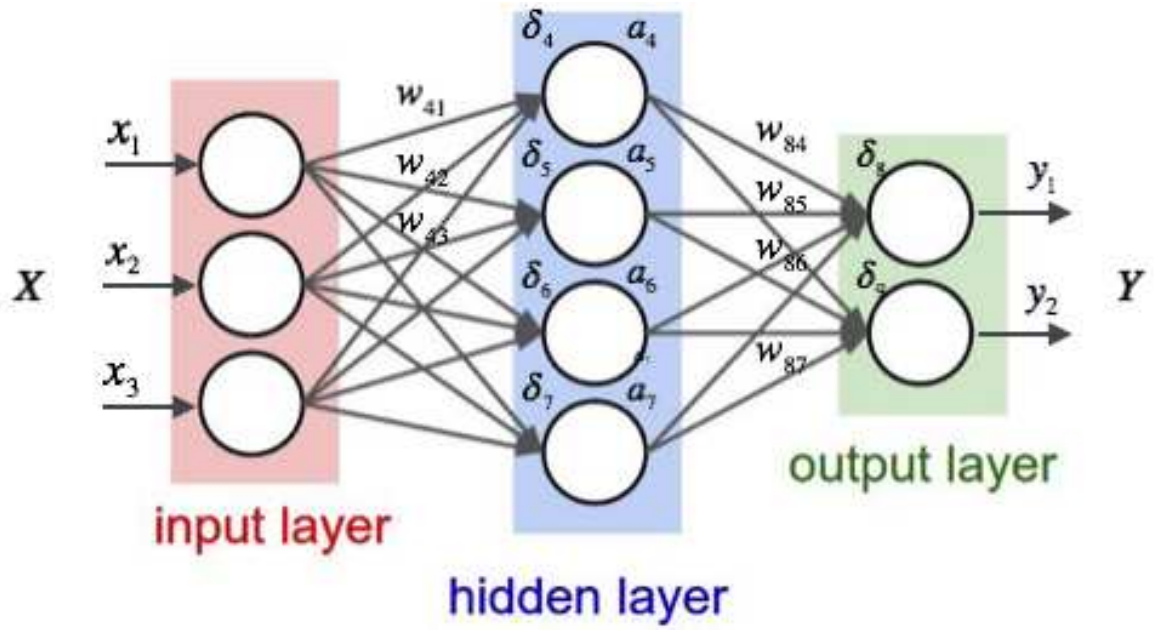
- Barron (1994; *Machine Learning*, **14**, 115-133) shows that in a technical sense which we describe in section 4.4, neural networks avoid the COD.
- Zhao and Atkeson (1991; *NIPS'91*, **4**, 936-943) show that in a sense similar to Barron's, Projection Pursuit Regression can evade the COD.
- Wozniakowski (1991; *Bulletin of the American Mathematical Society*, N.S., **24**, 184-194) used a modification of Hammersley points (chosen to minimize the discrepancy from the uniform distribution in a Kolmogorov-Smirnov test) to dodge the COD in the context of multivariate integration.

However, these evasions all rely upon special assumptions or impractical constructions. The COD is fundamental.

2.2 Basic Neural Networks

A third version of the additive model is **neural networks**. These methods are very close to PPR.

There are many different fiddles on the neural network strategy. We focus on the basic feed-forward network with one hidden layer.



A neural network with a single hidden layer is an example of an **additive model**. In an additive model, one fits

$$\hat{Y} = \sum_{k=1}^{K} sm_k(\boldsymbol{\beta}_k' \boldsymbol{x}).$$

Neural networks fit a model of the form

$$\hat{Y} = \beta_0 + \sum_{k=1}^{K} \beta_k \psi(\boldsymbol{w}_k' \boldsymbol{x} + b_k)$$

where ψ is a sigmoidal (or logistic) function, or ReLU, or other standard choice, and the other parameters (except K) are estimated from the data.

The only difference between Projection Pursuit Regression and the one-hidden-layer neural net is that neural nets assumes that the additive functions have a specific parametric form; e.g.,

$$\psi(\boldsymbol{x}) = \frac{1}{1 + \exp(b + \boldsymbol{w}'\boldsymbol{x})}.$$

The parametric assumption allows neural nets to be trained by backpropagation, an iterative fitting technique similar to backfitting (a statistical algorithm), but somewhat faster because it does not require smoothing.

Barron (1993; *IEEE Transactions on Information Theory*, **39**, 930-945) showed that neural networks evade the Curse of Dimensionality in specific, rather technical, sense. We sketch his result.

A standard way of assessing the performance of a nonparametric regression procedure is in terms of **Mean Integrated Square Error** (MISE). Let $f(\boldsymbol{x})$ denote the true function and $\hat{f}(\boldsymbol{x})$ denote the estimated function. Then

$$\mathbf{MISE}[\hat{g}] = \mathbb{E}_G \left[\int [\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})]^2 \, d\boldsymbol{x} \right]$$

where the expectation is taken with respect to the randomness in the data $\{(Y_i, \boldsymbol{X}_i)\}$.

Before Barron's work, it had been thought that the COD implied that for any regression procedure, the MISE had to grow faster than linearly in p, the dimension of the data. Barron showed that neural networks could attain an MISE of order $\mathcal{O}(r^{-1}) + \mathcal{O}(rp/n) \ln n$ where r is the number of hidden nodes.

Recall that $a_n = \mathcal{O}(h(n))$ means there exists c such that for n sufficiently large, $a_n \leq ch(n)$.

Barron's theorem is technical. It applies to the class of functions $f \in \Gamma_c$ on $\mathbb{R}^p$ whose Fourier transforms $\tilde{f}(\omega)$ satisfy

$$\int |\omega| \tilde{f}(\omega)\, d\omega \leq c$$

where the integral is in the complex domain and $|\cdot|$ denotes the complex modulus.

The class Γ_c is **thick**, meaning that it cannot be parameterized by a finite-dimensional parameter. It contains an open ball in $\mathbb{R}^d$. But it excludes important functions such as hyperflats.

The strategy in Barron's proof is:

- Show that for all $f \in \Gamma_c$, there exists a neural net approximation $\hat{f}^*$ such that $\|f - \hat{f}^*\|^2 \leq c^*/n$.
- Show that the MISE in estimating any of the $\hat{f}^*$ functions is bounded.
- Combine these results to obtain a bound on the MISE of a neural net estimate $\hat{f}$ for an arbitrary $f \in \Gamma_c$.

2.3 Backpropagation

Backpropagation is the standard algorithm for training supervised feed-forward neural nets. Ut isn't actually a learning algorithm, but a way of computing the gradient of the loss function with respect to the network parameters. Mathematically, it's just the chain rule for derivatives, but it has an intuitive interpretation in terms of passing messages between the nodes.

Stochastic gradient descent (SGD) is an iterative optimization algorithm that can be applied to functions that are a linear combination of differentiable functions. These functions often arise when the full objective function is a linear combination of objective functions at each data point; e.g., a least squares objective function. While batch gradient descent uses the full gradient of the function, SGD approximates the full gradient by using the gradient at each of the functions in the linear combination; e.g., the gradient of the objective function at each data point. SGD is often used to optimize non-convex functions, such as those that arise in neural networks.

2.4 Variable Selection

Variable selection is generally helpful in nonparametric regression, especially when the number of explanatory variables is large. It is challenging when $d >> n$. Over the last decade, great progress has been made, and we discuss:

- ridge regression
- the LASSO and its relatives
- the Dantzig selector and relevant theory
- Hoff's method

Variable selection, and the related area of dimension reduction, are very active areas of research in the data analytics community. This review just covers a fraction of the thinking.

2.4.1 Ridge Regression

Ridge regression is an old idea, used originally to protect against multicollinearity. It shrinks the coefficients in a regression towards zero (and each other) by imposing a penalty on the sum of the squares of the coefficients.

$$\hat{\beta} = \mathbf{argmin}_{\beta}\{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{d}\beta_j^2\}$$

where $\lambda \geq 0$ is a penalty parameter that controls the amount of shrinkage.

Recall Stein's result that when estimating a multivariate normal mean with squared error loss, the sample average is inadmissible and can be improved by shinking the estimator towards $\mathbf{0}$ (or, counterintuitively, any other value).

Neural net methods now often do similar shrinkage on the weights at each node, Thereby improving predictive squared accuracy.

Ridge regression methods are not equivariant under scaling, so one normally standardizes the x_{ij} sample values wrt j so that each has unit variance or lies between 0 and 1.

Traditional ridge regression solves

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

so the name derives from the stabilization of the inverse matrix obtained by adding a constant to the diagonal.

Note: This increases the trace of the hat matrix, which corresponds to the degrees of freedom used in fitting the model.

Ridge regression (and shrinkage methods in general) can be seen as the Bayesian posterior mode for a suitable prior. In fact, there is a one-to-one correspondence between a penalty and a prior. In ridge regression, the penalty corresponds to a prior that assumes independent normal distributions for each β_j, with common variance.

2.4.2 The LASSO

The LASSO method is analogous to ridge regression. The LASSO estimate is given by

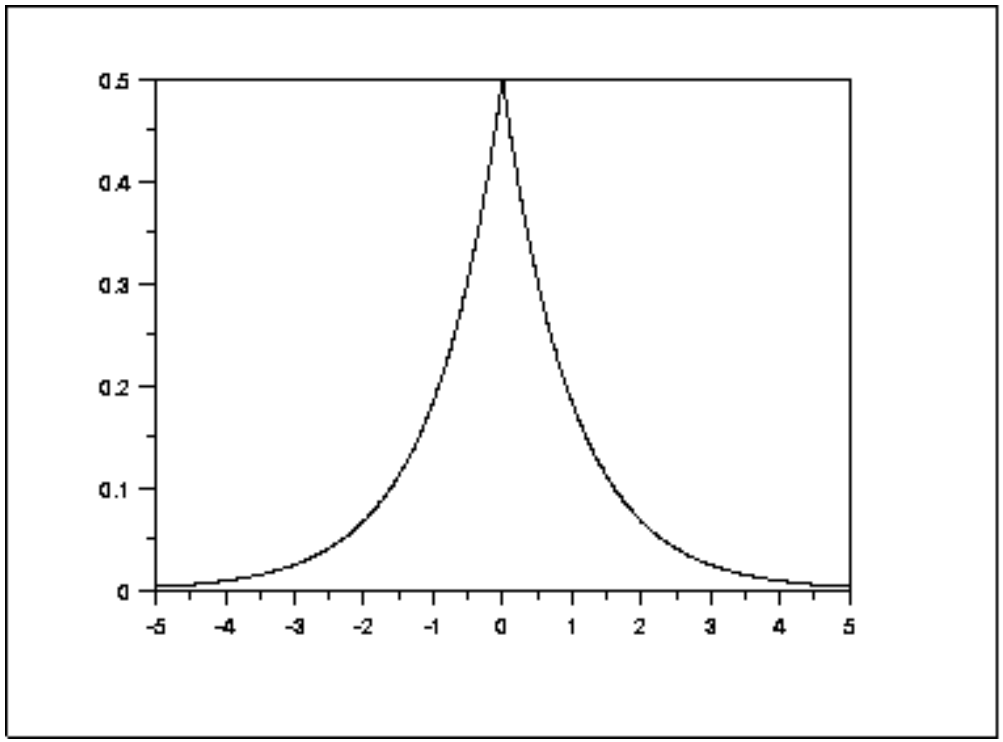
$$\hat{\beta} = \mathbf{argmin}_{\beta}\{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{d}|\beta_j|\}.$$

Often one sees trajectories of LASSO solutions, corresponding to minimizing the sum of squared deviations subject to the constraint that $\sum_{j=1}^{d}|\beta_j| \leq s$.

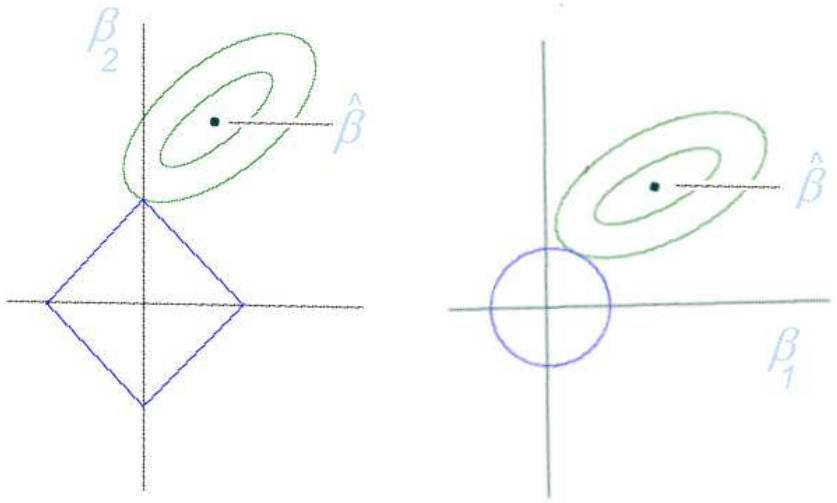
The LASSO replaces the quadratic penalty in ridge regression by a penalty on the sum of the absolute values of the β_j terms.

If s is larger than the sum of the absolute values of the least squares estimators, then the LASSO agrees with OLS. If s is small, then many of the β_j terms are driven to 0, so it is performing variable selection.

The LASSO corresponds to a Bayesian posterior mode in which the prior on each parameter is Laplacian.



Unlike ridge regression, the LASSO is apt to drive coefficients to zero (i.e., perform variable selection) when the penalty is tight. To see this, the left graph shows how the contours for the OLS estimates intersect with the constrained space for the LASSO, versus the right graph which shows the same situation for ridge regression.



See Tibshirani (1996; *Journal of the Royal Statistical Society, Series B*, **58**, 267-288).

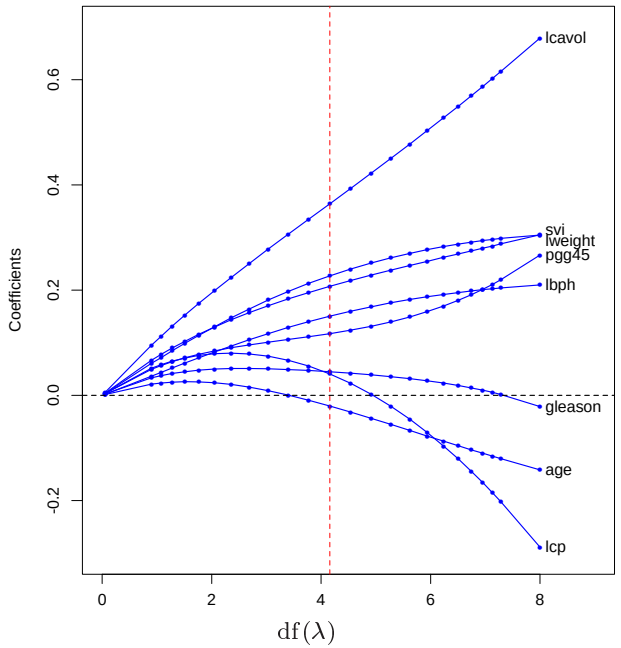


Figure 3.7: *Profiles of ridge coefficients for the prostate cancer example, as tuning parameter* λ *is varied. Coefficients are plotted versus* $\mathrm{df}(\lambda)$*, the effective degrees of freedom. A vertical line is drawn at* $\mathrm{df} = 4.16$*, the value chosen by cross-validation.*

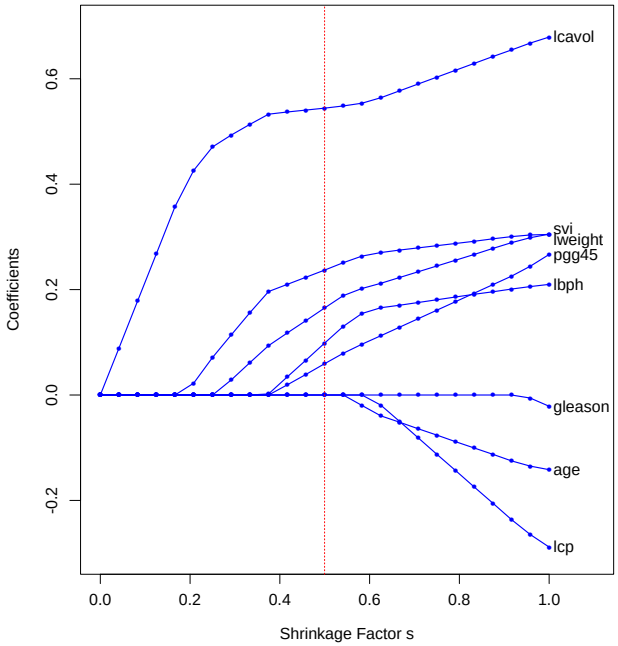
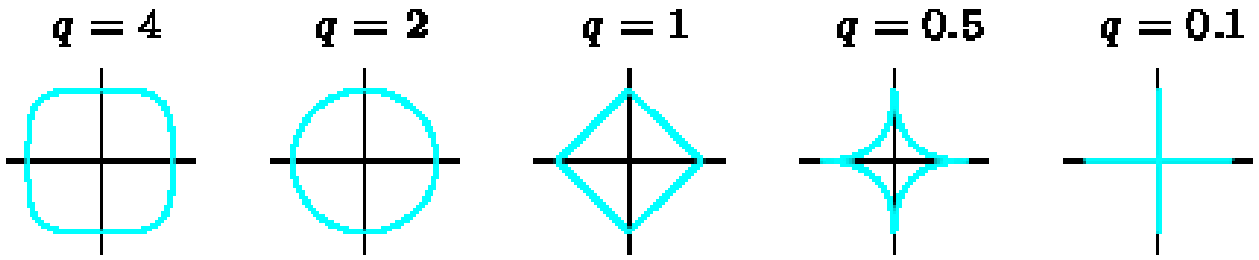


Figure 3.9: *Profiles of lasso coefficients, as tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.5$, the value chosen by cross-validation. Compare Figure 3.7 on page 7; the lasso profiles hit zero, while those for ridge do not.*


$q = 4$
$q = 2$
$q = 1$
$q = 0.5$
$q = 0.1$

2.4.3 The Dantzig Selector

The L^0 penalty strongly favors sparse solutions. But the penalty is not convex, so solution is difficult and often impossible.

Consider L^2 minimization with an L^0 penalty. It solves

$$\hat{\boldsymbol{\beta}} = \mathbf{argmin}_{\boldsymbol{\beta}}\{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{ij})^2 - \lambda\|\boldsymbol{\beta}\|_0\}$$

where $\|\boldsymbol{\beta}\|_0$ is number of non-zero components in the $\boldsymbol{\beta}$ vector.

The non-convexity of the penalty region means that one has to perform combinatorial search to find the solution. However, in some situations, solution is possible—in that case it is called the Dantzig selector by Candes and Tao (2007).

The Dantzig estimator $\hat{\boldsymbol{\beta}}_D$ is based on the fact that sometimes (often) the L^1 solution agrees with the L^0 solution.

The $\hat{\boldsymbol{\beta}}_D$ minimizes the L^1 norm of $\boldsymbol{\beta}$ subject to the condition that

$$\sup_{\boldsymbol{\beta}} \|\boldsymbol{X}'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\| \leq c_D \sigma \sqrt{2 \ln d}$$

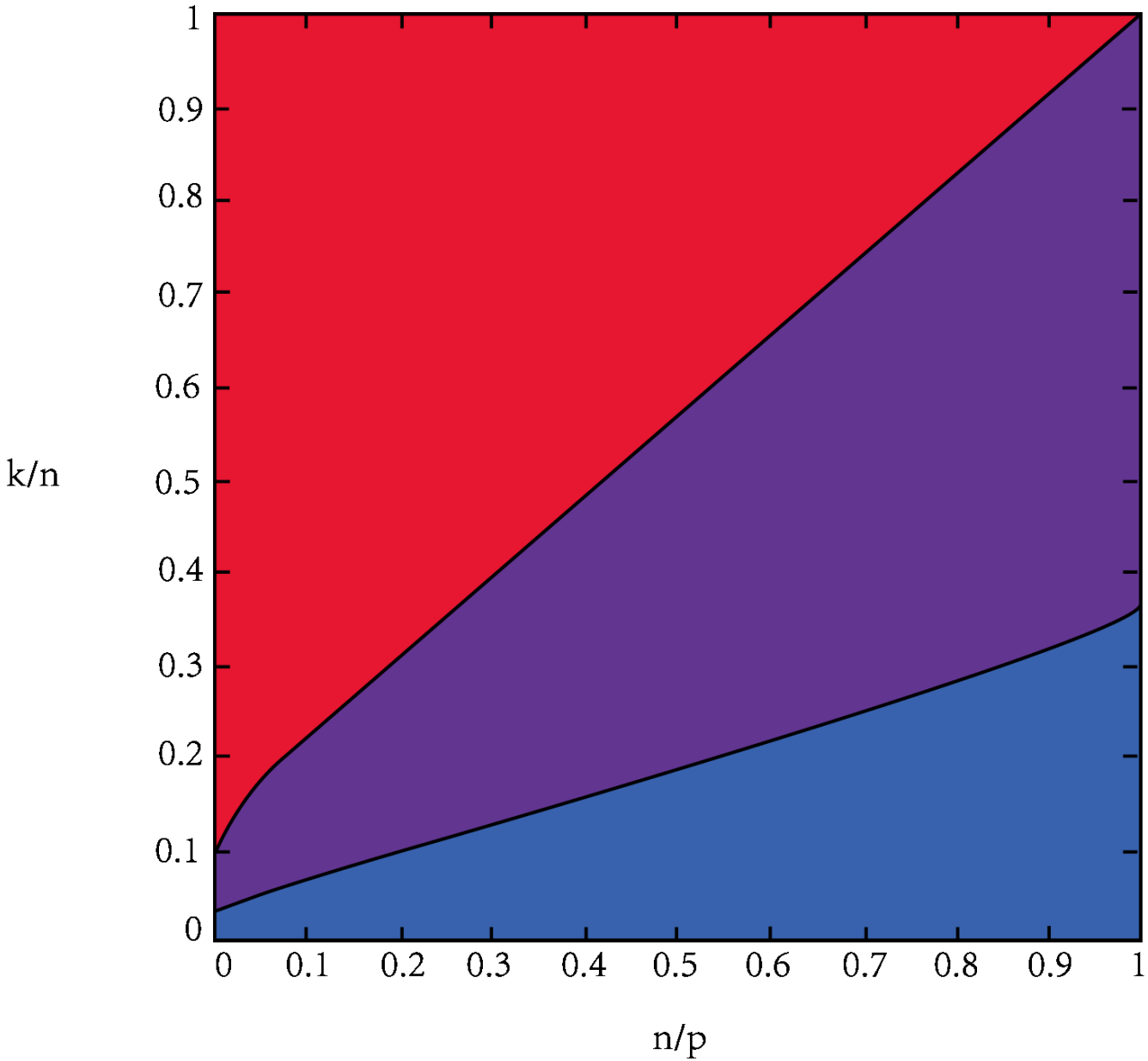
where c_D is a constant that requires some calculation. One uses linear programming to find the minimizing solution.

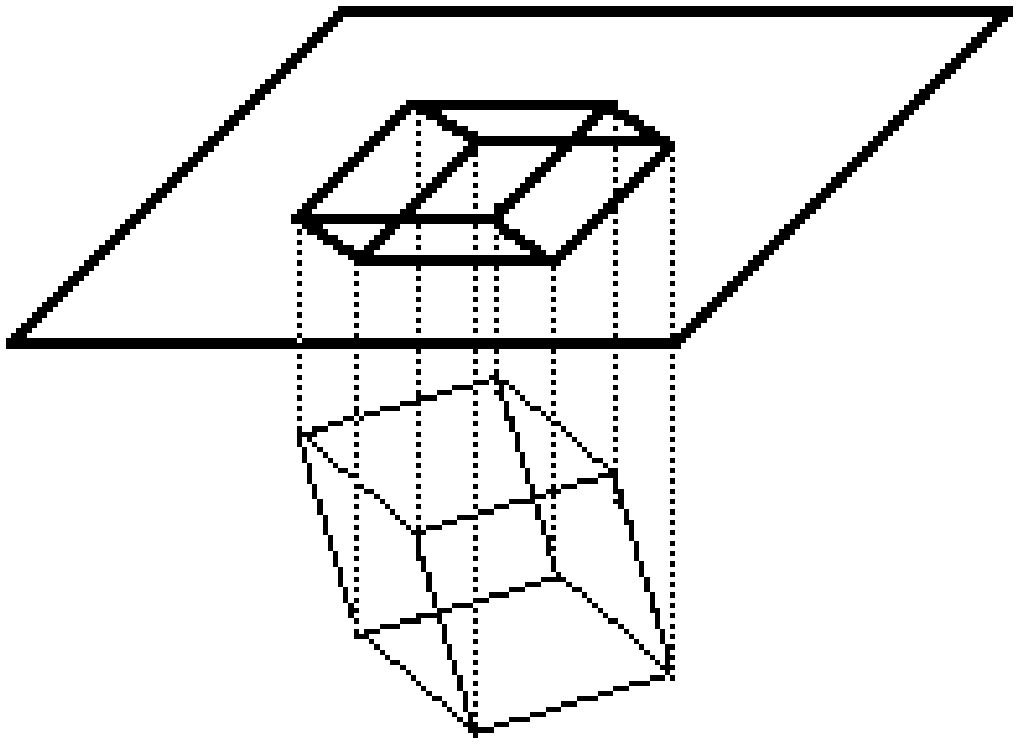
If the number of non-zero elements of the true $\boldsymbol{\beta}$ is $k < n$, then with high probability the Dantzig estimate does asymptotically as well as could be done if the identities of the non-zero components were known to the analyst. (Note: the sense of the asymptotics here is a bit delicate.)

A second framing of this problem is due to Donoho and Tanner (2007). They consider the noise-free case, in which $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$ but $\boldsymbol{X}$ is $n \times p$ with $d >> n$.

Since the $\boldsymbol{X}'\boldsymbol{X}$ matrix is singular, this system of equations cannot be solved. But, in the following diagram, if the vertical axis is k/n and the horizontal axis is n/d, then there appears to be a phase change, somewhere between the indicated boundaries. Above the upper line, solution is impossible; below the lower line, with high probability the solution can be obtained.

The ability to find the solution depends upon whether to solution to the full system is an interior point or an exterior point when the one projects onto a lower dimensional space.


k/n
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
n/p



The third approach to $d >> n$ regression is due to Wainwright (2008). He showed that under certain assumptions on $\boldsymbol{X}$, then with high probability the LASSO correctly identifies the non-zero elements of $\boldsymbol{\beta}$.

Specifically, in the limit,

- if $n > 2k \ln(d-k)$, then the probability that the LASSO succeeds tends to 1;
- if $n < \frac{1}{2} k \ln(d-k)$, then the probability that the LASSO succeeds tends to 0.

Note that all three approaches have similar results—under various conditions, sparse linear regression can work, even when $d >> n$.

But it would be nice to have a way to directly solve the non-convex optimization and impose an L^q penalty on the sum of squares minimization, where $0 \leq q < 1$.

The results for linear regression probably apply to nonparametric regression. Lafferty and Wasserman (2007) attempt to extend methods for nonparametric regression, based on additive models and generalized additive models, to the $d >> n$ situation. Their method is called SpAM (Sparse Additive Modeling). A combination of simulation and theory suggests it works fairly well.

Simulation studies suggest that the phase transition found by Donoho and Tanner for the linear model also arises in many nonlinear and nonparametric cases. But there is really no theory to rely upon, other than a general consensus that anything sensible is roughly approximate to the linear model.

Regarding $d >> n$ classification problems, less statistical work has been done, but most people believe the problem is similar to regression and perhaps somewhat easier.

2.4.4 Hoff's Method

Recall that the LASSO finds $\hat{\beta}$ for the model $\boldsymbol{y} \sim N_d(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I})$ as the minimizer of

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1$$

where $\|\cdot\|$ is the L^2 norm and $\|\cdot\|_1$ is the L^1 norm.

This is equivalent to minimizing

$$f(\boldsymbol{\beta}) = \boldsymbol{\beta}'\boldsymbol{Q}\boldsymbol{\beta} - 2\boldsymbol{\beta}'\boldsymbol{\ell} + \lambda\|\boldsymbol{\beta}\|_1$$

where $\boldsymbol{Q} = \boldsymbol{X}'\boldsymbol{X}$ and $\boldsymbol{\ell} = \boldsymbol{X}'\boldsymbol{y}$.

Now reparameterize the model so that $\boldsymbol{\beta} = \boldsymbol{u} \circ \boldsymbol{v}$ where $\circ$ denotes the Hadamard (element-wise) product. Then the previous minimization of $f(\boldsymbol{\beta})$ is equivalent to estimating $\boldsymbol{u}$ and $\boldsymbol{v}$ in

$$g(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} \circ \boldsymbol{v})'\boldsymbol{Q}(\boldsymbol{u} \circ \boldsymbol{v}) - 2(\boldsymbol{u} \circ \boldsymbol{v})'\boldsymbol{\ell} + \lambda(\boldsymbol{u}'\boldsymbol{u} + \boldsymbol{v}'\boldsymbol{v})/2.$$

Hoff (2016, arXiv) proved that:

- $\inf_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \inf_{(\boldsymbol{u},\boldsymbol{v})} g(\boldsymbol{u},\boldsymbol{v})$,
- if $(\hat{\boldsymbol{u}},\hat{\boldsymbol{v}})$ is a local minimum of g, then $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{u}} \circ \hat{\boldsymbol{v}}$ is a local minimum of f.

Since g is differentiable and biconvex, its local minimizers can be found using an alternating ridge regression algorithm. To see this, rewrite $(\boldsymbol{u} \circ \boldsymbol{v})'\boldsymbol{Q}(\boldsymbol{u} \circ \boldsymbol{v})$ as $\boldsymbol{u}'(\boldsymbol{Q} \circ \boldsymbol{v}\boldsymbol{v}')\boldsymbol{u}$ and $(\boldsymbol{u} \circ \boldsymbol{v})'\boldsymbol{\ell}$ as $\boldsymbol{u}'(\boldsymbol{v} \circ \boldsymbol{\ell})$. So

$$g(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{u}'(\boldsymbol{Q} \circ \boldsymbol{v}\boldsymbol{v}')\boldsymbol{u} - 2\boldsymbol{u}'(\boldsymbol{v} \circ \boldsymbol{\ell}) + \lambda(\boldsymbol{u}'\boldsymbol{u} + \boldsymbol{v}'\boldsymbol{v})/2.$$

This is quadratic in $\boldsymbol{u}$ for fixed $\boldsymbol{v}$, and the unique minimizer is $\hat{\boldsymbol{u}} = (\boldsymbol{Q} \circ \boldsymbol{v}\boldsymbol{v}' + \lambda\boldsymbol{I}/2)^{-1}(\boldsymbol{\ell} \circ \boldsymbol{v})$.

Symmetrically, the unique minimizer of g in $\boldsymbol{v}$ is $\hat{\boldsymbol{v}} = (\boldsymbol{Q} \circ \boldsymbol{u}\boldsymbol{u}' + \lambda\boldsymbol{I}/2)^{-1}(\boldsymbol{\ell} \circ \boldsymbol{u})$.

Alternately minimizing $\boldsymbol{u}$ for the current value of $\boldsymbol{v}$ and $\boldsymbol{v}$ for the current value of $\boldsymbol{u}$ provably converges to a stationary point that is a local minimum.

One can generalize the Hadamard product representation so that

$$\boldsymbol{\beta} = \boldsymbol{u}_1 \circ \boldsymbol{u_2} \circ \cdots \boldsymbol{u}_K.$$

Then one minimizes

$$g(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) = (\boldsymbol{u}_1 \circ \cdots \circ \boldsymbol{u}_K)' \boldsymbol{Q} (\boldsymbol{u}_1 \circ \cdots \circ \boldsymbol{u}_K) - 2(\boldsymbol{u}_1 \circ \cdots \circ \boldsymbol{u}_K)' \boldsymbol{\ell} + \frac{\lambda}{K} (\boldsymbol{u}_1' \boldsymbol{u}_1 + \cdots + \boldsymbol{u}_K' \boldsymbol{u}_K).$$

For $K = 1$ the minimizing $\boldsymbol{u}$ value is the L^2-penalized ridge regression estimate. For $K = 2$ it is the L^1-penalized LASSO estimate. Values greater than 2 give minimizers for L^q-penalties with $q = 2/K$.

Thus one can find solutions for non-convex penalties through the alternating (here, cycling) algorithm previously described.

2.5 Classification

Classification problems attempt to find rules that assign cases to categories, e.g.:

- correct versus incorrect tax returns
- medical diagnoses
- good versus bad credit risks
- terrorists versus non-terrorists.

One uses a training sample of cases for which the categories are known. Each case has a vectore of covariates that are used to build the classification rule.

For a new observation, one looks at its covariate and classifies it according to the classification rule.

There are three main strategies for building classification rules:

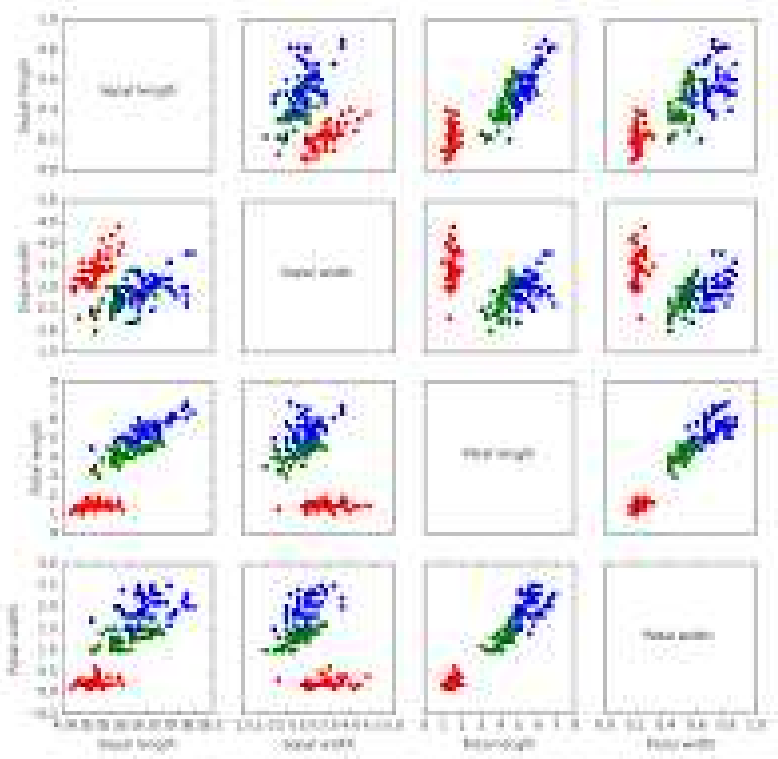
- Geometric, which includes discriminant analysis, flexible discriminant analysis, and recursive partitioning.
- Algorithmic, which includes neural nets (shallow and deep), nearest neighbor rules, support vector machines, and random forests.
- Probabilistic, which includes Bayes rules and hidden Markov models.

The probabilistic methods make strong assumptions about having random samples of data.

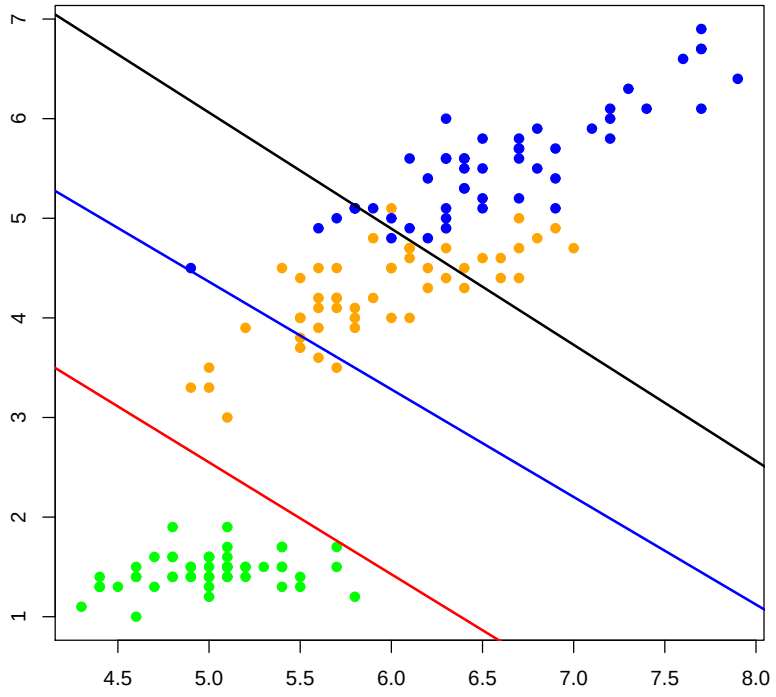
Most classification problems are very similar to regression problems. For problems with two categories, it is sometimes just a matter of fitting a model that predicts 1 or -1 according to the case label.

2.5.1 The COD and Discriminant Analysis

The geometric rules are easy to visualize. They started with Fisher's classification of iris species (1936; *Eugenics*, **7**, 179-188).



Linear discriminant analysis assumes one has multivariate measurements on random members from each population, that those measurements have a multivariate normal distribution, and that one knows the which individuals are from which population. Then one finds the hyperflats (rules) which assign previously unseen members to the most likely population. Below is an example using two of the four iris measurements.



For two classes, Fisher's linear discriminant analysis assumes that the two populations have multivariate normal distribution with common unknown covariance matrix and different unknown mean vectors. It assigns a new observation to the population whose mean has the smallest **Mahalanobis distance** to the observation:

$$d_M(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_j) = [(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)' \boldsymbol{S}^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)]^{1/2}$$

To analyze the effect of noise in linear discriminant analysis, suppose one has a fixed sample size n and assume the covariance matrices are known to be $\sigma^2 \boldsymbol{I}$. Write the estimates of the means as:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_1 &= \boldsymbol{\mu}_1 + \frac{\sigma}{\sqrt{n}} \boldsymbol{v}_1 \\ \hat{\boldsymbol{\mu}}_2 &= \boldsymbol{\mu}_2 + \frac{\sigma}{\sqrt{n}} \boldsymbol{v}_2. \end{aligned}$$

Also, write the new observation to classify as:

$$\boldsymbol{x} = \boldsymbol{\mu}_1 + \boldsymbol{v}.$$

Fisher's classification rule assigns population 1 if

$$d_M(\boldsymbol{x}, \hat{\boldsymbol{\mu}}_1) < d_M(\boldsymbol{x}, \hat{\boldsymbol{\mu}}_2)$$

and under our assumptions, this is equivalent to:

$$(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_1)'(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_1) < (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_2)'(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_2).$$

Writing $\boldsymbol{x}$, $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ in terms of $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, and $\boldsymbol{v}$ shows this criterion is equivalent to:

$$(\boldsymbol{v} - \frac{\sigma}{\sqrt{n}}\boldsymbol{v}_1)'(\boldsymbol{v} - \frac{\sigma}{\sqrt{n}}\boldsymbol{v}_1) < (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \boldsymbol{v} - \frac{\sigma}{\sqrt{n}}\boldsymbol{v}_2)'(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \boldsymbol{v} - \frac{\sigma}{\sqrt{n}}\boldsymbol{v}_2)$$

or, after further simplification,

$$\left[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - \frac{\sigma}{\sqrt{n}}(\boldsymbol{v}_1 + \boldsymbol{v}_2) + 2\sigma\boldsymbol{v}\right]' \cdot \left[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - \frac{\sigma}{\sqrt{n}}(\boldsymbol{v}_1 - \boldsymbol{v}_2) + 2\sigma\boldsymbol{v}\right] > 0.$$

As $n \to \infty$, this criterion converges to

$$2\sigma \boldsymbol{v}'(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) > \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2$$

and thus the asymptotic probability of misclassification is $\mathbb{P}[\boldsymbol{v} > \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|/2\sigma]$. Thus the error rate depends on the **signal-to-noise ratio** $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|/\sigma$.

However, without using asymptotics, then the rule assigning population 1 can be written as

$$2\sigma \boldsymbol{v}'(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 + \sqrt{\frac{2}{n}}\sigma \boldsymbol{v}_1) > -\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2 + \frac{2}{n}\sigma^2 \boldsymbol{v}_1'\boldsymbol{v}_2$$

so that the probability of misclassification is

$$1 - \Phi\left[-\frac{1}{2\sigma}\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|\left(1 + \frac{2\sigma^2}{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}\frac{d}{n}\right)^{1/2}\right].$$

For $d >> n$, this error is very large. (See S. Raudys, *Journal of Multivariate Analysis*, **89**, 1-35).

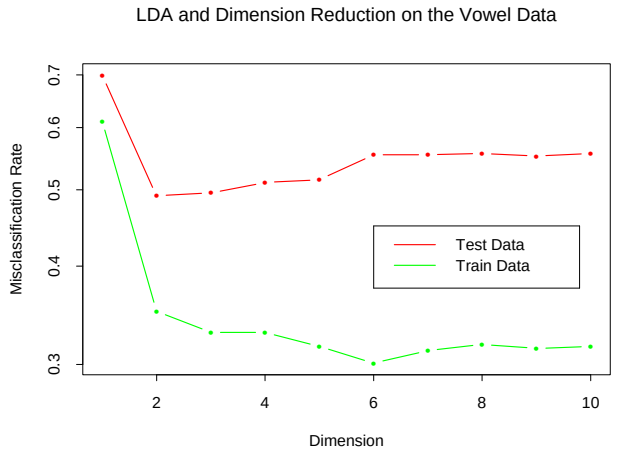


Figure 4.10: *Training and test error rates for the vowel data, as a function of the dimension of the discriminant subspace. In this case the best error rate is for dimension 2. Figure 4.11 shows the decision boundaries in this space.*

2.5.2 Nearest-Neighbor Classification

A very reasonable approach to classification is kth nearest-neighbor classification. In order to make a prediction about a new observation, one looks at the labels of its k nearest neighbors and uses a majority vote to make the prediction.

As the number of neighbors used in making the prediction increases, the decision boundaries become smoother—the bias increases, but the variance decreases. The "degrees-of-freedom" for nearest-neighbor rules is n/k.

The following images show how nearest-neighbor classification rules perform on data simulated from a 20-component mixture of normals, half of which were red, half of which were green. The effect of the bias-variance tradeoff is quite clear.

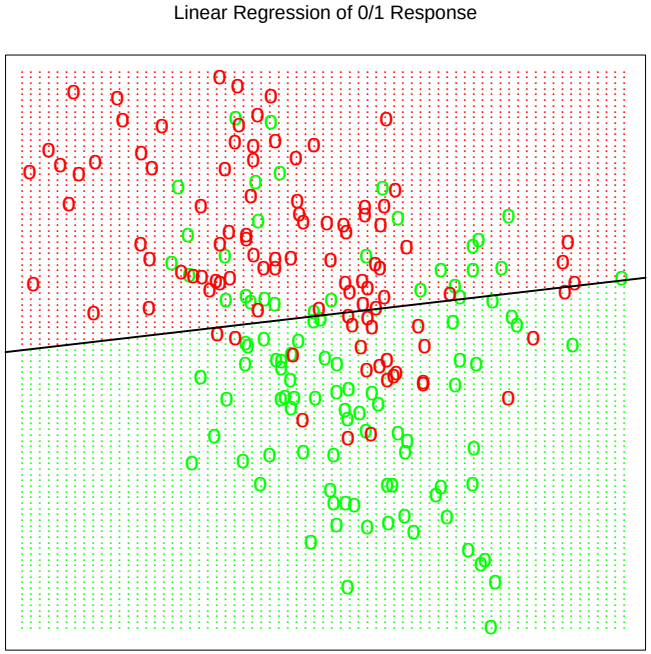


Figure 2.1: *A classification example in two dimensions. The classes are coded as a binary variable—* GREEN $= 0$, RED $= 1$*—and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The red shaded region denotes that part of input space classified as* RED*, while the green region is classified as* GREEN*.*

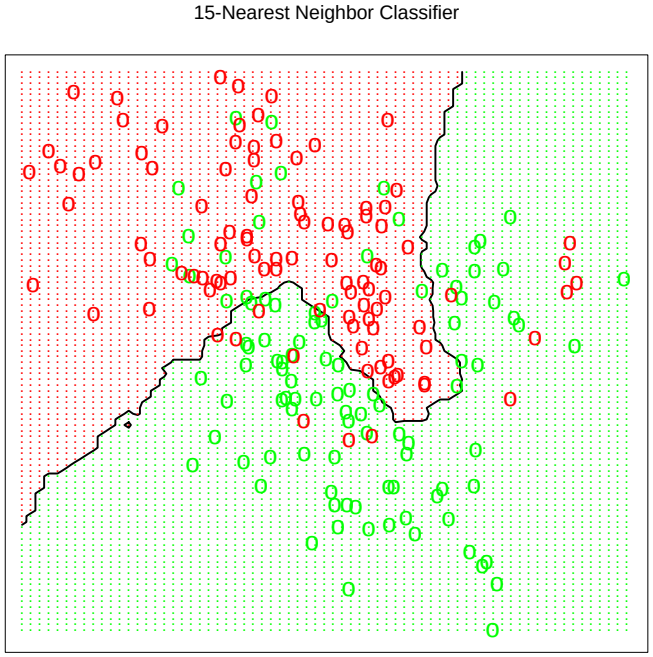


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

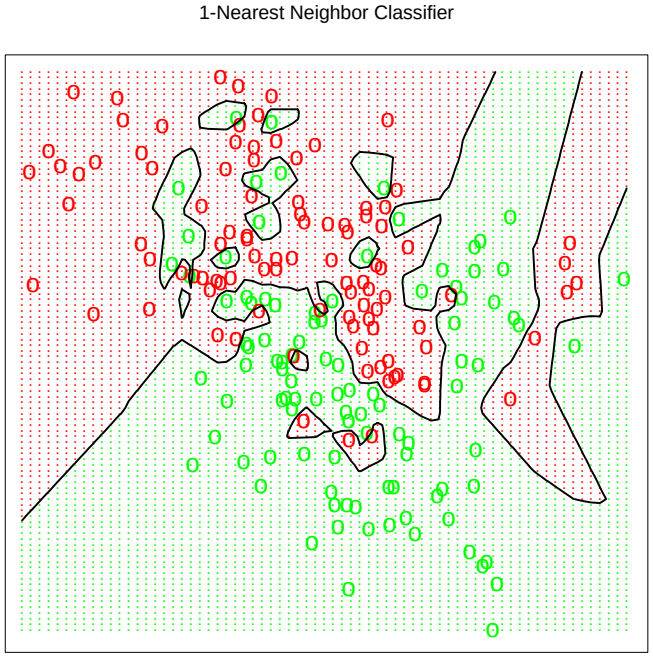


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$), *and then predicted by* 1*-nearest-neighbor classification.*

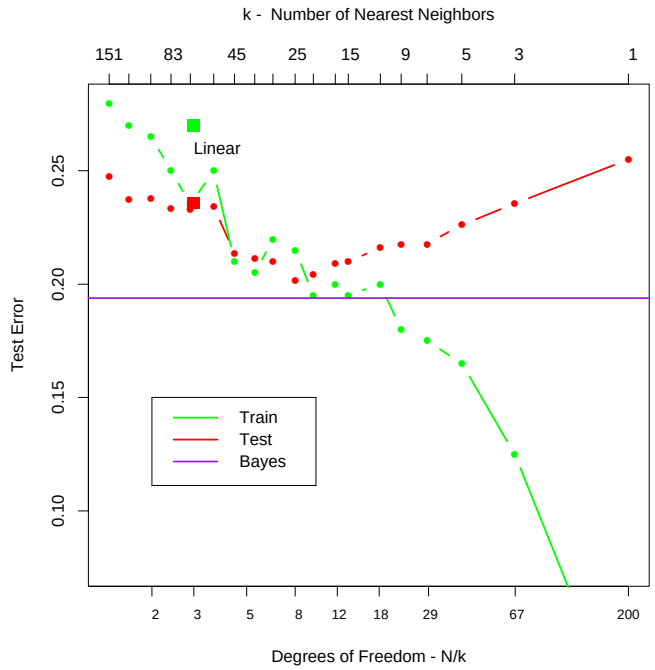


Figure 2.4: *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* $10,000$. *The red curves are test and the green are training error for k-nearest-neighbor classification. The results for linear regression are the bigger green and red dots at three degrees of freedom. The purple line is the optimal Bayes Error Rate.*

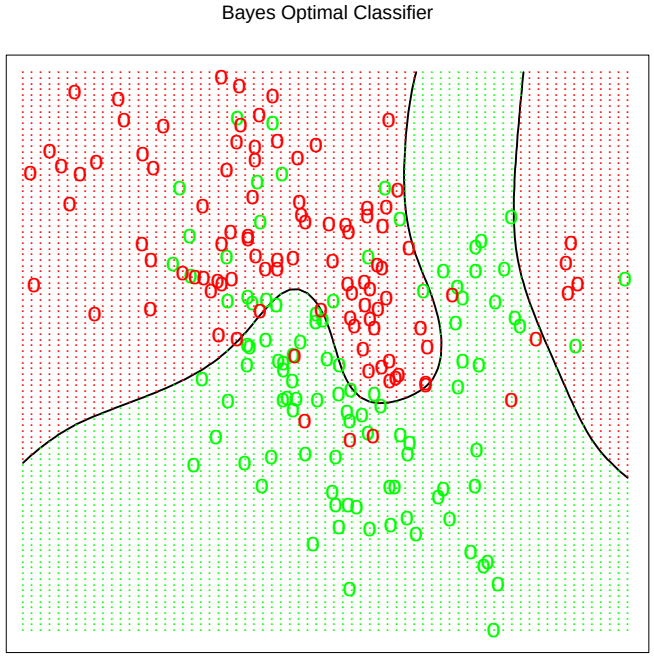


Figure 2.5: *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

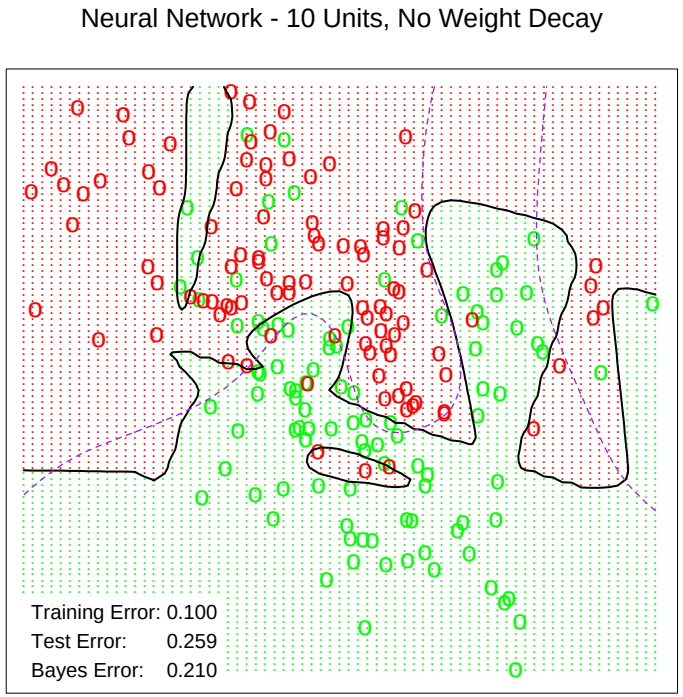
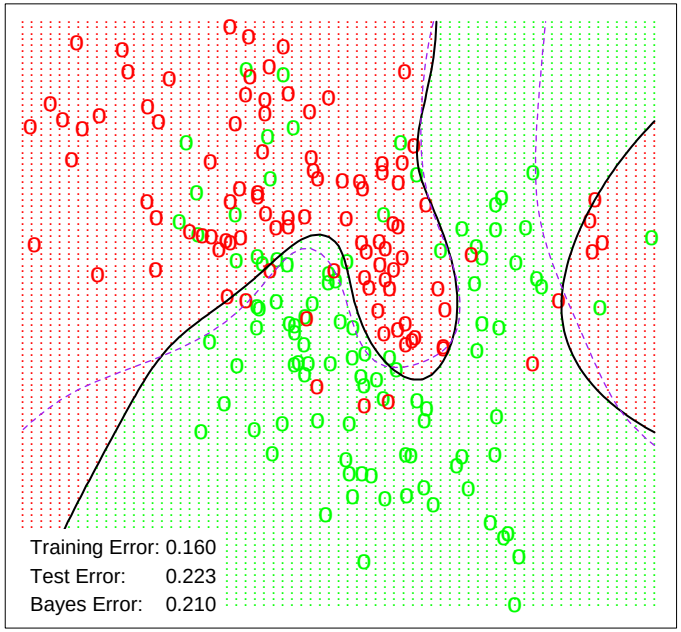


Figure 11.4: *A neural network on the mixture example of Chapter 2. The upper panel uses no weight decay, and overfits the training data. The lower panel uses weight decay, and achieves close to the Bayes error rate (broken purple boundary). Both use the softmax activation function and cross-entropy error.*

Neural Network - 10 Units, Weight Decay=0.02



2.5.3 Support Vector Machines

Support Vector Machines (SVMs) were invented by Vapnik (1996; *The Nature of Statistical Learning*, Springer). They use optimization methods to find surfaces that best separate categories.

Their key innovation is to express the surfaces in terms of a vastly expanded set of basis functions. Instead of using just a standard orthogonal basis, SVMs use many basis elements.

This expansion is called overcompleteness and it proving surprisingly successful in both regression and classification contexts.

The notion of expnading the basis set is alien to the way that statisticians think, and thus valuable.

SVMs have been developed in many different directions, and have become quite elaborate. This survey focuses upon classifying two groups, and develop the ideas in three steps:

- Linear machines trained on separable classes;
- Linear machines trained on overlapping classes;
- Nonlinear machines.

The extension to more-than-two category classification can be done in the way that linear discriminant analysis is extended, but there are subtleties. See Zhang and Liu (2013, Journal of Machine Learning Research, 1349–1386).

SVMs, along with Random Forests, are two classification procedures that regularly achieve excellent performance in bake-offs.

2.5.3.1 Linear Machines, Separable Data

Suppose the n observations are $\{(\boldsymbol{x}_i, y_i)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. And suppose one seeks a simple linear classification rule of the form

$$g(\boldsymbol{x}) = \mathbf{sign}[\boldsymbol{\beta}^T \boldsymbol{x} + \beta_0]$$

where $\boldsymbol{\beta}$ is determined from the data. (WLOG, assume $\|\boldsymbol{\beta}\| = 1$.)

If the two classes are **separable** then there are many possible $\boldsymbol{\beta}$ that work. A natural strategy is to pick the $\boldsymbol{\beta}$ that creates the biggest **margin** between the two classes.

The margin is twice perpendicular distance from the closest value with label $+1$ to the separating hyperflat (or the sum of the two smallest distances, one from each class).

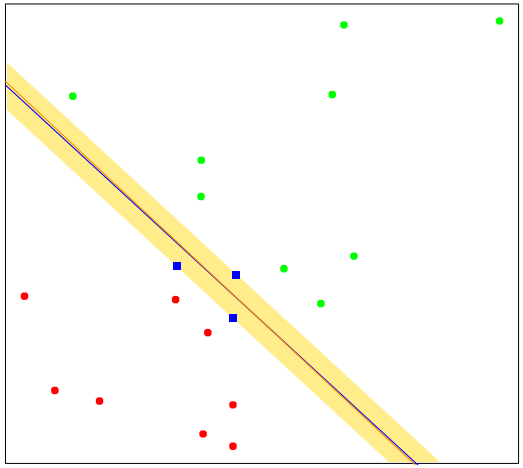


Figure 4.15: *The same data as in Figure 4.13. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

Denote the margin by w. Then the optimization problem is to

$$\max_{\boldsymbol{\beta}, \beta_0, \|\boldsymbol{\beta}\|=1} w$$
$$\textbf{subject to} \quad y_i(\boldsymbol{\beta}^T \boldsymbol{x}_i + \beta_0) \geq w, \;\; i = 1, \ldots n.$$

One can rewrite this as a convex optimization problem with with a quadratic criterion and linear constraints:

$$\min \|\boldsymbol{\beta}\|$$
$$\textbf{subject to} \quad y_i(\boldsymbol{\beta}^T \boldsymbol{x}_i + \beta_0) \geq 1, \;\; i = 1, \ldots n$$

where the requirement that $\boldsymbol{\beta}$ have unit norm is dropped. It turns out that $w = \|\boldsymbol{\beta}\|^{-1}$.

Note that

- this solution is not equivalent to the one obtained from linear discriminant analysis;
- the solution depends only upon the two closest points in the two classes.

To solve the rewritten problem we can use a Lagrangian:

$$L = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^{n} \lambda_i y_i (\boldsymbol{x}_i' \boldsymbol{\beta} + \beta_0) + \sum_{i=1}^{n} \lambda_i$$

The goal is to minimize L with respect to $\boldsymbol{\beta}$ and β_0, while simultaneously requiring $\lambda_i \geq 0$ and that the derivatives of L with respect to the λ_i vanish.

The Lagrangian formulation has two advantages:

- the original constraints are replaced by constraints on the Lagrange multipliers, which are easier to handle;
- the training data only appear once, as dot products in a sum, which allows generalization to nonlinear machines.

The dual problem of the Lagrangian minimization is to maximize L subject to:

$$\begin{aligned} 0 &= \frac{\partial L}{\partial \beta_i} \textbf{ for all } i \\ 0 &= \frac{\partial L}{\partial \beta_0} \\ 0 &\leq \lambda_i \textbf{ for all } i. \end{aligned}$$

This is called the **Wolfe dual** (see Fletcher, 1987, *Practical Methods of Optimization*, Wiley).

The zero-gradient requirement generates equality constraints, leading to

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j$$

which is maximized under the constraint that $\sum_{i=1}^{n} \lambda_i y_i = 0$ and $\lambda_i \geq 0$ for all i.

Training the SVM amounts to solving this optimization problem. We used a primal-dual solution, but anything that satisfies the Karush-Kuhn-Tucker conditions will be necessary and sufficient.

Note that for each observation, there is a Lagrange multiplier λ_i. Those observations with $\lambda_i > 0$ are the support vectors, and determine the margin. For all the other observations the λ_i are zero. If the problem were reworked using only the support vectors data, the same solution would be found.

Also note that the β_0 is determined implicitly. Solve

$$\lambda_i[y_i(\boldsymbol{\beta}'\boldsymbol{x}_i + \beta_0) - 1] = 0$$

using any support vector observation.

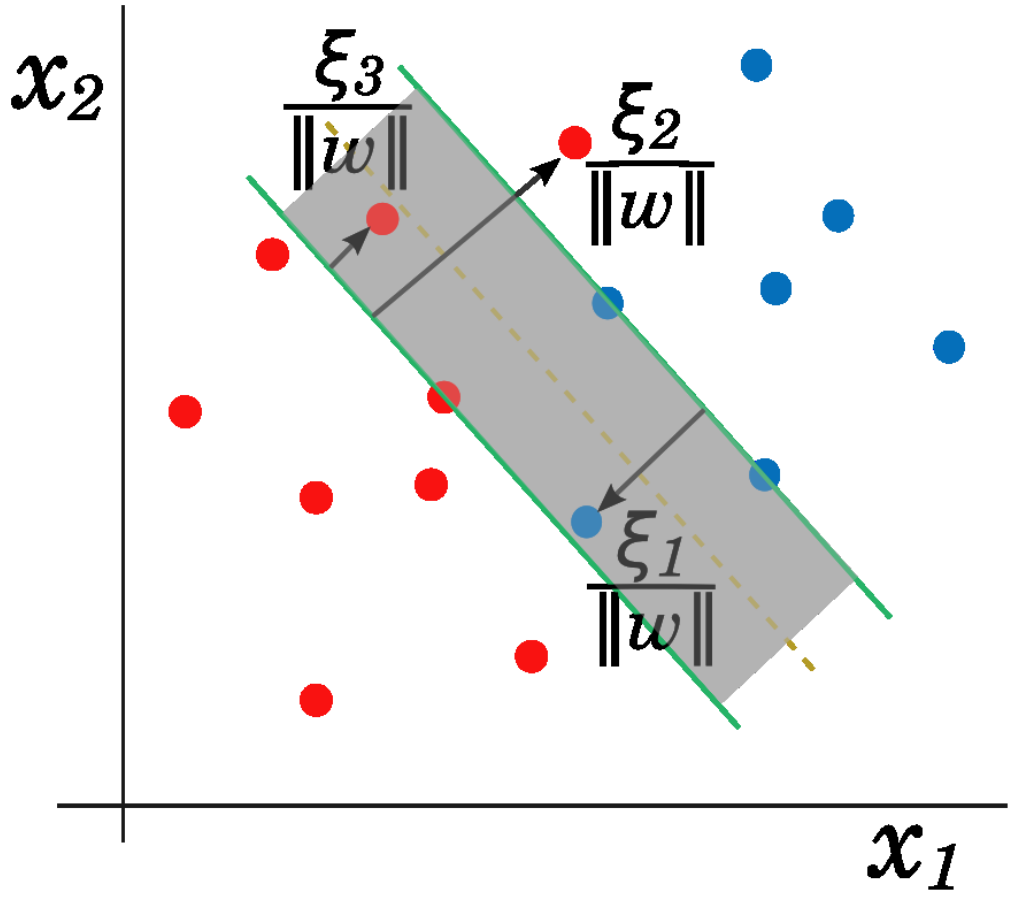
2.5.3.2 Linear Machines, Nonseparable Data

In practice, it usually happens that the two classes are not separable. In that case one wants to find the hyperflat that minimizes the sum of the perpendicular distances for the data that violate the rule. This leads to a slightly more advanced optimization problem that includes **slack variables**. This can be solved using Lagrange multipliers.

Cortes and Vapnik (1995; *Machine Learning*, **20**, 273-297) developed support vector machines to extend the optimization strategy we have described.

One can also view this as a problem in physics, in which each of the support vectors exerts a force on a rigid sheet. The equilibrium solution determines the separator.

The following figure shows a situation in which one is trying to use a linear machine with nonseparable data. The three arrows indicate the slack variables, the sum of whose lengths will be minimized.



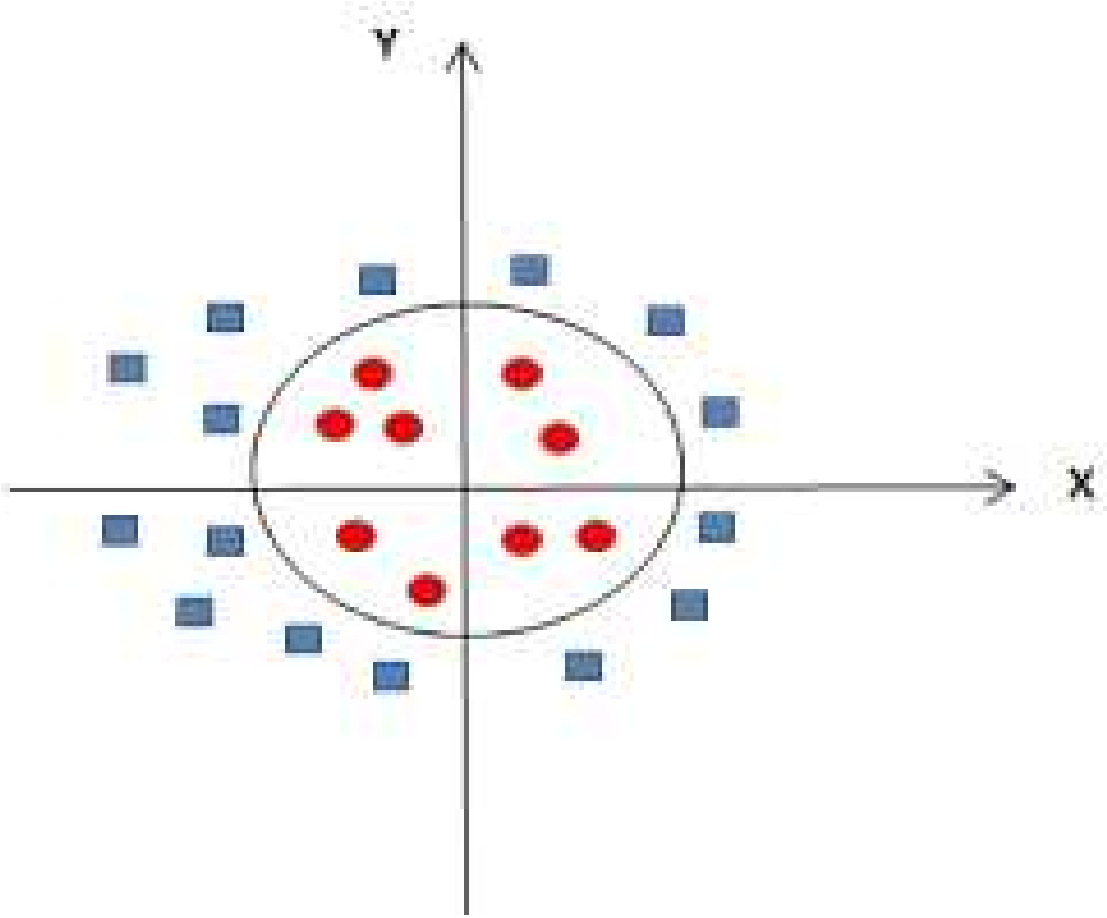
2.5.3.3 Nonlinear Machines

The mathematics in the SVM literature can be serious, but the key idea is to find linear combinations of basis functions of $\boldsymbol{x}$ that describe good separating surfaces.

As previously noted, in real problems one wants more flexible separating surfaces than hyperflats. Boser, Guyon, and Vapnik (1992; *Fifth Annual Workshop on Computational Learning Theory*, ACM) showed how to do this in the SVM context, using an old idea of Aizerman et al. (1964; *Automation and Remote Control*, **25**, 821-837).

The main innovation is to express the surfaces in terms of a vastly expanded set of basis functions. SVMs map the problem to a higher dimensional space, and then solve the linear separation problem using nonlinear basis elements.

As a simple example of this strategy, consider the following figure.



There is clearly no linear separator. But if one expands the basis functions to include terms of the form $(x-a)^2+(y-b)^2$, then one can find a linear separator in the expanded space which maps back to a separating circle in the original space.

Consider the problem of building a classification rule for just two types of objects, where each object has measurements in $\mathbb{R}^d$. One can find separating hyperplanes or quadratic surfaces, as in linear or quadratic discriminant analysis, or one can fit more complex separating surfaces.

In principle, one would like to be able to fit very curvy surfaces, when the data warrant, that can separate complex structure in the two classes. Such surfaces can be formed by linear combinations of basis functions.

Recall that some of the standard basis sets are the Fourier basis, the Legendre polynomials, and so forth.

A basis set is set of functions $\{f_i\}$ such that any function g in the space can be written as

$$g(\boldsymbol{x}) = \sum_{i=1}^{\infty} \beta_i f_i(\boldsymbol{x})$$

and no element of $\{f_i\}$ can be written as a linear combination of the other elements.

The SVM strategy is to greatly expand the dimension of the input space beyond d, and then find hyperflats in that expanded space that classify the training sample. Then one maps back down to the lower-dimensional space, and it generally happens that the linear separating rules in the high-dimensional space become non-linear rules in the p dimensional space.

Consider expanding the set of inputs to include additional functions of the inputs, say $\boldsymbol{h}(\boldsymbol{x}_i) = (h_1(\boldsymbol{x}_i, \ldots, h_q(\boldsymbol{x}_i))^T$. If these are just a single component of the $\boldsymbol{x}_i$ data then $q = d$ and this reduces to the simple case described in 2.5.3.1 or 2.5.3.2.

One can show that the separating surface has the form

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} \beta_i < \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}_i) > +\beta_0$$

where $< \cdot, \cdot >$ denotes an inner product.

Formally, suppose we use a mapping H to take the data from $\mathbb{R}^d$ to a (possibly infinite-dimensional) Euclidean space $\mathcal{H}$.

Recall that the optimization problem can be written so that it only depends upon dot products of the data, $\boldsymbol{x}_i'\boldsymbol{x}_j$. Thus after the mapping, the solution in $\mathcal{H}$ depends only upon functions of the form $\boldsymbol{h}(\boldsymbol{x}_i)'\boldsymbol{h}(\boldsymbol{x}_j)$.

Therefore it is sufficient to find a function

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{h}(\boldsymbol{x}_i)'\boldsymbol{h}(\boldsymbol{x}_j).$$

The nice thing is that we do not need to know the mapping H.

Determining the functions in $\boldsymbol{h}(\boldsymbol{x})$ turns out to be equivalent to selecting a basis for a particular subspace. This depends upon a **kernel function**.

A kernel function is a positive semidefinite function

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{x}^*) &= < \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}^*) > \\ &= \sum_{j=1}^{q} h_j(\boldsymbol{x})' h_j(\boldsymbol{x}^*). \end{aligned}$$

This is related to reproducing kernel Hilbert spaces.

Three common choices for kernel functions in SVM applications are:

- $K(\boldsymbol{x}, \boldsymbol{x}^*) = \exp(-\|\boldsymbol{x} - \boldsymbol{x}^*\|^2/c)$, known as the radial basis;
- $K(\boldsymbol{x}, \boldsymbol{x}^*) = \tanh(\alpha_1 < \boldsymbol{x}, \boldsymbol{x}^* > +\alpha_2)$, or the neural network basis; and
- $K(\boldsymbol{x}, \boldsymbol{x}^*) = (1+ < \boldsymbol{x}, \boldsymbol{x}^* >)^r$, the rth degree polynomials.

To see how this works, suppose $d = 2$ and use the rth degree polynomial basis with $r = 2$. Then

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{x}^*) &= (1 + < \boldsymbol{x}, \boldsymbol{x}^* >)^2 \\ &= (1 + x_1 x_1^* + x_2 x_2^*)^2 \\ &= 1 + 2x_1 x_1^* + 2x_2 x_2^* + (x_1 x_1^*)^2 + (x_2 x_2^*)^2 + \\ &\quad 2x_1 x_2 x_1^* x_2^*. \end{aligned}$$

Thus $q = 6$, and with a little algebra one can show that

$$\begin{aligned} h_1(\boldsymbol{x}) &= 1 & h_2(\boldsymbol{x}) &= \sqrt{2} x_1 \\ h_3(\boldsymbol{x}) &= \sqrt{2} x_2 & h_4(\boldsymbol{x}) &= x_1^2 \\ h_5(\boldsymbol{x}) &= x_2^2 & h_6(\boldsymbol{x}) &= \sqrt{2} x_1 x_2. \end{aligned}$$

So the SVM is looking for quadratic discriminant rules, and the programming problem finds the best quadratic surface (in terms of maximizing the margin in $\mathbb{R}^6$) that separates the classes.

As another example, consider the radial basis kernel

$$K(\boldsymbol{x}, \boldsymbol{x}^*) = \exp(-\|\boldsymbol{x} - \boldsymbol{x}^*\|^2/c).$$

In this case $q = \infty$; $\mathcal{H}$ is an infinite-dimensional Hilbert space.

The Nadaraya-Watson estimate for regression with a Gaussian kernel can be represented as a linear combination of radial basis functions:

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} y_i h_i(\boldsymbol{x}).$$

Here

$$h_i(\boldsymbol{x}) = a \exp[-b(\boldsymbol{x} - \boldsymbol{x}_i)'(\boldsymbol{x} - \boldsymbol{x}_i)]$$

where $a = \sum_{i=1}^{n} h_i(\boldsymbol{x})$ is the normalization constant and b is the bandwidth.

For SVMs, the separating surface is $g(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{h}(\boldsymbol{x}) + \beta_0$.

In the context of classification, the corresponding optimization problem that must be solved is to find $\boldsymbol{\beta}$ and β_0 is

$$\min_{\boldsymbol{\beta},\beta_0} \sum_{i=1}^{n} [1 - y_i g(\boldsymbol{x}_i)]_+ + \lambda \|\boldsymbol{\beta}\|^2$$

where $[\cdot]_+$ denotes the positive part of the argument.

This has the form of minimizing a loss function plus a penalty. The penalty is on the length of $\boldsymbol{\beta}$, and the loss is a weighted sum of the misclassifications.

SVMs do not automatically avoid the Curse of Dimensionality. For example, as d gets large, the 2nd degree polynomial basis becomes very much like multiple polynomial regression, and we already know that has problems.