Elements of statistical learning Ch. 4 notes

James Chuang

February 3, 2017

Contents

4.1 Introduction

Linear methods of classification– those where the *decision boundaries* between classes are linear.

Several different ways to find linear decision boundaries:

- fit a linear regression model to the class indicator variables, and classify to the largest fit (Ch. 2)
  - Suppose there are K classes, labeled $1, 2, \dots, K$, and the fitted linear model for the kth indicator response variable is $\hat{f}_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^T x$. The decision boundary between class k and l is that set of points for which $\hat{f}_k(x) = \hat{f}_\ell(x)$.
  - This regression approach is a member of a class of methods that model ***discriminant functions*** $\delta_k(x)$ for each class, and then classify x to the class with the largest value for its discriminant function.
    - Methods that model the posterior probabilities $P(G = k \mid X = x)$ are also in this class. If either the $\delta_k(x)$ or $P(G = k \mid X = x)$ are linear in x, then the decision boundaries will be linear.
      - This remains true for monotone transformations of $\delta_k(x)$ or $P(G = k \mid X = x)$. For example, in two-class classification, a popular model for the posterior probabilities is

$$P(G = 1 \mid X = x) = \frac{\exp\left(\beta_0 + \beta^T x\right)}{1 + \exp\left(\beta_0 + \beta^T x\right)},$$
$$P(G = 2 \mid X = x) = \frac{1}{1 + \exp\left(\beta_0 + \beta^T x\right)}$$

      - The monotone transformation used here is the ***logit*** transformation: $\log\left[\frac{p}{1-p}\right]$ (the inverse of the logistic sigmoid function)

$$\log\frac{P(G = 1 \mid X = x)}{P(G = 2 \mid X = x)} = \beta_0 + \beta^T x$$

        . Here the decision boundary is the set of points for which the *log-odds* are zero.
      - Two popular but different methods resulting in linear log-odds or logits: ***linear discriminant analysis*** and ***linear logistic regression***. The essential difference between the two is in the way the linear function is fit to the training data.
  - A more direct approach: explicitly model the boundaries between the classes as linear. For two classes, this amounts fo modeling the decision boundary as a hyperplane. Two methods for this: the ***perceptron*** model, which finds a separating hyperplane in the data, if it exists, and a method for finding an ***optimally separating hyperplane*** if one exists, or else a hyperplane that minimizes some measure of overlap in the training data.
  - The linear approaches in this chapter can be generalized with basis expansions.

4.2 Linear Regression of an Indicator Matrix

Code response in an ***indicator response matrix*** $\mathbf{Y}$, an $N \times K$ matrix of N training instances, where $Y_k = 1$ if $G = k$, else 0. Fit a linear regression model to each of the columns of $\mathbf{Y}$ simultaneously. The fit is given by

$$\hat{\mathbf{Y}} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

, where the $(p+1) \times K$ coefficient matrix $\mathbf{B} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$.

A new observation with input x is classified as follows:

- compute the fitted output $\hat{f}(x)^T = (1, x^T)\hat{\mathbf{B}}$, a K vector;
- identify the largest component and classify accordingly:

$$\hat{G}(x) = \text{argmax}_{k \in G}\hat{f}_k(x)$$

???????????????????? pg.104

4.3 Linear Discriminant Analysis

Decision theory for classification says that we need to know the class posteriors $P(G \mid X)$ for optimal classification. Suppose $f_k(x)$ is the class-conditional probability density of X in class $G = k$ (i.e. $P(X \mid G) = f_k(x)$), and let π_k be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$. Applying Bayes rule:

$$P(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$

In terms of ability to classify, having $f_k(x)$ is almost equivalent to having the quantity $P(G = k \mid X = x)$. Many techniques are based on models for the class densities $f_k(x)$:

- linear and quadratic discriminant analysis: $f_k(x)$ are Gaussian
- flexible mixtures of Gaussians allow nonlinear decision boundaries
- nonparametric density estimates for each class density allow the most flexibility
- ***Naive Bayes*** models are a variant of the previous case, and assume that each of the class densities are products of marginal densities; i.e., they assume that the inputs are conditionally independent in each class

Suppose that we model each class density as multivariate Gaussian:

$$f_k(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$$

Linear discriminant analysis (LDA) arises in the special case when we assume that the classes have a common covariance matrix $\Sigma_k = \Sigma \forall k$. In comparing two classes k and l, it is sufficient to look at the log-ratio:

$$
\begin{aligned}
&\log \frac{P(G=k \mid X=x)}{P(G=k \mid X=x)} \\
&= \log \frac{f_k(x)\pi_k}{f_\ell(x)\pi_\ell} \\
&= \log \frac{f_k(x)}{f_\ell(x)} + \log \frac{\pi_k}{\pi_\ell} \\
&= \log \frac{\pi_k}{\pi_\ell} + \log \frac{e^{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)}}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \frac{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}}{e^{-\frac{1}{2}(x-\mu_\ell)^T\Sigma^{-1}(x-\mu_\ell)}} \qquad \text{Normalization factors cancel} \\
&= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k) + \frac{1}{2}(x-\mu_\ell)^T \Sigma^{-1}(x-\mu_\ell) \\
&= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \frac{1}{2}x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_\ell + \frac{1}{2}\mu_\ell^T\Sigma^{-1}\mu_\ell \quad \text{Quadratic parts cancel} \\
&= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k-\mu_\ell)^T \Sigma^{-1}(\mu_k-\mu_\ell) + x^T\Sigma^{-1}(\mu_k-\mu_\ell)
\end{aligned}
$$

This equation for log-odds is linear in x, implying that the decision boundary between k and l is linear in x, and in p dimensions is a hyperplane.

In practice the parameters of the generating Gaussians are unknown, and need to be estimated from the training data:

- $\hat{\pi}_k = N_k/N$, where N_k is the number of class-k observations;
- $\hat{\mu}_k = \sum_{g_i=k} x_i/N_k$;
- $\hat{\mathbf{\Sigma}} = \sum_{k=1}^{k} \sum_{g_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (N-K)$

QDA arises in the case that the classes do not have common covariance matrices, and thus the quadratic term in the discriminant function does not cancel. ?????????????

4.4 Logistic Regression

Logistic regression- model the posterior probabiltiies of the K classes via linear functions in x, while ensuring that they sum to one and remain in $[0,1]$ (so that they can be interepreted as probabilties). The model:

$$
\begin{aligned}
\log \frac{P(G=1 \mid X=x)}{P(G=K \mid X=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{P(G=2 \mid X=x)}{P(G=K \mid X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log \frac{P(G=K-1 \mid X=x)}{P(G=K \mid X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x
\end{aligned}
$$

The model is specified in terms of $K-1$ log-odds or logit transformations (reflecting the constraint that the probabilities sum to one, so the Kth probability is determined by the previous $K-1$). From the log-odds above, we can calculate the probability of the last

(Kth) class:

$$\begin{aligned}
P(G = K \mid X = x) &= 1 - \sum_{\ell=1}^{K-1} P(G = \ell \mid X = x) \\
P(G = K \mid X = x) &= 1 - \sum_{\ell=1}^{K-1} P(G = K \mid X = x) \exp\left(\beta_{\ell 0} + \beta_\ell^T x\right) \\
P(G = K \mid X = x) &= 1 - P(G = K \mid X = x) \sum_{\ell=1}^{K-1} \exp\left(\beta_{\ell 0} + \beta_\ell^T x\right) \\
P(G = K \mid X = x) \left[1 + \sum_{\ell=1}^{K-1} \exp\left(\beta_{\ell 0} + \beta_\ell^T x\right)\right] = 1 \\
P(G = K \mid X = x) &= \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp\left(\beta_{\ell 0} + \beta_\ell^T x\right)}
\end{aligned}$$

and the probability for any class k:

$$\begin{aligned}
P(G = k \mid X = x) &= P(G = K \mid X = x) \exp\left(\beta_{k0} + \beta_k^T x\right) \\
P(G = k \mid X = x) &= \frac{\exp\left(\beta_{k0} + \beta_k^T x\right)}{1 + \sum_{\ell=1}^{K-1} \exp\left(\beta_{\ell 0} + \beta_\ell^T x\right)}
\end{aligned}$$

These probabilities sum to one as expected. The model uses the last class K as the denominator in the odds-ratios, however, the choice of the class to use as the denominator is arbitrary in that the estimates don't change by this choice. To emphasize the dependence on the entire parameter set $\theta = \{\beta_{10}, \beta_1^T, \ldots, \beta_{(K-1)0}, \beta_{K-1}^T\}$, we denote the probabilities $P(G = k \mid X = x) = p_k(x; \theta)$.

There is no closed form solution for the coefficients θ, so logistic regression models are usually fit by maximum likelihood, using the conditional likelihood of G given X. Since $P(G \mid X)$ completely specifies the conditional distribution, the *multinomial* distribution is appropriate. The log-likelihood for N observations is

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

, where $p_k(x_i; \theta) = P(G = k \mid X = x_i; \theta)$. Taking the two-class case, since the algorithms simplify considerably. Code the two-class g_i via a $0/1$ response y_i, where $y_i = 1$ when $g_i = 1$, and $y_i = 0$ when $g_i = 2$. Let $p_1(x; \theta) = p(x; \theta)$, and $p_2(x; \theta) = 1 - p(x; \theta)$. The log-likelihood can be written:

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N} \{y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))\} \\
&= \sum_{i=1}^{N} \left\{ y_i \log \frac{p(x_i; \beta)}{1 - p(x_i; \beta)} + \log\left(1 - p(x_i; \beta)\right) \right\} \\
&= \sum_{i=1}^{N} \left\{ y_i \beta^T x_i + \log\left(1 - \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}\right) \right\} \\
&= \sum_{i=1}^{N} \left\{ y_i \beta^T x_i + \log\left(\frac{1}{1 + e^{\beta^T x_i}}\right) \right\} \\
&= \sum_{i=1}^{N} \left\{ y_i \beta^T x_i - \log\left(1 + e^{\beta^T x_i}\right) \right\}
\end{aligned}$$

To maximize log-likelihood, take the derivative to get the *score* equations, and set to zero.

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N} \left\{ y_i \beta^T x_i - \log\left(1 + e^{\beta^T x_i}\right) \right\} \\
\frac{\partial \ell(\beta)}{\partial \beta} &= \sum_{i=1}^{N} \left\{ y_i x_i - \frac{1}{1 + e^{\beta^T x_i}} \left(e^{\beta^T x_i}\right)(x_i) \right\} \\
\frac{\partial \ell(\beta)}{\partial \beta} &= \sum_{i=1}^{N} x_i \left(y_i - p(x_i;\beta) \right) = 0
\end{aligned}$$

These are $p+1$ equations *nonlinear* in β. Since there's no closed form solution, we solve it numerically, using something called the Newton-Raphson algorithm, which requires the second derivative or Hessian matrix:

$$\begin{aligned}
\frac{\partial \ell(\beta)}{\partial \beta} &= \sum_{i=1}^{N} x_i \left(y_i - \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \right) \\
\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} &= \sum_{i=1}^{N} x_i \left(\frac{x_i e^{\beta^T x_i}\left(1 + e^{\beta^T x_i}\right) - x_i e^{\beta^T x_i}\left(e^{\beta^T x_i}\right)}{\left(1 + e^{\beta^T x_i}\right)^2} \right) \\
&= \sum_{i=1}^{N} x_i x_i^T \left(\frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} + \left(\frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \right)^2 \right) \\
&= \sum_{i=1}^{N} x_i x_i^T \left(p(x_i;\beta) + p^2(x_i;\beta) \right) \\
&= -\sum_{i=1}^{N} x_i x_i^T p(x_i;\beta)\left(1 - p(x_i;\beta)\right)
\end{aligned}$$

Starting with a guess β^{old}, a single Newton update is

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta}$$

, where the derivatives are evalued at β^{old}.

The score and the Hessian can be written conveniently in matrix notation. Let $\mathbf{y}$ denote the vector of y_i values, $\mathbf{X}$ the $N \times (p+1)$ matrix of x_i values, $\mathbf{p}$ the vector of fitted probabilities with ith element $p(x_i;\beta^{\text{old}})$ and $\mathbf{W}$ an $N \times N$ diagonal matrix of weights with ith diagonal element $p(x_i;\beta^{\text{old}})(1 - p(x_i;\beta^{\text{old}}))$. Then we have:

$$\begin{aligned}
\frac{\partial \ell(\beta)}{\partial \beta} &= \mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\
\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} &= -\mathbf{X}^T \mathbf{W} \mathbf{X}
\end{aligned}$$

The Newton step is thus:

$$\begin{aligned}
\beta^{\text{new}} &= \beta^{\text{old}} + \left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\
&= \left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{W} \left(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}) \right) \\
&= \left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{W} \mathbf{z}
\end{aligned}$$

Rewriting it this way re-expresses the Newton step as a weighted least squares step, with the response

$$\mathbf{z} = \mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$$

sometimes known as the *adjusted response*. This algorithm is known as ***iteratively reweighted least squares*** or IRLS, since each iteration solves the weighted least squares problem:

$$\beta^{\text{new}} \leftarrow \underset{\beta}{\text{argmin}} \left(\mathbf{z} - \mathbf{X}\beta\right) \mathbf{W} \left(\mathbf{z} - \mathbf{X}\beta\right)$$

4.5 Separating Hyperplanes

Separating hyperplane classifiers construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible. They provide the basis for ***support vector classifiers*** (Ch. 12).

A brief review of vector algebra.

Consider a hyperplane or *affine set* L defined by $f(x) = \beta_0 + \beta^T x = 0$. (in $\mathbb{R}^2$, this is a line.)

Some properties:

1. For any two points x_1, x_2 lying in L:

$$\begin{aligned} \beta_0 + \beta^T x_1 = \beta_0 + \beta^T x_2 &= 0 \quad \text{by definition of } L \\ \beta^T x_1 &= \beta^T x_2 \\ \beta^T x_1 - \beta^T x_2 &= 0 \\ \beta^T \left(x_1 - x_2\right) &= 0 \end{aligned}$$

   - $\beta^T \left(x_1 - x_2\right) = 0$ implies that the vector β is orthogonal to the vector $\left(x_1 - x_2\right)$ on the surface of L.
   - Hence, $\beta^* = \frac{\beta}{\|\beta\|}$ is the vector normal to the surface of L.

2. For any point $x_0 \in L$, $\beta^T x_0 = -\beta_0$. (This follows directly from the definition of L.)
3. The signed distance of any point x to L is given by

$$\begin{aligned} &\beta^{*T} \left(x - x_0\right) \qquad \text{projection of the vector } x - x_0 \text{ in the direction of } \beta^* \\ &= \frac{1}{\|\beta\|} \left(\beta^T x + \beta_0\right) \\ &= \frac{1}{\|f'(x)\|} f(x). \end{aligned}$$

Hence, $f(x)$ is proportional to the signed distance from x to the hyperplane defined by $f(x) = 0$.

4.5.1 Rosenblatt's Perceptron Learning Algorithm

Perceptron learning algorithm - find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

- if response $y_i = 1$ is misclassified, then $x_i^T \beta + \beta_0 < 0$
- if response $y_i = -1$ is misclassified, then $x_i^T \beta + \beta_0 > 0$.

Therefore, the goal can be stated as minimizing

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i \left(x_i^T \beta + \beta_0\right),$$

where $\mathcal{M}$ indexes the set of misclassified points.

$D(\beta, \beta_0)$ is:

- non-negative
- proportional to the distance of the misclassified points to the decision boundary $\beta^T x + \beta_0$.

The gradient (assuming $\mathcal{M}$ is fixed):

$$\nabla_\beta D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i x_i \qquad \nabla_{\beta_0} D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i$$

The perceptron algorithm uses ***stochastic gradient descent*** to minimize $D(\beta, \beta_0)$, i.e. the parameters are updated after visiting each observation rather than the entire training set. The update rule:

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \leftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}$$

, where ρ is the learning rate.

Problems with the perceptron algorithm:

- When the data are separable, there are many solutions, and the chosen solution depends on the initial values
- Although guaranteed to be finite, the number of steps can be very large. Smaller gaps require longer time to find.
- When the data are not separable, the algorithm will not converge, and cycles develop which can be hard to detect.

Optimal Separating Hyperplanes

Optimal separating hyperplane- separate two classes and maximize the distance to the closest point from either class

- provides a unique solution
- better performance on test data by maximizing the margin between classes

The optimization problem:

$$\max_{\beta, \beta_0, \|\beta\|=1} M \text{subject to } y_i(x_i^T \beta + \beta_0) \geq M, i = 1, \dots, N.$$

This is equivalent to:

$$\frac{1}{\|\beta\|} y_i \left(x_i^T \beta + \beta_0 \right) \geq M \qquad \text{note that this redefines } \beta_0$$
$$y_i \left(x_i^T \beta + \beta_0 \right) \geq M \|\beta\|$$

Since for any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, we can choose $\|\beta\| = \frac{1}{M}$. Then,

$$\min_{\beta, \beta_0} = \frac{1}{2} \|\beta\|^2 \text{subject to } y_i \left(x_i^T \beta + \beta_0 \right) \geq 1, \quad i = 1, \dots, N.$$

These constrains define an empty slab or margin around the linear decision boundary of thickness of $\frac{1}{\|\beta\|}$. Hence we choose β and β_0 to maximize its thickness. This is a convex optimization problem

?????the math got very hard, finish this later