Machine Learning
Lecture 11

Lecture by Dr. Shai Fine
Typeset by Steven Karas

2018-01-07
Last edited 21:03:13 2018-01-07

Disclaimer These lecture notes are based on the lecture for the course Machine Learning, taught by Dr. Shai Fine at IDC Herzliyah in the fall semester of 2017/2018. Sections may be based on the lecture slides written by Dr. Shai Fine.

Agenda

- Finish up Bayesian models
- Support Vector Machines

1 Linear Classification Models

Partitions the feature space as a linear space with linear decision boundaries.

1.1 Model Discriminant Function

For each class $k \in \Omega = \{\omega_1, \dots, \omega_K\}$, we define the discriminant function as:

$$\delta_k(x) : x \to \Omega$$

Classification is done by taking the discriminant function with maximal value:

$$g^*(x) = \arg\max_{\omega_k \in \Omega} \delta_k(x)$$

Note that the decision boundaries between classes is given by[1]:

$$\{x \mid \delta_k(x) = \delta_{k'}(x)\}$$

1.2 Probabilistic Classification

Given a discriminant function, we denote the conditional probability of x given the class ω_k as $\Pr(X = x \mid \omega_k)$, which we call the likelihood. We denote the posterior probability of the class ω_k given x as $\Pr(\omega_k \mid X = x)$.

Generally speaking, we will require that a monotone transformation of δ_k is linear.

We utilize the *logit* transformation:

$$\log\left[\frac{p}{1-p}\right] = \beta_0 + \beta^\intercal x$$

After applying this transformation, the decision boundaries are the set of points with log-odds of 0, and these are hyperplanes defined as $\{x \mid \beta_0 + \beta^\intercal x = 0\}$

There are two popular methods that use this approach: LDA, and logistic regression.

2 Linear Discriminant Analysis (LDA)

Performs dimensionality reduction while preserving class information. We want to find the direction that preserves class separation as much as possible.

[1]Forms a hyperplane for linear discriminant functions

Bayesian Deriviation From the prior probability of class ω_k as $\Pr(\omega_k)$ and the class conditional probability $\Pr(x \mid \omega_k)$ we get the posterior probabilty via Bayes Theorem:

$$\Pr(\omega_k \mid x) = \frac{\Pr(x \mid \omega_k)\Pr(\omega_k)}{\sum_{l=1}^{K} \Pr(x \mid \omega_l)\Pr(\omega_l)}$$

Set the class conditional probability of ω_k to a multivariate Gaussian:

$$\Pr(x \mid \omega_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_k)^\intercal \frac{1}{\Sigma_k}(x-\mu_k)\right]$$

LDA arises from common covariance between the classes $\Sigma_k = \Sigma \; \forall k$.

Degenerate Case: 2 classes It's sufficient to consider the log-odds in this case, which is linear in x:

$$\log \frac{\Pr(\omega_k \mid x)}{\Pr(\omega_l \mid x)} = \log \frac{\Pr(\omega_k)}{\Pr(\omega_l)} - \frac{1}{2}(\mu_k+\mu_l)^\intercal \frac{1}{\Sigma}(\mu_k-\mu_l) + x^\intercal \frac{1}{\Sigma}(\mu_k-\mu_l)$$

2.1 LDA with Feature Independence

Assume that $\Pr(x \mid \omega_k) = \prod_{i=1}^{d} \Pr(x_i \mid \omega_k)$ where $\Pr(x_i \mid \omega_k) \sim \mathcal{N}(x_i \mid \mu_{i,k}, \sigma^2)$

More details are on slide 9.

2.2 LDA as a Discriminant Function

Define the linear discriminant function for each class k:

$$\delta_k(x) = x^\intercal \frac{1}{\Sigma}\mu_k - \frac{1}{2}\mu_k^\intercal \frac{1}{\Sigma}\mu_k + \log \Pr(\omega_k)$$

Classify instances according to the largest value for $\delta_k(x)$:

$$g^*(x) = \arg\max_{\omega_k \in \Omega} \delta_k(x)$$

In practice, we don't have the parameters of the Gaussians, so we estimate them using the training set:

$$\begin{aligned}
\widehat{\Pr(\omega_k)} &= \frac{n_k}{n} \\
\widehat{\mu}_k &= \sum_{g_i=k} \frac{x_i}{n_k} \\
\widehat{\Sigma} &= \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^\intercal/(N-K)
\end{aligned}$$

Note that we also need the class independent covariance, which is given on slide 11.

3 Quadratic Discriminant Analysis (QDA)

If the covariance matrices Σ_k are not equal, then the convenient cancellations from the log-odds equation do not occur. In particular, the quadratic terms w.r.t. x remain.

This gives us quadratic discriminant functions:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^\intercal \frac{1}{\Sigma_k}(x-\mu_k) + \log \Pr(\omega_k)$$

Parameter estimates are similar to LDA, but each class has different covariance matrices.

This approach produces quadratic decision boundaries in the feature space, but for higher dimensions d, the number of parameters increases from $(K-1)(d+1)$ parameters for LDA to $(K-1)(\frac{d(d+1)}{2}+d+1)$ parameters for QDA. As a result, for the same error, we need significantly more data.

More details can be found on slide 12.

4 Regularized Discriminant Analysis (RDA)

A compromise approach between LDA and QDA. Shrinks the covariances from QDA towards a common covariance in LDA:

$$\widehat{\Sigma}_k(\alpha) = \alpha\widehat{\Sigma}_k + (1-\alpha)\widehat{\Sigma}$$

Where $\widehat{\Sigma}$ is the pooled covariance and $\alpha \in [0, 1]$.

5 Logistic Regression

Details on slide 14.

Comparison with LDA LDA is supposed to be 30% more efficient, but only if normality holds. In practice, they are equivalent, but logistic regression is robust to non-normal distributions.

5.1 K-class Logistic Regression

Details on slide 16.

6 Support Vector Machines (SVM)

Considered to be one of the best general purpose classifiers are SVMs.

Built on top of 3 old ideas: Max Margin, Soft Margin, and the Kernel Trick.

Vapnik put together these approaches and invented SVMs.

6.1 Max Margin

Selection of the separating hyperplane that maximizes the margin between classes.

Given a data set $\{x_1, \ldots, .x_n\}$. Let $y_i \in \{-1, 1\}$ be the class label of x_i.

The decision boundary should classify all points:

$$\forall i, \quad y_i(w^\intercal x_i + b) \geq 1$$

We can find the parameters of the decision boundary w, b by minimizing the norm: $\frac{1}{2}||w||^2$ subject to the above.

The quadratic optimization with linear constraints has a unique solution when feasible, and has a closed form:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

The classifier is defined as:

$$f_{w,b}(z) = sign\left(\sum_i \alpha_i y_i x_i^\intercal z + b\right)$$

6.2 Soft Margin

If we relax the assumption that the data sets are linearly separable, we can allow for violations and penalize for the margin violation ξ_i for instance i.

$\sum_i \xi_i$ is the magnitude of the margin violations.

Hinge loss 0-1 loss is nonconvex and difficult to optimize.

So we use Hinge Loss, which is defined as:

$$\xi_i = (1 - y_i[w^\intercal x_i + b])_+ = \max(0, 1 - y_i[w^\intercal x_i + b])$$

We can formulate this as a quadratically constrained optimization problem. Minimize $\frac{1}{2}||w||^2 + C\sum_i \xi_i$ subject to $y_i(w^\intercal x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$.

The optimization problem is similar:

$$w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j}$$

Except that in this case, there's an upper bound C on the αs: $0 \geq \alpha < C$.

Note that the support vectors (that lie on the margin) have some weight. The outliers (those points that fall within the margin) have weight C.

Choosing C A smaller C gives a larger margin, and lower model complexity. A larger C is less tolerant to violations, but may overfit.

Typically start with $C = 1$ and do a log-scale search $0.01, 0.1, 10, 100$.

Further search is typically unnecessary.

6.3 Kernel Trick

The Dual SVM optimization problem is:

$$\max W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \overbrace{x_i^\intercal x_j}^{\text{dot product}}$$

Subject to:

$$C \geq \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

If we calculate the inner product in the higher dimension, we don't need an explicit mapping:

$$K(x_i, x_j) = \phi(x_i)^\intercal \phi(x_j)$$

Example Let $\phi(\cdot)$ be given as:

$$\phi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = (1, \sqrt{2}u_1, \sqrt{2}u_2, \sqrt{2}u_1u_2, u_1^2, u_2^2)$$

The inner product in this higher dimension is:

$$\left\langle \phi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}\right) \right\rangle = (1 + u_1v_1 + u_2v_2)^2$$

Define the 2nd degree polynomial kernel function:

$$K(u, v) = (1 + u^\intercal v)^2$$

There is no need to compute $\phi(u)$ or $\phi(v)$ explicitly.

This trick is called the kernel trick.

Examples of Kernel Functions Details on slide 17.

7 Next week

- ???