6.867 Section 3: Classification

Contents

1 Intro

Now we will look at the supervised learning problem of *classification*, in which the data given is a set of pairs

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\} \ ,$$

with $x^{(i)} \in \mathbb{R}^D$ and $y^{(i)} \in \{c_1, \ldots, c_k\}$; that is, the output is a discrete value indicating the *class* of the example. By default, we'll think about the two-class classification problem, with the *classes* or *labels* often drawn from the set $\{0, 1\}$ or $\{+1, -1\}$.

We'll look at learning prediction rules, probabilistic models, and distributions over models, and focus on the case in which the model represent a linear relationship between inputs and outputs. In later parts of the course, we will return to classification and look at non-linear and non-parametric approaches.

Because we have recently been looking at probabilistic approaches to regression, we will begin by looking at probabilistic and Bayesian approaches to classification. When we are done with that, we will consider some interesting strategies for finding linear separators directly, without probabilistic modeling.

2 Representation

What does it mean to have a linear model for classification? Generally, it will be that we can express the output value $y^{(i)}$ by specifying a $D-1$-dimensional *hyperplane* in the D-dimensional feature space. Then, points that are on one side of the hyperplane are considered to be in one class, points on the other side, in the other class.

That is, there are some weight values w_0 and $w = (w_1, \ldots, w_D)$ such that

$$y = \begin{cases} +1 & \text{if } w_0 + w_1 x_1 + \ldots + w_D x_D > 0 \\ -1 & \text{otherwise} \end{cases} \ .$$

Such a model is known as a *linear separator*.

As in regression, we can transform the input space, via a set of non-linear basis functions, and find a linear separator in the transformed space. Such a separator will be non-linear when projected back down into the original space.

In the following, to simplify notation, we will omit the possibility of using basis function to transform the input values, but the extension is completely straightforward.

3 Probabilistic models

	No model	Prediction rule	Prob model	Dist over models
Classification			❋	

3.1 Estimating $\Pr(X, Y)$

3.1.1 Linear discriminant analysis

If we're going to estimate the joint distribution, we need to make some distributional assumptions. A common model is:

$$\begin{aligned} Y &\sim \text{Binomial}(\pi) \\ X \mid Y = c &\sim \text{Gaussian}(\mu_c, \Sigma_c) \end{aligned}$$

where π_c specifies the unconditional probability of getting an object of class c, and π is a vector of π_c for the possible class values c. We will use θ to name all of the parameters: π, μ, Σ.

We can find the maximum-likelihood parameter estimates for this model straightforwardly. Letting $n_c = |\{y^{(i)} = c \mid i = 1 \dots n\}|$ be the number of examples in $\mathcal{D}$ of class c, we have:

$$\begin{aligned} \pi_c &= \frac{n_c}{n} \\ \mu_c &= \frac{1}{n_c} \sum_{\{i|y^{(i)}=c\}} x^{(i)} \\ \Sigma_c &= \frac{1}{n_c} \sum_{\{i|y^{(i)}=c\}} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T \end{aligned}$$

Now, how should we make predictions? If we have 0-1 loss,

$$h(x) = \begin{cases} 1 & \text{if } \Pr(Y = 1 \mid X = x; \theta) > \Pr(Y = 0 \mid X = x; \theta) \\ 0 & \text{otherwise} \end{cases}$$

Let's concentrate on the conditions under which we predict 1:

$$\begin{aligned} \Pr(Y = 1 \mid X = x; \theta) &> \Pr(Y = 0 \mid X = x; \theta) \\ \Pr(X = x \mid Y = 1; \theta) \Pr(Y = 1; \theta) &> \Pr(X = x \mid Y = 0; \theta) \Pr(Y = 0; \theta) \\ \log \Pr(X = x \mid Y = 1; \theta) + \log \Pr(Y = 1; \theta) &> \log \Pr(X = x \mid Y = 0; \theta) + \log \Pr(Y = 0; \theta) \\ \log \Pr(X = x \mid Y = 1; \theta) - \log \Pr(X = x \mid Y = 0; \theta) &> \log \Pr(Y = 0; \theta) - \log \Pr(Y = 1; \theta) \\ -\frac{1}{2} \log \det(\Sigma_1) - \frac{1}{2}(x - \mu_1)\Sigma_1^{-1}(x - \mu_1)^T & \\ +\frac{1}{2} \log \det(\Sigma_0) + \frac{1}{2}(x - \mu_0)\Sigma_0^{-1}(x - \mu_0)^T &> \log \pi_0 - \log \pi_1 \\ -(x - \mu_1)\Sigma_1^{-1}(x - \mu_1)^T + (x - \mu_0)\Sigma_0^{-1}(x - \mu_0)^T &> 2(\log \pi_0 - \log \pi_1) + \log \det(\Sigma_1) - \log \det(\Sigma_0) \end{aligned}$$

This is pretty clearly quadratic in x, so it's what we would call a *quadratic separator* or *quadratic discriminant*.

If we assume that the covariances of the two classes are equal, $\Sigma_1 = \Sigma_0$, then things simplify and we are doing *linear discriminant analysis.* Now we have that (determinant of Σ cancels):

$$\Pr(Y = c \mid X = x) \propto \exp\left(\mu_c^T \Sigma^{-1} x - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c\right) \exp\left(-x^T \Sigma^{-1} x\right) \quad .$$

Define

$$\begin{aligned} \beta_c &= \Sigma^{-1}\mu_c \\ \gamma_c &= -\frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c \end{aligned}$$

Then

$$\begin{aligned} \Pr(Y = c \mid X = x) &= \frac{\exp\left(\beta_c^T x + \gamma_c\right) \exp\left(x^T \Sigma^{-1} x\right)}{\sum_{c'} \exp\left(\beta_{c'}^T x + \gamma_{c'}\right) \exp\left(x^T \Sigma^{-1} x\right)} \\ &= \frac{\exp\left(\beta_c^T x + \gamma_c\right)}{\sum_{c'} \exp\left(\beta_{c'}^T x + \gamma_{c'}\right)} \end{aligned}$$

Note that, in the two class case, this reduces to

$$\begin{aligned}\Pr(Y=1 \mid X=x) &= \frac{\exp\left(\beta_1^T x+\gamma_1\right)}{\exp\left(\beta_0^T x+\gamma_0\right)+\exp\left(\beta_1^T x+\gamma_1\right)} \\ &= \frac{1}{\exp\left(\beta_0^T x+\gamma_0-\beta_1^T x-\gamma_1\right)+1} \\ &= \text{sigmoid}(\beta_1^T x+\gamma_1-\beta_0^T x-\gamma_0)\end{aligned}$$

Bishop uses $\sigma(.)$ for the sigmoid function, but I find that just too confusing in this context, so I'll write out the name.

where the *sigmoid* function and its inverse, the *logit* function are defined as follows:

$$\begin{aligned}\text{sigmoid}(a) &= \frac{1}{1+\exp(-a)} \\ &= \frac{\exp a}{1+\exp a} \\ \text{logit}(\sigma) &= \log\left(\frac{\sigma}{1-\sigma}\right) \quad .\end{aligned}$$

The sigmoid is a "soft" step function, which takes a real number and maps it to the interval $(0,1)$. If we have an expression of the form $\text{sigmoid}(W^T X)$, then the larger the magnitude of W, the steeper slope on the sigmoid.

So, with two classes, we predict class 1 when

$$\begin{aligned}\Pr(Y=1 \mid X=x;\theta) &> \Pr(Y=0 \mid X=x;\theta) \\ \exp\left(\beta_1^T x+\gamma_1\right) &> \exp\left(\beta_0^T x+\gamma_0\right) \\ \beta_1^T x+\gamma_1 &> \beta_0^T x+\gamma_0 \\ x\left(\beta_1-\beta_0\right)+\gamma_1-\gamma_0 &> 0\end{aligned}$$

This is definitely a linear separator.

3.1.2 Factoring the class conditional probability

This is a method called 'Naive Bayes'. Let's assume now that $x^{(i)} \in \{0,1\}^D$. You could interpret the X values as numbers, encoded in binary, and put a multinomial distribution over all 2^D values for each class. But that is a lot of parameters to estimate!

Assume: Features are independent, given the class. That is, that

$$\Pr(X \mid Y=c)=\prod_{j=1}^{D}\Pr(X_j \mid Y=c) \quad .$$

So

$$\begin{aligned}Y &\sim \text{Binomial}(\pi) \\ X_j \mid Y=c &\sim \text{Binomial}(\theta_{c,j})\end{aligned}$$

So, now, if we have two classes, we have 2D parameters, each of which is easy to estimate:

$$\begin{aligned}\Pr(X_j=1 \mid Y=1) &= \theta_{1,j} \\ \Pr(X_j=1 \mid Y=0) &= \theta_{0,j}\end{aligned}$$

Use Bernoulli ML estimate, maybe with Laplace "correction":

$$\widehat{\theta}_{1j}=\frac{\#(X_j=1,Y=1)+1}{\#(Y=1)+2} \quad .$$

Now, for prediction. Given x, predict $C = 1$ if

$$\begin{aligned}
\Pr(x \mid C=1)\Pr(C=1) &> \Pr(x \mid C=0)\Pr(C=0) \\
\prod_{j=1}^{D} \Pr(x_j \mid C=1)\Pr(C=1) &> \prod_{j=1}^{D} \Pr(x_j \mid C=0)\Pr(C=0) \\
\sum_{j=1}^{D} \log \Pr(x_j \mid C=1) + \log \Pr(C=1) &> \sum_{j=1}^{D} \log \Pr(x_j \mid C=0) + \log \Pr(C=0) \\
\sum_{j=1}^{D} \log(\theta_{1j}^{x_j}(1-\theta_{1j})^{(1-x_j)}) + \log \pi_1 &> \sum_{j=1}^{D} \log(\theta_{0j}^{x_j}(1-\theta_{0j})^{(1-x_j)}) + \log \pi_0 \\
\sum_{j=1}^{D} (x_j \log \theta_{1j} + (1-x_j)\log(1-\theta_{1j})) + \log \pi_1 &> \sum_{j=1}^{D} (x_j \log \theta_{0j} + (1-x_j)\log(1-\theta_{0j})) + \log \pi_0 \\
\sum_{j=1}^{D} x_j \left(\log \frac{\theta_{1j}}{(1-\theta_{1j})} - \log \frac{\theta_{0j}}{1-\theta_{0j}} \right) &> \log \pi_0 - \log \pi_1 - \sum_{j=1}^{D} \log \frac{1-\theta_{1j}}{1-\theta_{0j}}
\end{aligned}$$

So, this is a linear separator of the form $x^T w + w_0 > 0$ with

$$W_j = \log \frac{\theta_{1j}}{1-\theta_{1j}} - \log \frac{\theta_{0j}}{1-\theta_{0j}} \ ,$$

and

$$W_0 = \log \frac{\pi_1}{\pi_0} + \sum_{j=1}^{n} \frac{\log(1-\theta_{1j})}{\log(1-\theta_{0j})} \ .$$

Interestingly, the probability model is also sigmoidal. We have

$$\begin{aligned}
\Pr(C=1 \mid X=x) &= \frac{\Pr(x \mid C=1)\Pr(C=1)}{\Pr(x \mid C=1)\Pr(C=1) + \Pr(x \mid C=0)\Pr(C=0)} \\
&= \frac{\exp(f_1(x))}{\exp(f_1(x)) + \exp(f_2(x))} \\
&= \frac{1}{1+\exp(f_2(x) - f_1(x))} \\
&= \text{sigmoid}(f_1(x) - f_2(x))
\end{aligned}$$

where

$$f_c(x) = \sum_{j=1}^{D} (x_j \log \theta_{cj} + (1-x_j)\log(1-\theta_{cj})) + \log \pi_c \ .$$

3.1.3 Exponential family

A really cool family of distributions.

- Only family for which conjugate priors exist
- Finite-size sufficient statistics
- Includes Normal, Bernoulli, Multinomial, Poisson, Gamma, Exponential, Beta, Dirichlet, various combinations

A (somewhat simplified) subset of exponential-family distributions can be written in the form:

$$\Pr(x \mid \eta) = h(x)g(\eta)\exp(\eta^T u(x))$$

where x may be a scalar or vector, discrete or continuous; η is called the *natural parameters*, and $g(\eta)$ is a normalization constant.

3.1.3.1 Binomial is in exponential family

$$\begin{aligned}\Pr(x \mid p) &= p^x(1-p)^{(1-x)} \\ &= \exp\left(x\log(p) + (1-x)\log(1-p)\right) \\ &= \exp\left(x\log(p) - x\log(1-p) + \log(1-p)\right) \\ &= (1-p)\exp\left(x\log\frac{p}{1-p}\right)\end{aligned}$$

This fits in the family with: $u(x) = x$, $\eta = \log\frac{p}{1-p}$, $h(x) = 1$, and $g(\eta) = 1 - p$. So

$$\Pr(x \mid \eta) = \text{sigmoid}(-\eta)\exp(\eta x) \quad .$$

3.1.3.2 Normal is in exponential family

We'll just look at the one-dimensional case, but it's true for multi-variate Gaussian as well.

$$\begin{aligned}\Pr(x \mid \mu, \sigma^2) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2\right)\end{aligned}$$

To make this match up, let

$$\begin{aligned}\eta &= \begin{pmatrix}\mu/\sigma^2 \\ -1/2\sigma^2\end{pmatrix} \\ u(x) &= \begin{pmatrix}x \\ x^2\end{pmatrix} \\ h(x) &= (2\pi)^{1/2} \\ g(\eta) &= \sqrt{-2\eta_2}\exp\left(\frac{\eta_1^2}{4\eta_2}\right).\end{aligned}$$

3.1.3.3 LDA for exponential family

Cool result! If we have two classes, and $\Pr(X \mid Y = c)$ is a distribution in the exponential family, with the restrictions that $u(x) = x$, and that the scale parameters are shared among the classes (e.g., the covariance in the Gaussian case) so that the form is

$$\Pr(x \mid \lambda_c, s) = \frac{1}{s}h\left(\frac{1}{s}x\right)g(\lambda_c)\exp\left(\frac{1}{s}\lambda_c^T x\right) \quad ,$$

then

- The separator is linear and
- The predictive probability $\Pr(Y = 1 \mid x)$ is a sigmoid on an activation function which is the difference between the logs of the class membership probabilities of the two classes. That is,

  $$\Pr(Y = 1 \mid x) = \text{sigmoid}(a(x)) \quad ,$$

  where

  $$a(x) = (\lambda_1 - \lambda_0)^T x + \log g(\lambda_1) - \log g(\lambda_0) + \log\pi_1 - \log\pi_0$$

3.2 Estimating $\Pr(Y \mid X)$

This is called estimating a *discriminative model*.

3.2.1 Least squares

A favorite trick is to reduce our current problem to a previous problem we already know how to solve. In this case, we could try to treat classification as a regression problem, by taking the Y values to be in $\{+1, -1\}$ and applying one of our standard regression methods. We could then predict class $+1$, given x, if

$$W^T x + W_0 > 0 \ .$$

That will generate a separator, but it turn out to be a bad idea. There is generally no hypothesis that does a good job of representing the data, and it is easy to show that there are situations in which a linear separator for the data exists, but the hypothesis that regression comes up with does not separate the data.

One reason that this doesn't work out is that it isn't founded on a sensible probabilistic model: the algorithm fundamentally assumes that the Y values are normally distributed, but they're not.

3.2.2 Logistic regression

So, what assumption can we reasonably make about the distribution of outputs, in the classification case? Rather than directly trying to predict the value, we could try to find a regression model that predicts $\Pr(Y = 1 \mid X = x)$ as a function of x. One thought would be to use the form

$$\Pr(Y = 1 \mid X = x) = W^T x \ ,$$

but the problem is that we need the probabilities to be in the interval $[0, 1]$, and that linear form is unconstrained.

We can take inspiration from what we saw in the LDA case: that in at least two cases we considered, the predictive distribution could be described as a sigmoid applied to a dot product. So, let's consider models of the form

$$\Pr(Y = 1 \mid X = x) = \text{sigmoid}(W^T x) \ .$$

Which is confusing, since we're using them for classification...but the idea is that we're doing a regression to find a probability value.

Such models are called *logistic regression* models.

So, how does this differ from LDA? We are not making any distributional assumptions about X. So, we're going to train a predictive model from the same class, but using a different criterion.

Given data $\mathcal{D}$, what are the maximum-likelihood estimates of the parameters W? Assuming $y^{(i)} \in \{0, 1\}$, the likelihood function is:

$$\Pr(\mathcal{D} \mid W) = \prod_{i=1}^{n} s\left(W^T x^{(i)}\right)^{y^{(i)}} \left(1 - s\left(W^T x^{(i)}\right)\right)^{(1-y^{(i)})} \ ,$$

Which we want to minimize.

where s is short for sigmoid. The negative log likelihood is:

$$\text{NLL}(W) = -\sum_{i=1}^{n} \left(y^{(i)} \log s\left(W^T x^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - s\left(W^T x^{(i)}\right)\right)\right) \ .$$

Here's the cool thing about the derivative of the sigmoid: $\frac{d}{dx} s(x) = s(x)(1 - s(x))$

The gradient with respect to the weights is

$$\nabla_W \text{NLL}(W) = \sum_{i=1}^{n} \left(s\left(W^T x^{(i)}\right) - y^{(i)}\right) x^{(i)} \ .$$

We'll see why in the next section.

This is the same as the gradient of the error function for the linear regression model!

If the data is linearly separable, the optimization will want to make the weights as large as possible (to make the sigmoid as steep as possible, which will push the output values closer and closer to $+1$ and -1. Furthermore, again, if the data are separable, there is an infinity of separators that are equally good; so the optimization problem is not well posed. This can be addressed by adding a regularization penalty or using a prior.

An old trick from neural network days was to use targets of $+0.6$ and -0.6, which are actually attainable with finite weights.

Unfortunately, there is no closed form solution for the maximum likelihood values of W, so we need to use an iterative optimization method. Two common choices are:

- *Gradient descent*, especially *stochastic gradient descent*, which goes through the data points one by one and does an update to the weights with a very small step size based on each individual point. This can be guaranteed to converge, though it might be slow, but it can be less likely than batch gradient descent to get stuck in local optima on non-convex functions.
- *Iterative reweighted least squares*, which is essentially Newton's method. It can be computationally challenging, because it uses the Hessian (matrix of second derivatives), but is more reliable than regular gradient descent.

3.2.3 Generalized linear models

Both linear regression and logistic regression are instances of a more general framework of *generalized linear models*:

- The distribution of $Y \mid X$ is an exponential family distribution
- The goal is to predict $E[Y \mid X; \theta]$
- The *natural parameter* η and the inputs are linearly related: $\eta = \theta^T x$.

What is cool about this is that any regression-type problem that satisfies the requirements above can be handled in common way. So, there are general purpose methods for

- Maximum likelihood parameter estimation (IRLS)
- Bayesian estimation
- Various statistical tests

This also means that these models are not too sensitive to the distributional assumptions made (beyond the general GLM assumptions), since the parameter-fitting is independent of which exponential-family distribution you have.

3.3 Generative versus discriminative

This discussion cribbed from *Machine Learning: A Probabilistic Perspective* by Kevin Murphy. (**A great book!**)

Given a choice, should we choose a generative or discriminative model?

- Generative classifiers are usually easier to fit.
- Generative classifiers, in the multi-class case, generalize more easily to adding a new class (since the per-class models are independent).
- Generative models can be estimated relatively easily in the presence of missing or unlabeled data.
- Generative models can be run "backwards" (used to predict X from Y).
- Discriminative models apply very well when we expand the input space using a basis set of feature functions; generative models can get into trouble due to correlation among the inputs.

- *If the distributional assumptions are true*, generative models can be well estimated with fewer training examples than discriminative models.
- *If the distributional assumptions are not true*, generative models can result in really bad predictions.

In the research literature, there is some interesting work on making models that are a kind of mixture of the generative and discriminative.

4 Distributions over models

	No model	Prediction rule	Prob model	Dist over models
Classification				❋

Rather than finding a single best weight vector W^* and using that to make predictions $\Pr(y \mid x; W^*)$, we might again wish to be Bayesian, by putting a prior on W and then using the posterior on W, that is $\Pr(W \mid \mathcal{D})$ to make predictions that take uncertainty in the weights into account.

Unfortunately, there is no conjugate prior for logistic regression. Here, we will show one strategy (that can be used in all sorts of cases with non conjugate prior, sometimes to good effect, sometimes less so) for approximating the posterior. The idea is to assume a Gaussian prior, compute the posterior, which is

$$\log \Pr(W \mid \mathcal{D}) = -\frac{1}{2}(W - \mathbf{m}_0)^T S_0^{-1}(W - \mathbf{m}_0) + \sum_{i=1}^{n} (y^{(i)} \log o^{(i)} + (1 - y^{(i)}) \log(1 - o^{(i)})) \ ,$$

where $o^{(i)} = \text{sigmoid}(W^T x^{(i)})$.

This clearly does not have the form of a Gaussian. So we want to find a Gaussian approximation to the posterior, which means finding parameters $\mathbf{m}_n$ and $\mathbf{S}_n$ that make a good approximation. Using a second-order Taylor-series expansion about the mode, we find that the MAP estimate of the weights, which under a simple diagonal Gaussian prior is

$$W_{\text{map}} = \arg\min_W \text{NLL}(W) + \lambda W^T W \ ,$$

the appropriate choice of $\mathbf{m}_n$.

The gradient of the negative log likelihood at the mode is:

$$\left.\frac{\partial \text{NLL}(W)}{\partial W}\right|_{W_{\text{MAP}}} = \mathbf{s}_0^{-1}(W - \mathbf{m}_0) - \sum_{i=1}^{n} (y^{(i)} - o^{(i)}) x^{(i)} \ ,$$

where output $o^{(i)} = \text{sigmoid}(x^{(i)T} W_{\text{MAP}})$.

The covariance matrix is then the inverse of the Hessian, which is a matrix of second derivatives of NLL. So,

$$\mathbf{s}_n^{-1} = H = \left.\frac{\partial^2 \text{NLL}(W)}{\partial W \partial W^T}\right|_{W_{\text{MAP}}} = \mathbf{s}_0^{-1} + \sum_{i=1}^{n} o^{(i)}(1 - o^{(i)}) x^{(i)} x^{(i)T} \ .$$

Going from here to a predictive distribution requires further approximation. We won't go into it in detail. However, there is one more useful concept to get out of this, which is the *Bayesian information criterion* or BIC. We can approximate

$$\log \Pr(\mathcal{D}) \approx \log \Pr(\mathcal{D} \mid W_{\text{MAP}}) + \log \Pr(W_{\text{MAP}}) + \frac{D}{2} \log(2\pi) - \frac{1}{2} \log |H| \ .$$

The second two terms, called the *Occam factor*, measure the complexity of the model. If the prior on weights is uniform, then we can simplify to

$$\log \Pr(\mathcal{D}) \approx \log \Pr(\mathcal{D} \mid W_{\text{ML}}) - \frac{1}{2} \log |H| \ .$$

The determinant of a covariance matrix can be seen informally as characterizing the "size" of the covariance ellipse: if it's large then we are more uncertain about the weight values, which means the marginal likelihood will be lower.

If we think of H as being a sum of H_i over the individual data points, and assume that they are approximable by single $\widehat{H}$, then

$$\begin{aligned} \log |H| &= \log |n\widehat{H}| \\ &= \log \left(n^D |\widehat{H}| \right) \\ &= D \log n + \log |\widehat{H}| \end{aligned}$$

We drop $\log |\widehat{H}|$ because it is independent of n and D, and wind up with a fairly gross approximation:

$$\log \Pr(\mathcal{D}) \approx \log \Pr(\mathcal{D} \mid W_{\text{ML}}) - \frac{D}{2} \log n \ .$$

This is the BIC score, which can be used as a quick-and-dirty way of selecting model complexity.

BIC is asymptotically consistent as a selection criterion: that is, given a set of models, as $n \to \infty$, BIC will select the correct model. It has a tendency to select models that are too simple, with small n, though.

5 Fitting predictors directly

	No model	Prediction rule	Prob model	Dist over models
Classification	❋			

Another strategy is to abandon the idea of fitting probabilistic models and then using decision theory to make decisions and, instead, to try to find a prediction rule directly. This strategy makes no distributional assumptions at all, and is generally unable to make probabilistic predictions. The fact that it doesn't make distributional assumptions means that it can be applied in a much broader variety of situations; but the lack of assumptions tends to mean that more data is required to obtain a predictor with good generalization properties.

We will restrict our attention to linear separators, again with the idea that you might want to transform your original input space by applying a basis set of feature-functions ϕ_j to the original input vector $x^{(i)}$.

5.1 Linear discriminant functions

A linear discriminant function has the form

$$y(x) = w^T x + w_0 \ ,$$

and we will select output 1 for input x if $y(x) \geqslant 0$, and output 0 otherwise. The decision boundary is the locus of points x for which $y(x) = 0$. It is a $D-1$-dimensional hyperplane in a D-dimensional space. Figure 1 illustrates a linear discriminant function in two dimensions.

Here are some important properties and definitions:

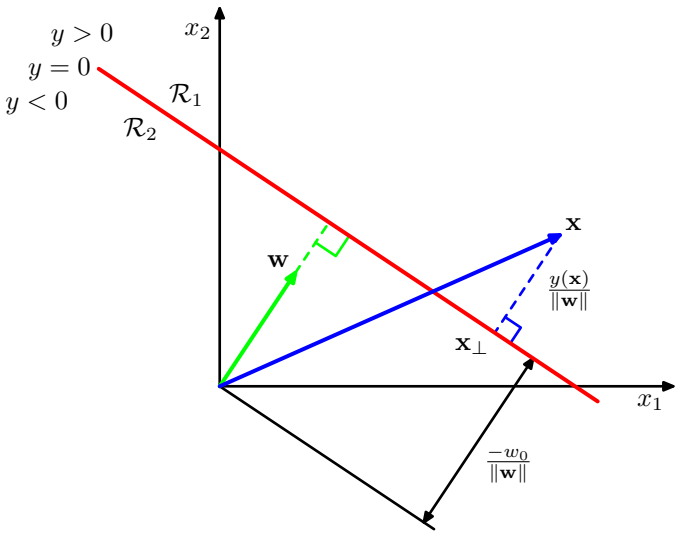

Figure 1: Geometry of linear discriminant functions (Bishop Figure 4.1).

- *The decision surface is orthogonal to w.*

  Consider two points on the decision surface, x_a and x_b; the vector from x_a to x_b lies within that surface. We know that $y(x_a) = y(x_b) = 0$, so $w^T(x_a - x_b) = 0$; so w is orthogonal to $x_a - x_b$.

- *The perpendicular distance from the origin to the decision surface is*

$$\frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|}$$

  We are interested in the magnitude of the projection of x, where x is on the decision surface, onto w. Let θ be the angle between x and w. Then the perpendicular distance is $\|x\| \cos\theta$. We'd like to express this in terms of dot products. We remember from high school that

$$w^T x = \|w\|\|x\| \cos\theta \quad .$$

  Combining these two facts, we can easily derive the perpendicular distance. Note that w_0 determines the location of the decision surface.

- *The signed distance r of a point x to the decision surface* is $(w^T x + w_0)/\|w\|$.

  Let $x_\perp$ be the orthogonal projection of x onto the decision surface. We know that $\frac{w}{\|w\|} r$ is the perpendicular vector from x to the decision surface (because $\frac{w}{\|w\|}$ is a unit vector in the right direction and r is its length). So

$$\begin{aligned} x &= x_\perp + r\frac{w}{\|w\|} \\ w^T x + w_0 &= w^T x_\perp + r w^T \frac{w}{\|w\|} + w_0 \\ \frac{w^T x + w_0}{\|w\|} &= r \end{aligned}$$

  The last step is because $w^T x_\perp + w_0 = 0$ (because $w_\perp$ is on the decision surface) .

- *The* **geometric margin** $\gamma^{(i)}$ *of a training example* $(x^{(i)}, y^{(i)})$ *is the distance of the point on the "right" side of the decision surface:*

$$\gamma^{(i)} = y^{(i)} \frac{w^T x^{(i)} + w_0}{\|w\|}$$

  It is positive if either:

  - $y^{(i)} = +1$ and $w^T x^{(i)} + w_0 > 0$
  - $y^{(i)} = -1$ and $w^T x^{(i)} + w_0 < 0$

- *The* **functional margin** $\hat{\gamma}^{(i)}$ *of a training example* $(x^{(i)}, y^{(i)})$ *is an unnormalized notion of margin:*

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + w_0)$$

5.2 Support vector machines for separable data

The idea here is that, if we're looking for a separator, it seems reasonable to pick one that is as far as possible from the closest points. That is to say, it *maximizes the margin* between the decision surface and the closest points. Two motivations:

- If there is some noise in the x values, we would like not to be too close to any of the training points.
- Imagine we use a mixture-of-Gaussians to represent $\Pr(X \mid Y = c)$, and we are very extreme, using one Gaussian per training example. As we let the variance go to 0, the optimal predictive hypothesis becomes the one that maximizes the margin. It is because it is only the closest points that have any influence.

5.2.1 Primal form

So, let's maximize the margin! That would mean finding

$$\arg\max_{w, w_0} \left(\frac{1}{\|w\|} \min_i \left(y^{(i)}(w^T x^{(i)} + w_0) \right) \right) \ .$$

Unfortunately, this isn't an instance of any kind of optimization problem we know how to deal with. So, we'll find a new formulation.

First, note that if we multiplied all of the values in w and w_0 by some constant c, the separator would be unchanged. So we really have an extra degree of freedom in the problem. Let's just, arbitrarily decide to set the maximum functional margin to be 1. That means, for the closest point, $x^{(c)}$, to the surface,

$$y^{(c)}(w^T x^{(c)} + w_0) = 1 \ ,$$

and so, for all points,

$$y^{(i)}(w^T x^{(i)} + w_0) \geqslant 1 \ . \tag{1}$$

A constraint (or its associated data point) is *active* if the constraint holds with equality, else it is *inactive*.

Subject to those constraints we would like to make $\|w\|$ as **small** as possible, since that will make the geometric margin bigger, which is what we really want to do. So, we'll solve the optimization problem

$$\arg\min_{w, w_0} \frac{1}{2}\|w\|^2 \ ,$$

subject to the constraints 1. We've changed the objective a bit, squaring the norm and dividing by two for later convenience.

This is good news. It is a *quadratic programming* problem, with a quadratic objective and linear constraints. There is software for doing this, and there is a single optimum so we can actually solve pretty reliably.

I'm following Andrew Ng's CS229 story here. Also look in Bishop appendix E.

Note that although w_0 isn't in the objective, it is in the constraints, and as we change w we will have to change w_0.

5.2.2 Dual form

Now, let's formulate this constrained optimization problem with Lagrange multipliers: we'll introduce $\alpha_1, \ldots, \alpha_n$, one for each constraint, yielding

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y^{(i)}(w^T x^{(i)} + w_0) - 1 \right) \quad .$$

We let

$$\theta_P(w) = \max_{\alpha : \alpha_i \geqslant 0} L(w, \alpha) \quad ;$$

If w violates any constraints then this value will be infinity; if it satisfies the constraints, it will just be the value of the original unconstrained objective. So, this problem

$$p^* = \min_w \theta_P(w) = \min_w \max_{\alpha : \alpha_i \geqslant 0} L(w, \alpha)$$

has the same solutions as the original, *primal* problem, with optimal *value* p^*.

Now, we're going to consider a different, *dual* problem. Define

$$\theta_D(\alpha) = \min_w L(w, \alpha) \quad .$$

The dual problem is

$$d^* = \max_{\alpha : \alpha_i \geqslant 0} \theta_D(w) = \max_{\alpha : \alpha_i \geqslant 0} \min_w L(w, \alpha) \quad .$$

That the objective and constraints be convex

In general, $d^* \leqslant p^*$, but under a set of conditions that are satisfied in this problem but we're not going to go into in detail, $d^* = p^* = L(w^*, \alpha^*)$. In addition, the KKT conditions are satisfied for all i:

$$\frac{\partial}{\partial w_i} L(w^*, \alpha^*) = 0 \tag{2}$$

$$\alpha_i^* \left(y^{(i)}(w^{*T} x^{(i)} + w_0^*) - 1 \right) = 0 \tag{3}$$

$$\left(y^{(i)}(w^{*T} x^{(i)} + w_0^*) - 1 \right) \geqslant 0 \tag{4}$$

$$\alpha_i \geqslant 0 \tag{5}$$

This means that, at every point, either $\alpha_i^* = 0$ or $\left(y^{(i)}(w^{*T} x^{(i)} + w_0^*) - 1 \right) = 0$; that is, that the constraint on point i is tight. We will call points for which the constraint is tight, *support vectors*.

So, let's find the dual form in detail. We need to find a function $\theta(\alpha) = \min_{w, w_0} L(w, w_0, \alpha)$ by setting derivatives with respect to w to 0 and solving for w in terms of α:

$$\nabla_w L(w, w_0, \alpha) = w - \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)} = 0 \quad .$$

So,

$$w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)} \ .$$

Also, we can take the derivative with respect to w_0 and set to 0 to get

$$\frac{\partial}{\partial w_0} L(w, w_0, \alpha) = \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \ .$$

Rewriting L as

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y^{(i)} w^T x^{(i)} - w_0 \sum_{i=1}^{n} \alpha_i y^{(i)} + \sum_{i=1}^{n} \alpha_i \ ,$$

plugging this definition of w in, and noticing that

$$\sum_{i=1}^{n} \alpha_i y^{(i)} w^T x^{(i)} = \|w\|^2 \ ,$$

we get

$$\begin{aligned} \theta_D(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)T} x^{(j)} - w_0 \sum_{i=1}^{n} \alpha_i y^{(i)} \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)T} x^{(j)} \end{aligned}$$

So, finally, we end up with the dual optimization problem:

$$\begin{gathered} \max_\alpha \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \left(x^{(i)T} x^{(j)} \right) \\ \text{subject to} \ \ \alpha_i \geqslant 0 \\ \textstyle\sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \end{gathered}$$

If we solve the dual problem for the α_i^*, then we can use them to compute w^*. Note that α_i^* will be 0 for all points that are not on the margin: so the weights can be computed only as a function of the support vectors (points with non-zero α_i). Let S be the set of indices of support vectors.

In addition, once we know α, we have our prediction rule:

$$y(x) = \sum_{i \in S} \alpha_i y^{(i)} \left(x^T x^{(i)} \right) + w_0 \ .$$

To compute w_0, note that, for any support vector, $x^{(j)}$, the functional margin is 1. So

$$\begin{aligned} y^{(j)} \left(\sum_{i \in S} \alpha_i y^{(i)} \left(x^{(j)T} x^{(i)} \right) + w_0 \right) &= 1 \\ w_0 &= \frac{1}{y^{(j)}} - \sum_{i \in S} \alpha_i y^{(i)} \left(x^{(j)T} x^{(i)} \right) \end{aligned}$$

Recall that $y = (1/y)$ because y is always either $+1$ or -1.

For more robustness (using all the support vectors), do this instead:

$$w_0 = \frac{1}{|S|} \sum_{j \in S} \left(y^{(j)} - \sum_{i \in S} \alpha_i y^{(i)} \left(x^{(j)T} x^{(i)} \right) \right) \ .$$

Critical point: In the optimization problem for finding α and in the resulting prediction rule, we depend on x only through dot products with other x values. This will let us do a cool feature-space maneuver called the *kernel trick*. Stay tuned.

In the primal, there are D parameters. In the dual, there are n.

5.3 Support vector machines for non-separable data

Now, we have two issues to think about:

1. Although the SVM separator is very robust with respect to moving non-support vectors, it is highly sensitive to moving the support vectors or to "outliers" close to the margin.

2. The SVM optimization problem, as we have formulated it in the previous section, is infeasible if the data are not linearly separable.

It would be great if we could just "give up" on a few points and construct the classifier on the basis of the remaining points. This would be a hard problem to solve but we can do something close to it by introducing slack variables for the margin constraints in support vector machines.

For each training example, we will add *slack variable* ξ_i, which is allowed to "take up the slack" in the associated constraint; so that the constraints now have the form

$$y^{(i)}(w \cdot x^{(i)} + w_0) \geqslant 1 - \xi_i \quad .$$

If we picked the slack values in advance, we would have a problem of the same form as before. But, we will leave them as variables in the optimization, to be chosen in such a way as to minimize the objective (which now penalizes the sum of the slack variable values) subject to satisfying the constraints.

The primal *soft-margin* SVM optimization problem is given by

$$\begin{aligned} \text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{subject to} \\ y^{(i)}(w \cdot x^{(i)} + w_0) \geqslant 1 - \xi_i, \quad i = 1, \ldots, n \\ \xi_i \geqslant 0, \quad i = 1, \ldots, n \end{aligned}$$

Recall that, for the hard-margin SVM (without slack variables), the geometric margin is defined as the minimum distance from any point to the separating hyperplane. This distance can be shown to be $1/\|w\|$, where the 1 in the numerator comes from the classification constraints $y^{(i)}(w \cdot x^{(i)} + w_0) \geqslant 1$ in the optimization problem. When we use slack variables (regardless of whether the data is separable), some of the points may be misclassified, so we can no longer measure the margin directly in terms of the minimum distance to the separating hyperplane. However, we can still use the expression $1/\|w\|$ as the definition of the margin, where w comes from solving the SVM optimization problem with slack variables. Indeed, all the points that lie exactly on the margin will satisfy $y^{(i)}(w \cdot x^{(i)} + w_0) = 1$ as they would without slack variables.

You can think of

$$\sum_{i=1}^{n} \xi_i$$

as the total amount of distance from the margin to points that are on the wrong side of it. As we increase C, we will try harder to keep all of the points on the appropriate side of the margin, at the cost of having a smaller margin. As $C \to \infty$ it turns back into the hard-margin SVM.

- Data points for which $\xi_i = 0$ are correctly classified and are either on the margin or on the correct side of the margin.

- Data points for which $0 < \xi_i \leqslant 1$ are inside the margin but on the correct side of the decision boundary.
- Data points for which $\xi_i > 1$ are on the wrong side of the decision boundary and are therefore misclassified.

We can set this up as a Lagrangian optimization problem; we will have to introduce a new set of Lagrange multipliers μ_i, which we use to enforce the constraint that the ξ_i be non-negative.So, we have

What a horrible notation choice is μ! But I decided to follow Bishop here...

$$L(w, w_0, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_n - \sum_{i=1}^{n}\alpha_i\left(y^{(i)}x^{(i)}\cdot w - 1 + \xi_i\right) - \sum_{i=1}^{n}\mu_i\xi_i \quad .$$

The detailed derivation is in Bishop. First, we find that the weights are as before!

$$w = \sum_{i=1}^{n}\alpha_i y^{(i)}x^{(i)} \quad .$$

Then we end up, after taking derivatives with respect to w, w_0, and ξ, setting to 0, and using the results to write the Lagrangian only in terms of α, we end up with this optimization problem to find the αs:

$$\text{maximize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i-1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(x^{(i)}\cdot x^{(j)}\right) \quad \text{subject to}$$

$$0 \leqslant \alpha_i \leqslant C$$

$$\sum_{i=1}^{n}\alpha_i y^{(i)} = 0$$

This is a quadratic program, so it's not too hard to solve. There are strategies for solving really big ones by optimizing with respect to only a few multipliers at a time. SMO (sequential minimal optimization) scales somewhere between linearly and quadratically with the number of points.

Support vectors are points with $\alpha_i > 0$. It's important to note that the associated slack variables for these points can be in any of the cases above: they can be exactly on the margin, inside the margin, or on the wrong side of the separator. Each support vector satisfies, the following, because the constraint is tight against the slack variable:

$$y^{(i)}x^{(i)}\cdot w = 1 - \xi_i \quad .$$

In addition, as a consequence of gradient of L with respect to ξ_i being 0, we have

$$\alpha_i = C - \mu_i \quad .$$

If $\alpha_i < C$, then $\mu_i > 0$, so the point lies on the margin. If $\alpha_i = C$, then they might be correctly classified or misclassified, depending on the value of ξ_i.

To compute w_0, we average over $\mathcal{M}$ which is the set of indices for which $0 < \alpha_i < C$:

$$w_0 = \frac{1}{|\mathcal{M}|}\sum_{j\in m}\left(y^{(j)} - \sum_{i\in S}\alpha_i y^{(i)}\left(x^{(j)\mathsf{T}}x^{(i)}\right)\right) \quad .$$

5.4 Perceptron

	No model	Prediction rule	Prob model	Dist over models
Classification	❋			

Here's a very different method for finding a linear separator. We don't even have an objective function....just an algorithm. We will be looking for w such that $y(x; w) = x \cdot w$, and our prediction $h(x)$ will be $+1$ if $y(x; w) > 0$ and -1 otherwise.

Assume we've added a column of 1's to the data

- Start with w assigned to any value, but $\mathbf{0}$ is typical.
- Go through the training examples $x^{(i)}y^{(i)}$ in order, and go back to the beginning of the training set until you have a weight vector that categorizes the whole training set correctly.
- If $y^{(i)} \neq h(x^{(i)}; w)$

$$w \leftarrow w + y^{(i)}x^{(i)} \quad .$$

Parameter updates tend to correct mistakes. When we make a mistake on $x^{(i)}$,

- the sign of $w \cdot x^{(i)}$ disagrees with $y^{(i)}$
- the product of $y^{(i)} w \cdot x^{(i)}$ is negative
- the updated parameters are $w + y^{(i)}x^{(i)}$

If we try to classify with the new parameters, we will have

$$\begin{aligned} y(x; w') &= y^{(i)}(w + y^{(i)}x^{(i)}) \cdot x^{(i)} \\ &= y^{(i)}(w \cdot x^{(i)}) + {y^{(i)}}^2(x^{(i)} \cdot x^{(i)}) \\ &= y^{(i)}(w \cdot x^{(i)}) + |x^{(i)} \cdot x^{(i)}|^2 \end{aligned}$$

So the value increases as a result of the update. Eventually (if this update is repeated) the point will be classified correctly.

Theorem: If the data are separable, the algorithm is guaranteed to terminate with w representing a separator. It will not terminate if the data are not separable.

Note also that the weights are a weighted combination of input vectors...sound familiar?

Theorem: If

- There exists R such that, for all i, $\|x^{(i)}\| < R$,
- There exists w^* with geometric margin γ_g

This is the margin of the closest point to the separator.

Then the perceptron algorithm makes at most

$$\frac{R}{\gamma_g}$$

errors.

Note that this result is independent of the number of data points n, and the dimension of the space D. A large margin, relative to R, implies that we will make few mistakes.

Can prove this by showing that the angle between w^* and w is decreasing by a finite amount after every mistake.