Linear Methods for Classification

-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

An Overview of Classification

Some examples:

- A person arrives at an emergency room with a set of symptoms that could be 1 of 3 possible conditions. Which one is it?
- A online banking service must be able to determine whether each transaction is fraudulent or not, using a customer's location, past transaction history, etc.
- Given a set of individuals sequenced DNA, can we determine whether various mutations are associated with different phenotypes?

All of these problems are not regression problems. They are classification problems.

The Set-up

It begins just like regression: suppose we have observations

$$\mathcal{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

Again, we want to estimate a function that maps X into Y that helps us predict as yet observed data.

(This function is known as a classifier)

The same constraints apply:

- We want a classifier that predicts test data, not just the training data.
- Often, this comes with the introduction of some bias to get lower variance and better predictions.

How do we measure quality?

In regression, we have $Y_i \in \mathbb{R}$ and use squared error loss

Instead, let $Y \in \mathcal{G} = \{1, \ldots, G\}$

(This is arbitrary, sometimes other numbers, such as $\{-1, 1\}$ will be used)

We again make predictions $\hat{Y}$ based on $\mathcal{D}$

Our loss function is now a $G \times G$ matrix L with

- zeros on the diagonals
- $\ell(g, g')$ on the off diagonal ($g \neq g'$)

How do we measure quality?

Again, we appeal to risk

$$R(\hat{g}) = \mathbb{E}_Z \ell_{\hat{g}}(Z)$$

If we use the law of total probability, this can be written

$$R(\hat{g}) = \mathbb{E}_X \sum_{y=1}^{G} \ell_{\hat{g}}(Z = (y, X)) \mathbb{P}(Y = y|X)$$

This can be minimized point wise over x, to produce

$$g^*(x) = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \sum_{y=1}^{G} \ell_g(z = (y, x)) \mathbb{P}(Y = y|X = x)$$

(This is the Bayes' classifier. Also, $R(g^*)$ is the Bayes' limit)

Best classifier

If we make specific choices for ℓ, we can find g^* exactly

As Y takes only a few values, zero-one prediction risk is natural

$$\ell_g(Z) = \mathbf{1}_{Y \neq g(X)}(Z) \Rightarrow R(g) = \mathbb{E}[\ell_g(Z)] = \mathbb{P}(g(X) \neq Y),$$

(This means we want to label or classify a new observation (X, Y) such that $\hat{f}(X) = Y$ as often as possible)

Under this loss, we have

$$g^*(x) = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \left[1 - \mathbb{P}(Y = g|X = x)\right] = \underset{g \in \mathcal{G}}{\arg\max}\, \mathbb{P}(Y = g|X = x)$$

Best classifier

Suppose we encode a two-class response as $Y \in \{0, 1\}$

Let's continue to use squared error loss: $\ell_f(Z) = (Y - f(X))^2$

Then, the Bayes' rule is

$$m(X) = \mathbb{E}[Y|X] = \mathbb{P}(Y = 1|X)$$

Hence, we achieve the same Bayes' rule/limit with squared error classification by discretizing the probability:

$$g^*(x) = \mathbf{1}_{m(x)>1/2}(x)$$

Classification is easier than regression

Let $\hat{m}$ be any estimate of m

Let $\hat{g}(x) = \mathbf{1}_{\hat{m}(x)>1/2}(x)$

It can be shown that

$$\begin{aligned}
&\mathbb{P}(Y \neq \hat{g}(X)|X = x) - \mathbb{P}(Y \neq g^*(X)|X = x) = \\
&= (2m(x) - 1)(\mathbf{1}_{g^*(x)=1}(x) - \mathbf{1}_{\hat{g}(x)=1}(x)) \\
&= |2m(x) - 1|\mathbf{1}_{g^*(x)\neq\hat{g}(x)}(x) \\
&= 2|m(x) - 1/2|\mathbf{1}_{g^*(x)\neq\hat{g}(x)}(x)
\end{aligned}$$

[Exercise]

Classification is easier than regression

Now

$$g^*(x) \neq \hat{g}(x) \Rightarrow |\hat{m}(x) - m(x)| \geq |\hat{m}(x) - 1/2|$$

Therefore

$$\begin{aligned}
&\mathbb{P}(Y \neq \hat{g}(X)) - \mathbb{P}(Y \neq g^*(X)) = \\
&= \int (\mathbb{P}(Y \neq \hat{g}(X)|X = x) - \mathbb{P}(Y \neq g^*(X)|X = x)) d\mathbb{P}_X(x) \\
&= \int 2|\hat{m}(x) - 1/2| \mathbf{1}_{g^*(x) \neq \hat{g}(x)}(x) d\mathbb{P}_X(x) \\
&\leq 2 \int |\hat{m}(x) - m(x)| \mathbf{1}_{g^*(x) \neq \hat{g}(x)}(x) d\mathbb{P}_X(x) \\
&\leq 2 \int |\hat{m}(x) - m(x)| d\mathbb{P}_X(x)
\end{aligned}$$

(If $\hat{m}$ gets close to m on average, we do good classifications. The converse is not true)

Bayes' rule and class densities

Using Bayes' theorem

$$\begin{aligned} m(x) &= \mathbb{P}(Y = 1 | X = x) \\ &= \frac{p(x|Y = 1)\mathbb{P}(Y = 1)}{\sum_{y \in \{0,1\}} p(x|Y = y)\mathbb{P}(Y = y)} \\ &= \frac{f_1(x)\pi}{f_1(x)\pi + f_0(x)(1 - \pi)} \end{aligned}$$

We call $f_g(x)$ the class densities

The Bayes' rule can be rewritten[♯]

$$g^*(x) = \begin{cases} 1 & \text{if } \frac{f_1(x)}{f_0(x)} > \frac{1-\pi}{\pi} \\ 0 & \text{otherwise} \end{cases}$$

How to find a classifier

All of these prior expressions for g^* give rise to classifiers

- Empirical risk minimization: Choose a set of classifiers Γ and find $\hat{g} \in \Gamma$ that minimizes some estimate of $R(g)$

  (This can be quite challenging as, unlike in regression, the training error is nonconvex)

- Regression: Find an estimate $\hat{m}$ and plug it in to the Bayes' rule
- Density estimation: Estimate f_g from the appropriate Z and $\hat{\pi} = \overline{Y}$ and plug them in to g^*

Linear classifiers

Linear classifier

As our classifier $\hat{g}$ takes a discrete number of values, it is equivalent to partitioning the covariate space into regions

The boundaries between these regions are known as decision boundaries

These decision boundaries are sets of points at which $\hat{g}$ is indifferent between two classes

A linear classifier is a $\hat{g}$ that produces linear decision boundaries

Linear classifier: Example

Suppose $\mathcal{G} = \{0, 1\}$ and we form the GLM logistic regression

The posterior probabilities are

$$\mathbb{P}(Y = 1|X = x) = \frac{\exp\{\beta_0 + \beta^\top x\}}{1 + \exp\{\beta_0 + \beta^\top x\}}$$
$$\mathbb{P}(Y = 0|X = x) = \frac{1}{1 + \exp\{\beta_0 + \beta^\top x\}}$$

The logit (i.e.: log odds) transformation forms a linear decision boundary

$$\log\left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)}\right) = \beta_0 + \beta^\top x$$

The decision boundary is the hyperplane $\{x : \beta_0 + \beta^\top x = 0\}$

(Log-odds below 0, classify as 0, above 0 classify as a 1)

Linear classifier: Extensions

The term "linear classifier" can be used to describe a classifier that has linear decision boundaries in a higher dimensional space, but which as a nonlinear decision boundary in the original covariate space

For instance, if I include as features:

$$x_1^2, \ldots, x_p^2, x_1 x_2, \ldots, x_1 x_p, \ldots$$

and thereby add $p(p+1)/2$ additional features, a linear classifier in this enhanced space will be nonlinear (and in fact quadratic) in the original covariates

This is a parametric kernel method

Bayes' rule-ian approach

The decision theory for classification indicates we need to know the posterior probabilities: $\mathbb{P}(Y = g|X)$ for doing optimal classification

Suppose that

- $p_g(x) = \mathbb{P}(x|Y = g)$ is the likelihood of the covariates given the class labels
- $\pi_g = \mathbb{P}(Y = g)$ is the prior

Then

$$\mathbb{P}(Y = g|X = x) = \frac{p_g(x)\pi_g}{\sum_{g \in \mathcal{G}} p_g(x)\pi_g} \propto p_g(x)\pi_g$$

Conclusion: Having the class densities almost gives us the Bayes' rule as the training proportions can usually be used to estimate π_g

(Though, sometimes estimating π_g can be nontrivial/impossible)

Bayes' rule-ian approach: Summary

There are many techniques based on this idea

- Linear/quadratic discriminant analysis
  (Estimates $p_g(x)$ assuming multivariate Gaussianity)
- General nonparametric density estimators
- Naive Bayes (Factors $p_g(x)$ assuming conditional independence)

Discriminant analysis

Suppose that

$$p_g(x) \propto |\Sigma_g|^{-1/2} e^{-(x-\mu_g)^\top \Sigma_g^{-1}(x-\mu_g)/2}$$

Let's assume that $\Sigma_g \equiv \Sigma$.

Then the log-odds between two classes g, g' is:

$$\begin{aligned}\log \frac{\mathbb{P}(Y=g|X=x)}{\mathbb{P}(Y=g'|X=x)} &= \log \frac{p_g(x)}{p_{g'}(x)} + \log \frac{\pi_g}{\pi_{g'}} \\ &= \log \frac{\pi_g}{\pi_{g'}} - (\mu_g + \mu_{g'})^\top \Sigma^{-1}(\mu_g - \mu_{g'})/2 \\ &\quad + x^\top \Sigma^{-1}(\mu_g - \mu_{g'})\end{aligned}$$

This is linear in x, and hence has a linear decision boundary

Types of discriminant analysis

The linear discriminant function is (proportional to) the log posterior:

$$\delta_g(x) = \log \pi_g + x^\top \Sigma^{-1} \mu_g - \mu_g^\top \Sigma^{-1} \mu_g / 2$$

and we assign $g(x) = \text{argmin}_g \, \delta_g(x)$

(This is just minimum Euclidean distance, weighted by the covariance matrix and prior probabilities)

Now, we must estimate μ_g and Σ. If we...

- use the intuitive estimators $\hat{\mu}_g = \overline{X}_g$ and $\frac{1}{n-G} \sum_{g \in \mathcal{G}} \sum_{i \in g} (X_i - \hat{\mu}_g)(X_i - \hat{\mu}_g)^\top$ then we have produced linear discriminant analysis (LDA)
- regularize these 'plug-in' estimates, we can form regularized discriminant analysis (Friedman (1989)). This could be (for $\lambda \in [0, 1]$):

$$\hat{\Sigma}_\lambda = \lambda \hat{\Sigma} + (1 - \lambda) \hat{\sigma}^2 I$$

Quadratic discriminant analysis (QDA)

If we drop the assumption regarding identical covariances, we get the following discriminant function:

$$\delta_g(x) = \log \pi_g + x^\top \Sigma_g^{-1} \mu_g - \mu_g^\top \Sigma_g^{-1} \mu_g / 2 - \log |\Sigma_g| / 2$$

where Σ_g can be estimated by the sample covariance of the observations in group g

In my experience, QDA works well if n is large relative to p

(However, it isn't often computable in practice; too many parameters)

We can augment regularized discriminant analysis to shrink each $\hat{\Sigma}_g$ to $\hat{\Sigma}$ or even to $\hat{\Sigma}_\lambda$

$$\hat{\Sigma}_{g,(\gamma,\lambda)} = \gamma \hat{\Sigma}_g + (1-\gamma)\hat{\Sigma}_\lambda$$

(To the best of my knowledge, little is formally known about this procedure. See Guo et al. (2006) for an empirical comparison)

Reduced rank LDA

Part of the popularity of LDA is that it provides dimension reduction as well

The G class centroids μ_g must all lie in an affine subspace of dimension $G - 1$ (presuming $G < p$)

(Let $\mathcal{H}_{G-1}$ be this subspace)

If G is much less than p, this will be a substantial drop in dimension

Reduced rank LDA

In practice, we can compute LDA from spectral information:

$$\begin{aligned}\delta_g(x) &= \log \pi_g + x^\top \Sigma^{-1} \mu_g - \mu_g^\top \Sigma^{-1} \mu_g / 2 \\ &\propto \log \pi_g + (x - \mu_g)^\top \Sigma^{-1} (x - \mu_g)/2\end{aligned}$$

So,

1. Spectrum: Form $\hat{\Sigma}_\lambda = UDU^\top$
2. Sphere: Rewrite your data as $\tilde{X} \leftarrow D^{-1/2} U^\top X$
3. Assign: Classify to the closest mean in transformed space

   (Penalizing by estimate of prior probability)

Reduced rank LDA

We can ignore any information orthogonal to $\mathcal{H}_{G-1}$, as it contributes to each class equally (in the sphered space)

So, project $\tilde{X}$ onto $\mathcal{H}_{G-1}$ and make distance computations there

When $G = 2, 3$, this means we can plot the projection onto $\mathcal{H}_{G-1}$ with no loss of information about the LDA solution

If $G > 3$, then we may wish to project onto a reduced space $\mathcal{H}_L \subset \mathcal{H}_{G-1}$

We'd like $\mathcal{H}_L$ to maintain the most amount of information possible for assigning to classes

Reduced rank LDA

This can be done via the following procedure

1. Centroids: Compute $G \times p$ matrix M of class centroids
2. Covariance: Form $\hat{\Sigma}$ as the common covariance matrix
3. Sphere: $\tilde{M} = M\hat{\Sigma}^{-1/2}$
4. Between Covariance: Find covariance matrix for $\tilde{M}$, call it B
5. Spectrum Compute $B = VSV^{\top}$

Now, $\text{span}(V_L) = \mathcal{H}_L$

Also, the coordinates of the data in this space are $Z_k = v_k^{\top}\hat{\Sigma}^{-1/2}X$

These derived variables are commonly called canonical coordinates

Reduced rank LDA: Summary

- Gaussian likelihoods with identical covariances leads to linear decision boundaries (LDA)
- We can actually do all relevant computations/graphics on the reduced space $\mathcal{H}_{G-1}$
- If this isn't small enough, we can do 'optimal' dimension reduction to $\mathcal{H}_L$

As an aside, this procedure is identical to Fisher's discriminant analysis

Logistic regression

Logistic regression for two classes simplifies to a likelihood:

(Using $\pi_i(\beta) = \mathbb{P}(Y = 1 | X = X_i, \beta)$)

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{n} \left(y_i \log(\pi_i(\beta)) + (1 - y_i) \log(1 - \pi_i(\beta)) \right) \\
&= \sum_{i=1}^{n} \left(y_i \log(e^{\beta^\top X_i} / (1 + e^{\beta^\top X_i})) - (1 - y_i) \log(1 + e^{\beta^\top X_i}) \right) \\
&= \sum_{i=1}^{n} \left(y_i \beta^\top X_i - \log(1 + e^{\beta^\top X_i}) \right)
\end{aligned}$$

This gets optimized via Newton-Raphson updates and iteratively reweighed least squares

Sparse logistic regression

This procedure suffers from all the same problems as least squares

We can use penalized likelihood techniques in the same way as we did before

This means maximizing (over β_0, β):

$$\sum_{i=1}^{n} \left(y_i(\beta_0 + \beta^\top X_i) - \log(1 + e^{\beta_0 + \beta^\top X_i}) \right) - \lambda(\alpha||\beta||_1 + (1-\alpha)||\beta||_2^2)$$

(Don't penalize the intercept and do standardize the covariates)

This is the logistic elastic net

Sparse logistic regression: Software

Using the `R` package `glmnet` finds the minimum CV solution over a grid of λ values

Unfortunately, the computations are more difficult for path algorithms (such as the `lars` package) due to the coefficient profiles being only piecewise smooth

`glmpath` is an `R` package that does quadratic approximations to the profiles, while still computing the exact points at which the active set changes

(Park, Hastie (2007). It is necessary to set a 'step' size argument for the approximation.)

Logistic versus LDA

The log posterior odds via the Gaussian likelihood (LDA) for class g versus G are

$$\begin{aligned}\log \frac{\mathbb{P}(Y=g|X=x)}{\mathbb{P}(Y=G|X=x)} &= \log \frac{\pi_g}{\pi_G} - (\mu_g+\mu_G)^\top \Sigma^{-1}(\mu_g-\mu_G)/2 \\ &\quad + x^\top \Sigma^{-1}(\mu_g-\mu_G) \\ &= \alpha_{g,0} + \alpha_g^\top x\end{aligned}$$

Likewise, multi class logistic follows (for $g = 1, \ldots, G-1$):

$$\log \frac{\mathbb{P}(Y=g|X=x)}{\mathbb{P}(Y=G|X=x)} = \beta_{g,0} + \beta_g^\top x$$

(The choice of base class G is arbitrary)

They both specify the log-odds as linear models!

Logistic versus LDA

We can write the joint distribution of Y and X as

$$\mathbb{P}(X, Y) = \mathbb{P}(Y|X)\mathbb{P}(X)$$

The previous slide shows that $\mathbb{P}(Y|X)$ is the same for both methods:

$$\mathbb{P}(Y = g|X = x) = \frac{e^{\alpha_{g,0}+\alpha_g^\top x}}{1 + \sum_{k=1}^{G-1} e^{\alpha_{k,0}+\alpha_k^\top x}}$$

- Logistic regression leaves $\mathbb{P}(X)$ arbitrary, and implicitly estimates it with the empirical measure

  (This could be interpreted as a frequentist approach, where we are maximizing the likelihood only and using the improper uniform prior)
- LDA models

$$\mathbb{P}(X, Y = g) = \mathbb{P}(X|Y = g)\mathbb{P}(Y = g) = N(x; \mu_g, \Sigma)\pi_g$$

Logistic versus LDA

Some remarks:

- Forming logistic requires fewer assumptions
- The MLEs under logistic will be undefined if the classes are perfectly separable
- If some entries in X are qualitative, then the modeling assumptions behind LDA are suspect
- In practice, the two methods tend to give very similar results

Support vector machines

Optimal separating hyperplanes

A main initiative in early computer science was to find separating hyperplanes among groups of data

(Rosenblatt (1958) with the perceptron algorithm)

The issue is that if there is a separating hyperplane, there is an infinite number

An optimal separating hyperplane can be generated by finding support points and bisecting them.

(Vapnik (1996))

Basic linear geometry

A hyperplane in $\mathbb{R}^p$ is given by

$$\mathcal{H} = \{x \in \mathbb{R}^p : f(x) = \beta_0 + \beta^\top x = 0\}$$

1. The vector β is normal to $\mathcal{H}$
2. For any point $x \in \mathbb{R}^p$, the (signed) length of its orthogonal complement to $\mathcal{H}$ is $f(x)$

Support vector machines (SVM)

Let $Y_i \in \{-1, 1\}$

(w.l.o.g let $||\beta||_2 = 1$)

A classification rule induced by this hyperplane is

$$\hat{Y}(x) = \text{sgn}(x^\top \beta + \beta_0)$$

Separating hyperplanes

As our classification rule is based on a hyperplane $\mathcal{H}$

$$\hat{Y}(x) = \text{sgn}(x^\top \beta + \beta_0)$$

we know the signed distance to $\mathcal{H}$ is $f(x) = x^\top \beta + \beta_0$

Under classical separability, we can find a function such that $Y_i f(X_i) > 0$

(That is, makes perfect classifications via $\hat{Y}$)

The larger the quantity $Y_i f(X_i)$, the more separated the classes

Optimal separating hyperplane

This idea can be encoded in the following convex program

$$\max_{\beta_0,\beta} M \text{ subject to}$$

$$Y_i f(X_i) \geq M \text{ for each } i$$

Drop the norm constraint on β and divide both sides. Then we have the equivalent program

$$\min_{\beta_0,\beta} ||\beta||_2 \text{ subject to}$$

$$Y_i f(X_i) \geq 1 \text{ for each } i$$

(Convex optimization program: quadratic criterion, linear inequality constraints)

Optimal separating hyperplane

Of course, we can't realistically assume that the data are linearly separated (even in a transformed space)

In this case, the previous program has no feasible solution

We need to introduce slack variables that allow for overlap among the classes

SVMs

$$\min_{\beta_0,\beta} ||\beta||_2 \text{ subject to}$$

$$Y_i f(X_i) \geq 1 - \xi_i, \xi_i \geq 0, \sum \xi_i \leq c, \text{ for each } i$$

(Convex optimization program: quadratic criterion, linear inequality constraints)

This can be rewritten as

$$\min_{\beta_0,\beta} ||\beta||_2 /2 + C \sum \xi_i \text{ subject to}$$

$$Y_i f(X_i) \geq 1 - \xi_i, \xi_i \geq 0, , \text{ for each } i$$

Note that

- C is the cost parameter
- The separable case corresponds to $C = \infty$

SVMs

The corresponding Lagrange function to the constrained optimization problem is

$$\ell_{SVM}(\beta, \beta_0, \xi) = ||\beta||_2 /2 + C \sum \xi_i - \\ - \sum_{i=1}^{n} \gamma_i [Y_i f(X_i) - (1 - \xi_i)] - \sum_{i=1}^{n} \lambda_i \xi_i$$

Minimize with respect to β_0, β, ξ_i via partial derivatives:

$$\beta = \sum_{i=1}^{n} \gamma_i y_i x_i$$

$$0 = \sum_{i=1}^{n} \gamma_i y_i$$

$$\gamma_i = C - \lambda_i$$

$$\gamma_i, \lambda_i, \xi_i \geq 0$$

SVMs

Outline of optimization steps

1. Formulate constrained form of problem
2. Convert to (primal) Langragian form
3. Take all relevant (sub)-derivatives and set to zero
4. Substitute these conditions back into primal Langrangian $\longrightarrow$ dual Lagrangian

   (This forms a lower bound on the solution to the constrained primal form of objective)
5. Form Karush-Kuhn-Tucker (KKT) conditions for inequality constraints
6. Examine the conditions for strong duality which implies that the primal and dual forms have the same solution

   (The usual method is via Slater's condition, which says strong duality holds if it is a convex program with non-empty constraint region)

SVMs

Once minimizers for $\hat{\gamma}$ are found, we can plug-in and get

$$\hat{\beta} = \sum_{i=1}^{n} \hat{\gamma}_i Y_i X_i$$

Due to the KKT conditions[♯]:

$$\gamma_i[Y_i(X_i^\top \beta + \beta_0) - (1 - \xi_i)] = 0 \quad (1)$$

$$\lambda_i \xi_i = 0 \quad (2)$$

$$Y_i(X_i^\top \beta + \beta_0) - (1 - \xi_i) = 0 \quad (3)$$

The $\hat{\gamma}_i$ are nonzero only for i such that (3) holds (by (1))

These observations are called the support vectors

SVMs

By the previous conditions, either $\hat{\xi}_i = 0$ or $\hat{\gamma}_i = C$

Using the condition: $\gamma_i[Y_i(X_i^\top \beta + \beta_0) - (1 - \xi_i)] = 0$

For support vector i, if $\hat{\xi}_i = 0$:

$$Y_i(X_i^\top \beta + \beta_0) - (1 - \xi_i) = 0 \Leftrightarrow \hat{\beta}_0 = 1/Y_i - X_i^\top \hat{\beta}$$

(These estimates are usually averaged to get a final estimate of β_0)

Now, the final classification is given by

$$\hat{Y}(x) = \text{sgn}(\hat{f}(x)) = \text{sgn}(x^\top \hat{\beta} + \hat{\beta}_0)$$

The tuning parameter is given by the cost C

Kernel SVM

If we return to step 4. from the SVM outline:

"Substitute these conditions back into primal Langrangian $\longrightarrow$ dual Lagrangian"

We get that this dual Lagrangian is:

$$\ell_D(\gamma) = \sum_i \gamma_i - \frac{1}{2} \sum_i \sum_{i'} \gamma_i \gamma_{i'} Y_i Y_{i'} X_i^\top X_{i'}$$

with side conditions: $\gamma_i \in [0, C]$ and $\gamma^\top Y = 0$

The term $X_i^\top X_{i'} = \langle X_i, X_{i'} \rangle$ is an inner product

SVMs therefore depend on the covariates via an inner product only

This leaves them ripe for a kernel method

Kernel Methods

Three related methods

The following are seemingly disparate methods

- Smoothness penalization: Regularizing a loss functions with a penalty on smoothness

  (Example: Kernel SVM)
- Feature creation: Imposing a feature mapping $\Phi : \mathbb{R}^p \to \mathcal{A}$ thus creating new features e.g. via polynomials or interactions

  (Example: Regression splines or polynomial regression)
- Gaussian processes: Modeling the regression function as a Gaussian process with a given mean and covariance

  (Example: Gaussian process regression)

It turns out these concepts are all the same and each forms a reproducing kernel Hilbert space (RKHS)

(Many of these ideas are in Wahba (1990). It was introduced to the ML community in Vapnik et al. (1996) and summarized in a nice review paper in Hofmann et al. (2008))

Kernel methods

Suppose $k : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ is a positive definite kernel

(This means $\int \int k(x, y) f(x) f(y) dx dy > 0$)

To be concrete, think of $\mathcal{A} = \mathbb{R}^p$

(However, any set of objects will do as long as an inner product can be defined)

Let's consider the space of functions generated by the completion of

$$\mathcal{H}_k = \{k(\cdot, y) : y \in \mathbb{R}^p\}$$

(This loosely speaking all functions of the form $f(x) = \sum_{j=1}^{J} \alpha_j k(x, y_j)$)

Kernel methods

Write the eigenvalue expansion of k as

$$k(x, y) = \sum_{j=1}^{\infty} \theta_j \phi_j(x) \phi_j(y)$$

with

- $\theta_j \geq 0$ (nonnegative definite)
- $\left|\left|(\theta_j)_{j=1}^{\infty}\right|\right|_2 < \infty$

(This is called Mercer's theorem, and such a k is called a Mercer kernel)

We can write any $f \in \mathcal{H}_k$ with two constraints

- $f(x) = \sum_{j=1}^{\infty} f_j \phi_j(x)$
- $\langle f, f \rangle_{\mathcal{H}_k} = ||f||_{\mathcal{H}_k}^2 = \sum_{j=1}^{\infty} f_j^2 / \theta_j < \infty$

Kernel methods via regularization

After specifying a kernel function[1] k, we can define an estimator via

$$\min_{f \in \mathcal{H}_k} \hat{\mathbb{P}} \ell_f + \lambda ||f||^2_{\mathcal{H}_k}$$

This is a (potentially) infinite dimensional optimization problem

(hard, especially with a computer)

It can be shown that the solution has the form

$$\hat{f}(x) = \sum_{i=1}^{n} \beta_i k(x, x_i)$$

(This is known as the representer theorem)

[1] Or crucially and equivalently a set of eigenfunctions and eigenvalues

Kernel methods via regularization

$$\hat{f}(x) = \sum_{i=1}^{n} \beta_i k(x, x_i)$$

The terms $k(x, x_i)$ are the representers, as

$$\langle k(\cdot, x_i), f \rangle_{\mathcal{H}_k} = f(x_i)$$

and $\mathcal{H}_k$ is called a reproducing kernel Hilbert space (RKHS) as

$$\langle k(\cdot, x_i), k(\cdot, x_{i'}) \rangle_{\mathcal{H}_k} = k(x_i, x_{i'})$$

Kernel methods via regularization

Due to these properties, we can write the optimization problem as

$$\min_{\beta} \hat{\mathbb{P}} \ell_{\mathbf{K}\beta} + \lambda \beta^\top \mathbf{K} \beta$$

where $\mathbf{K} = [k(x_i, x_{i'})]$

This provides a prescription for forming an incredibly rich suite of estimators:

Choose a

- kernel k
- loss function ℓ

and then minimize

Kernel methods via regularization: Example

Suppose that $\ell_{\mathbf{K}\beta}(Z) = (Y - \mathbf{K}\beta)^2$

Then:

$$\hat{\beta} = \underset{\beta}{\text{argmin}}\, \hat{\mathbb{P}}\ell_{\mathbf{K}\beta} + \lambda\beta^\top \mathbf{K}\beta = (\mathbf{K} + \lambda I)^{-1} Y$$

and

$$\hat{f} = \mathbf{K}\hat{\beta} = \mathbf{K}(\mathbf{K} + \lambda I)^{-1} Y = (\lambda \mathbf{K}^{-1} + I)^{-1} Y$$

are the fitted values

(This should be compared with the notes on ridge regression)

Kernel methods via regularization: Example

Alternatively, statisticians have been including polynomial terms/interactions for ages

Form

$$k_d(x, y) = (x^\top y + 1)^d$$

k_d has $M = \binom{p+d}{d}$ eigenfunctions

These span the space of polynomials in $\mathbb{R}^p$ with degree d

Kernel methods via regularization: Example

Example: Let $d = p = 2 \Rightarrow M = 6$ and

$$\begin{aligned} k(x, y) &= 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= \sum_{u=1}^{M} \Phi_u(x)\Phi_u(y) \\ &= \langle \Phi(x), \Phi(y) \rangle \end{aligned}$$

where[♯]

$$\Phi(y)^\top = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

(See Vapnik (1996) for more on this example)

Kernel methods: Summary

From this example, we see that we could have generated this same RKHS via:

- Specifying the eigenfunctions (or another set of functions with the same span) and projecting
- Defining the kernel k explicitly and minimizing
- Forming the feature map Φ directly and implicitly defining $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$

This last technique corresponds to kernelization, where inner products in the original covariate space are replaced with inner products of features

Kernel SVMs

Kernel SVM: A reminder

The dual Lagrangian is:

$$\ell_D(\gamma) = \sum_i \gamma_i - \frac{1}{2} \sum_i \sum_{i'} \gamma_i \gamma_{i'} Y_i Y_{i'} X_i^\top X_{i'}$$

with side conditions: $\gamma_i \in [0, C]$ and $\gamma^\top Y = 0$

Let's replace the term $X_i^\top X_{i'} = \langle X_i, X_{i'} \rangle$ with $\langle \Phi(X_i), \Phi(X_{i'}) \rangle$

Kernel SVMs

Typically, specifying Φ is unnecessary,

We need only define a kernel that is symmetric, positive definite

Some common choices for SVMs:

- Polynomial: $k(x, y) = (1 + x^\top y)^d$
- Radial basis: $k(x, y) = e^{-\lambda ||x-y||_\tau^\tau}$

Kernel SVMs

Reminder: the solution form for SVM is

$$\beta = \sum_{i=1}^{n} \gamma_i Y_i X_i$$

Kernelized, this is

$$\beta = \sum_{i=1}^{n} \gamma_i Y_i \Phi(X_i)$$

Therefore, the induced hyperplane is:

$$f(x) = \Phi(x)^\top \beta + \beta_0 = \sum_{i=1}^{n} \gamma_i Y_i \langle \Phi(x), \Phi(X_i) \rangle + \beta_0$$

The final classification is still

$$\hat{Y}(x) = \operatorname{sgn}(f(x))$$

SVMs via penalization

It is important to note that SVMs can be derived from penalized loss methods

Writing $f(x) = \Phi(x)^\top \beta + \beta_0$, consider

$$\min_{\beta,\beta_0} \sum_{i=1}^{n} [1 - Y_i f(X_i)]_+ + \lambda \left|\left|\beta\right|\right|_2^2 /2$$

This optimization problem produces the same solution as using $C = 1/\lambda$

Surrogate losses

It is tempting to minimize (analogous to linear regression)

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{Y_i \neq \hat{Y}(x)}(X_i) + \lambda\rho(\beta)$$

for some penalty term ρ

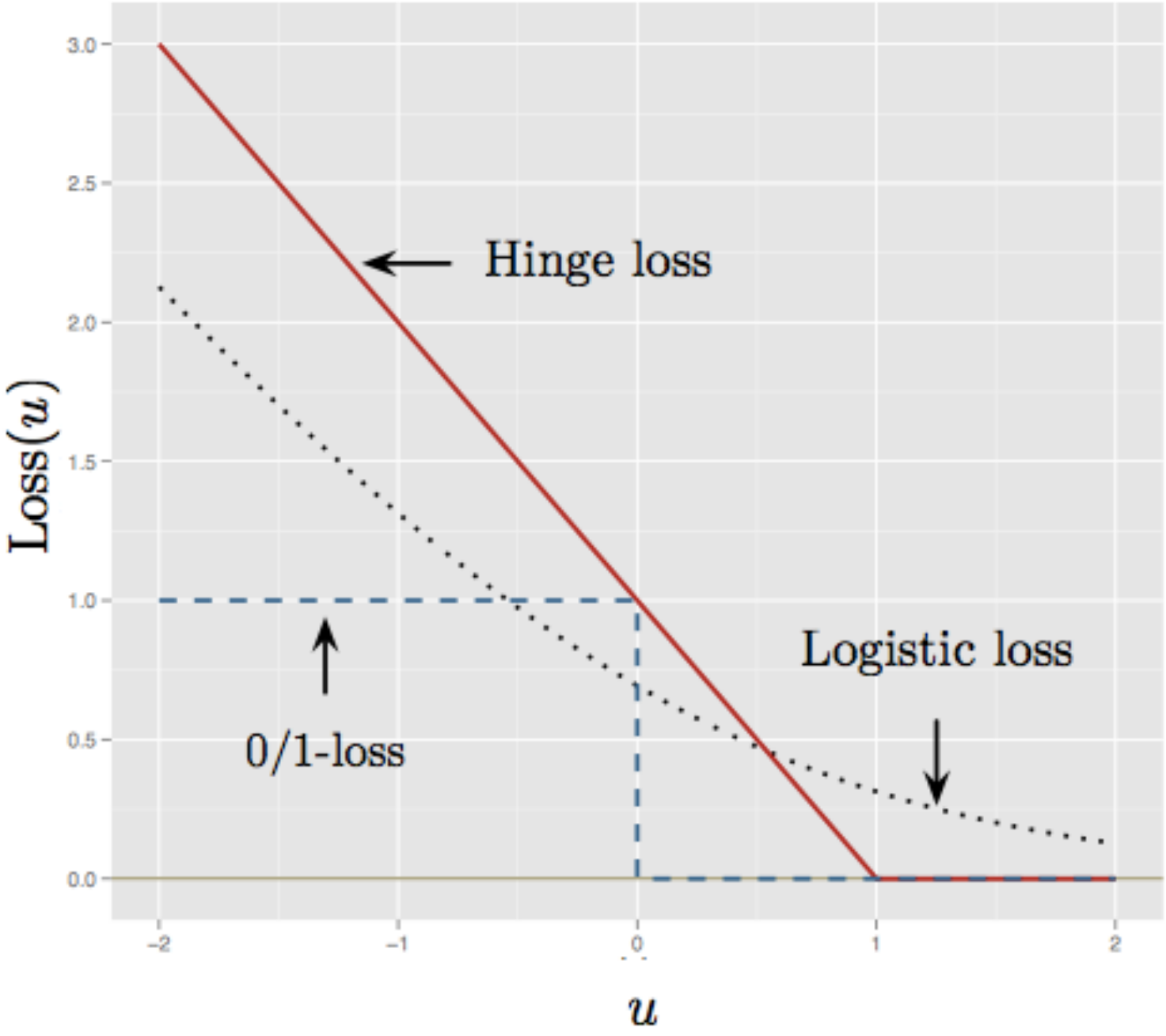
However, this is nonconvex

Idea: We can use a surrogate loss that mimics this function while still being convex

It turns out we have already done that!

- Hinge: $[1 - Yf(X)]_+$
- Logistic: $\log(1 + e^{-Yf(X)})$

Surrogate losses



SVMs in practice

General functions: The basic SVM functions are in the C++ library libsvm

R package: The R package e1071 calls libsvm

Path algorithm: `svmpath`

(Hastie et al (2004))

For a discussion and comparison see Karatzoglou, Meyer (2006).