1 Classification

Logistic

Is a glm. A glm has :

1. A distribution modeling Y (which is a probability π_i for logistic).
2. A linear predictor, $\eta = X\beta$
3. A link function $g(\mu) = X\beta = E(Y|X)$

We use certain functions to map things from $[-\infty, \infty]$ to $[0, 1]$ like the logit link, for example. See review notes from hierarchical. Solved with maximum likelihood approaches.

$$\log(\frac{\pi_i}{1-\pi_i}) = X_i^T\beta$$

1. β_0 : log-odds of the event happening with all other predictors set to 0
2. Continuous β_k : The change in the log-odds of response, comparing two populations whose value of X differs by 1 unit.
3. Categorical β_k : Each β_k represents the change in the log-odds of the event happening when the predictor is in category k compared to the reference level.
4. $\exp\{\beta_0\}$: odds of response with all other predictors set to 0
5. Continuous $\exp\{\beta_k\}$: the ratio of the odds of response when X = 1 to that when X = 0

LDA

Supervised algo (need labels)

Assumptions:

1. Each class is MVN
2. Every class has a common covariance matrix Σ

QDA

Assumptions:

Supervised algo

Assumptions:

1. Each class is MVN
2. Each class has unique covariance matrix Σ

Distinct covariance prevents some cancellations and makes the rule quadratic in x.

SVM

Have to use these in contrast to decision boundaries that are possible with separable data. We still want to maximize the margin, but we allow some points to be on the wrong side of the line.

hard margin vs. soft margin classifier

Naive Bayes

A classification method. The naivety is assuming that for a joint p-dimensional density function, we assume that its components are independent, which makes estimation much easier. This eliminates our need to understand how the p-covariates relate to each other. We don't actually think this is true, but is probably "good enough". This introduces bias, but reduces variance.

Bayes Decision Boundary

Test error is minimized, on average, by a classifier that *assigns each observation to the most likely class given its predictor values.* Assign to class j with predictor vector x_0 for which: $P(Y = j|X = X_0)$ is the largest. But this is usually not actually computable! The error rate is then $1 - max_j P(Y = j|X = x_0)$. If we have some class overlap, this will be greater than 0 (i.e. points intermingled with the other class).

But we usually don't know the distribution of Y given X!

---

2 Regression

Linear Regression

This is linear in the predictors, X. If you had like, β^2, that's just another scalar so it doesn't really matter. A squared term would still be a linear combination of the predictors, but the function wouldn't be linear.

Solving:

$$\text{argmin}_\beta ||y - X\beta||_2^2$$

y is projected onto the hyperplane spanned by X. Solve with calculus or QR factorization.

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Assumptions:

Examine residual plots to check for these

1. normality of residuals
2. homoskedasticity
3. linearity between predictors and target
4. independence of observations

Dummy encoding: creates n-1 new columns where n is the number of categories. If we made n columns, would have multicollinearity.

Ridge

$$\text{argmin}_\beta ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_2^2$$

Larger λ is more shrinkage.
Solving: closed-form solution.
Considerations: scale inputs before solving

LASSO

$$\text{argmin}_\beta ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1^2$$

Sets some coefficients to actual 0.

---

3 Other Methods

Bootstrap

Resampling technique. With n observations, sample n WITH REPLACEMENT and calculate the statistics of interest (mean, SD, etc.) Repeat thousands of times.

From this, we can use the bootstrap distribution to estimate properties of the sampling distribution of the statistic, like mean, variance etc.

May want to use when difficult or impossible to get new samples, like an observational study or clinical trial.

Splines

Can decompose into points in d domains and fit polynomials with a given basis

1. interpolation: passing through points exactly
2. approximation: fitting to minimize residuals
3. these offer a parametric set of curves (a small number of parameters).
4. a parametric function could be linear regression, where βs are the parameters.
5. note that a non-parameterized curve has some inputs and outputs, but without specifying a parameterized function.

GAMs

$$g(E(Y)) = \beta_0 + f_1(X_1) + ... + f_p(X_p)$$

where $f_i()$ are smooth functions of the predictor variables X_i. This allows for a non-linear relationship between predictors and response variable.

Assumptions:

1. linearity in parameters
2. independent observations
3. assumption about response variable, Y

PCA

Say we have n observations on a set of p features, $X_1, X_2, ..., X_p$ and want to visualize them for EDA. There are way too many variables to look at $\binom{p}{2}$. We want a low-dimensional representation of the data that captures as much information as possible. The first *principal component* is a normalized (all squared ϕ's sum to 1) linear combination of the features:

$$Z_1 = \phi_{11}X_1 + \ldots + \phi_{p1}X_p$$

which has the LARGEST variance. The ϕ's are the loadings. Combined we have the loading vector $\phi_1 = (\phi_{11}...\phi_{p1})^T$.

Computing

Since we only care about variance, assume every variable has mean 0. Conceptually we want to maximize variance subject to the constraint that the squared ϕ's sum to 1, which we solve with the SVD.

1. Find the covariance matrix of your data, Σ
2. Compute the SVD, $\Sigma_{var} = U\Sigma V^T$
3. The principal components are the columns of U, so the first PC is the first column of U.
4. Then, we perform dimensionality reduction by projecting the data onto the space spanned by the principal components. This is done by multiplying the data matrix X by the principal components. The space spanned by two vectors in R^3 is a plane, so every combination of those two vectors forms a subspace. ISL says this is the space spanned by $span(\phi_1, \phi_2, \phi_3)$, maybe confusing terminology or something.
5. The next loading vector is orthogonal to ϕ_1.

---

4 Regression and Classification Methods

Neural Networks

Takes an input vector of p variables, $X_1, X_2, ..., X_p$, and builds a nonlinear function $f(X)$ to predict Y. Other nonlinear models are trees, boosting, and GAMs. NN's just have a particular structure.

Say we have p = 4. Those 4 covariates get fed into the input layer. These get fed into K hidden units. Initially, the model is:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$

$A_k = h_k(X)$ is just some nonlinear transformation of a linear combination of the inputs. The functions are learned during training.

These are then activated and fed into the output layer:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

The output layer is a linear model that uses these activations A_k as inputs, resulting in $f(X)$.

Most common activation function is the ReLU:

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & otherwise \end{cases}$$

The hidden units are the "neurons" because they fire or don't based on the sigmoid activation.

Computing

1. Choose a loss function: depends on type of response var. Measures the discrepancy between predicted and actual values.
2. gradient descent to move in the direction that minimizes the loss function the most.
3. forward pass and back propagation for number of epochs, compute the errors layer by layer
4. update params
5. iterate for number of epochs n or until convergence criteria are met

Tree-based Methods

Partition the feature space into a set of rectangles and fit a simple model in each one.

CART - Classification and Regression Tree

Start with a unit square. Recursively split each region (to reduce error etc.) at different constants. Classify a new point as whatever region the point falls in. (p. 268 esl). Solved with a greedy algorithm to find best split points. A **greedy algorithm** tries to find the best choice locally in the hopes of finding a global optimum. It chooses the choice that gives immediate benefit without considering the global consequences.
Tree depth is a tuning parameter. A very deep tree might overfit, but a shallow tree might not capture the important structure. Different measures of node impurity are misclassification error, gini index, and cross-entropy.
Can have high variance.

Boosting

Combining many weak classifiers to produce a strong committee. Number of iterations is a tuning parameter. Initialize some observation weights as 1/N for each obs. For m iterations, fit the jth classification model, like a tree. Then compute some errors and update the weights. Output the sign of the aggregated classifiers.

The success is because you're fitting an additive expansion in a set of elementary "basis" functions. The basis functions here are the classifiers $G_m(x) \in [-1, 1]$:

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

Where β are the coefficients and $b(x;\gamma)$ are usually some simple functions of x with param γ.

Assumptions:

Bagging

Short for bootstrap aggregating. An ensemble method.

1. create multiple datasets by randomly sampling the original with replacement.
2. train a model on each sample - each model learns slightly different parameters based on different data.
3. predictions are made by aggregating predictions of each model. For regression, average all the predictions. For classification, take majority vote or some probability based approach.

Random Forest

Considered less interpretable compared to a single tree.

1. A collection of decision trees, each of them built with a random subset of features and random subset of data (with replacement).
2. For each tree: 1) randomly select features 2) randomly select data 3) grow the tree until you meet some break criteria like tree depth.
3. For classification, take majority vote. For regression, average.
4. Reduces overfitting compared to a single tree, since you're aggregating the votes. Also robust to noise and outliers.

---

5 Clustering:

K-means

KNN

We attempt to solve the unattainable Bayes' Classifier by estimating the conditional distribution of Y given X, then classify a point based on the class with the highest estimated probability. With an integer K and a test observation x_0, we estimate that probability for class j as the fraction of points in N_0, the cluster, whose responses equal j:

$$P(Y = j|X = x_0) = \frac{1}{K}\sum_{i \in N_0} I(y_i = j)$$

You train the points using an integer K, and this generates a decision boundary.

Hierarchical Clustering

1. start with each node and merge with the nearest node based on distance
2. repeat until only 1 cluster is left or until some criteria (# of clusters) is met.

Agglomerative is the most popular, divisive starts with everything in one cluster and recursively splits the least similar nodes.

You DONT have to specify the number of clusters in advance, can trim after it's fit.

GMMs

Assume that the data is generated from a mixture of gaussian distributions.

solving:

1. each point is given a probabiliy using Bayes' theorem based on distance to mean and covariance
2. the params and mixing coefficients (weights) of the Gaussians are estimated by the EM algo.
3. E step computes posterior probabilities of each data point given each gaussian given current param estimates. M step updates gaussian parameters to maximize the likelihood based on posterior probabilities calculated in E step.

6 General Concepts

6.1 Kernel Methods

The kernel transforms the data into a higher dimensional space. If the data is, say, not linearly separable in R^2, it might be separable by a hyperplane in R^3 (commonly used in SVM, that's how we get those cool squiggly lines that separate the data). The kernel function is $K(x, x')$ where these are data points of the input space. The function is trying to find the similarity between different points.

The kernel trick is that we only have to know the inner product, not the specific coordinates of things in these higher dimensional spaces.

6.2 Regularization

Add a penalty to an objective function to reduce overfitting but only slightly increase bias. When might we use either type? L1 is more strict and sets some coefficients to 0, so it performs variable selection.

6.3 Gradient Descent

Goal is to optimize (minimize) a cost or loss function.

$$x_{t+1} = x_1 - \alpha_t \nabla f(x_t)$$

Until we meet some sort of break condition. α_t is the learning rate, which is a hyperparameter. Algo moves in this direction each step. But, this process can be expensive or get stuck in local minima or saddle points.

6.4 Stochastic Gradient Descent

The randomness is that it performs this process on mini batches sampled from the data set. It also randomizes the starting values, like weights and biases, by sampling from a normal or uniform distribution

7 SVD

Variable Selection

Why?

1. Performing selection can increase bias but reduce variance enough to improve prediction accuracy.

2. Interpretation: with a larges number of variables, we might want to interpret only a small subset

Bias-Variance

1. with low model complexity, we have high bias but low variance
2. with high complexity, high variance but low bias. Overfitting here.
3. very strong assumptions like linear regression will give high bias but low variance, something like a neural net can have high variance but low bias because we're not making many assumptions.
4. overly flexible models capture too much noise, while overly simple models will ignore signal

Decomposition: for an input point x_0, using squared error loss:
$\text{Error}(x_0) = E[(Y - \hat{f}(x_0))^2 | X = x_0]$
$= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$

Model Selection

1. split into training, validation, and test, depending on amount of data
2. AIC: It takes into account the model's goodness of fit as well as its complexity, penalizing models that are overly complex.
3. BIC: is similar to AIC but places a stronger penalty on models with more parameters.
4. Deviance: used for GLMs to generalize the concept of RSS to non-linear link functions and non-normal response variables.

Types of Loss Functions and When to Use

1. MSE: linear reg / NN for regression. Used because it penalizes large errors quadratically while being smooth and easy to optimize.
2. binary cross-entropy: Encourages output close to 0 or 1 for the correct class. Used for used w/ binary classification tasks, logistic reg or NN for binary classification
3. categorical cross-entropy: multiclass classification tasks. Encourage high probabilities for correct class.
4. Hinge Loss: SVMs. Penalizes misclassifications more severely as they move away from devision boundary.

Cross Validation

1. K-Fold: divide set into K folds. Test on the fold, and train using remaining folds. Repeat K times and take the mean.

2. LOOCV: same thing but with one observation left out.

Then apply to the test set. Could divide whole dataset depending on amount of data.

ROC Curve

Plot the true positive rate of a classifier on the y axis and false positive rate on the x. Each point is calculated by varying the threshold value (e.g. ¿ 0.5 is classified as point 1). A line that hugs top left is desirable. Diag line from (0,0) to (1,1) represents the performance of a random classifier. **AUC** represents the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. A perfect classifier has AUC = 1 while a random classifier has AUC = 0.5.

Penalties

L1 or L2 penalties encourage simpler models and penalzie larger weights.

Spaces

1. feature space: the n-dimensional space of features we have. If we're predicting Y, and have 3 features, X, the feature space is R^3. If we do, say, $X_4 = \frac{X_1}{X_3}$, we've expanded to R^4 with some function that "maps" to the expanded feature space.

2. sample space: possible outcomes of an experiment. So rolling a dice, sample space is {1,2,3,4,5,6}

3. parameter space: (also called the hypothesis space or model space) is the space of all possible parameter values. The logistic map $x_{n+1} = rx_n(1 - x_n)$ has one parameter, r, which can take any positive value. The parameter space is therefore positive real numbers.

4. input space: the sets of all possible inputs into a model