Diagonalizable Matrices

- A matrix R is *diagonalizable* when it can be written as a product shown below
- P is a non-singular matrix
- Called a *similarity transformation*
  - How an operator changes under a change of basis

$$R = P^{-1}AP \tag{1}$$

Define Polynomials of Matrices

- Polynomial Algebra of matrices:
  - 1 corresponds to identity
  - scalar multiplication
  - powers are products of matrix with itself or iterated compositions
- Cayley-Hamilton theorem: any matrix satisfies its own characteristic polynomial

$$f(A) = a_n A^n + \cdots + a_1 A + a_0 I \tag{2}$$

Characteristic Polynomial

- The Characteristic Polynomial of a matrix is computed by means of a special determinant
- The roots of characteristic polynomial are the *eigenvalues*
- Eigenvectors are vectors satisfying the expression

$$\Delta(\lambda) = |(\lambda I - A)| \tag{3}$$

$$Rv = \lambda v \tag{4}$$

$$0 = (\lambda I - R)v \tag{5}$$

Diagonal Factorization, Spectral Decomposition

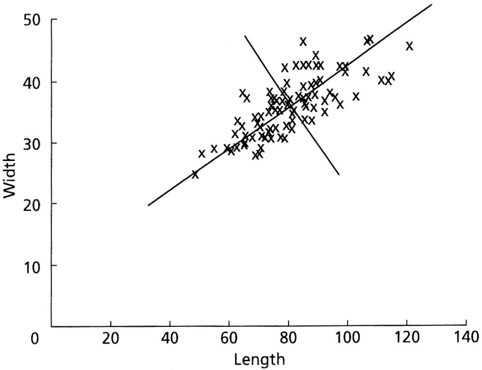
- Reversing the process we can write a matrix as a similarity transformation of a diagonal matrix
- If A is real-symmetric then the roots of its characteristic polynomial are guaranteed to be real.
- Eigenvectors belonging to distinct eigenvalues are orthogonal to one another $u \cdot v = 0$

$$A = PDP^{-1} \tag{6}$$

Principal Component Analysis in Two Dimensions

Box 1

A (supervised) ML workflow will minimally contain: *data*,*feature selection process*, *cross-validation*,*an ML algorithm*, and a means to *evaluate* performance



Model Evaluation

- Would like to assess, quantitatively, how well our algorithm can make predictions
- *Resubstitution Loss*: Performance of the classifier on the training data set
- Receiver Operating Characteristic (ROC): Visualize how the sensitivity and specificity of a test changes as some threshold parameter is varied

Cross-Validation

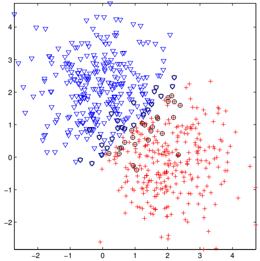
- Separation of Data into Training and Test sets
- k-Fold: The data-set is split into k separate partitions, each partition is called a *fold*
- We perform our evaluation k times, withholding a different fold from the training set in each case
- The performance of the result is reported as the average over all of the separate folds
- *leave-one-out* CV is a limiting case where each fold is a singleton

Ensemble Methods

- Goal: Combine a number of weak-predictors into a single strong-predictor
- *Bagging* (Bootstrap Aggregating): Train the same machine learning algorithm on a set of distinct bootstrap samples and combine the results by majority voting
- Make a predictor more robust by getting a variety of different views of a dataset

Basic Two-Outcome Tests

- Purpose of machine learning is to create some function in feature space that allows us to classify data points as belonging to some known classes
- Confusion matrix is used to characterize the different types of error that can occur



Performance Metrics

- In any binary classification task two types of error can occur, false positives and false negatives
- In general we need to keep track of both of these errors to understand how well our classifier is performing
- The *Receiver Operating Characteristic* (ROC) keeps track of both of these error rates as we vary the threshold

$$\text{threshold} > w^T x \tag{7}$$

Different Types of Error, Classifier Performance

$$\text{Confusion Matrix} = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \tag{8}$$

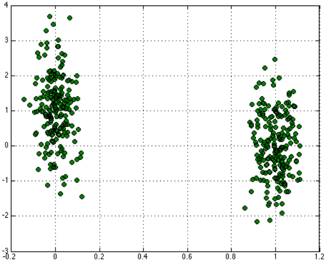
- Accuracy: (TP + TN) / TOTAL
- Sensitivity: TP / (TP + FN)
- Specificity: TN / (FP + TN)

Making Distinction Between Covariance and Confusion

- The *covariance matrix* describes how two quantities vary in relation to one another
- The diagonal terms are simply the variance of that particular variable and the off diagonal correspond to each pair
- The *confusion matrix* enters when we have a classification problem, it is a metric of how well some classification procedure performed compared to known values

An example

- Typical classification problem
- In this case, data consists of ordered pairs such that $(\text{class label} \in (0,1), \text{variate})$
- Various calculations we can do with this data



We can look at the covariance and correlation

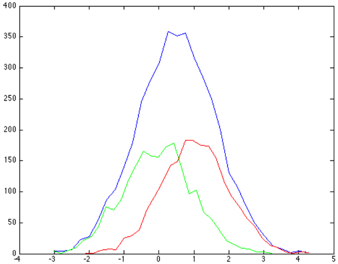
- Measure the natural tendency of these two numbers to vary together
- In this case, a p-value can be recovered from the correlation and is of the order 10^{-24}

$$R_{cov} = \begin{bmatrix} 1.1477 & -0.2549 \\ -0.2549 & 0.2506 \end{bmatrix} \tag{9}$$

$$R_{corr} = \begin{bmatrix} 1.0000 & -0.4752 \\ -0.4752 & 1.0000 \end{bmatrix} \tag{10}$$

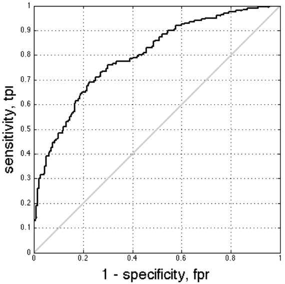
We can also examine the histogram

- With enough data points, the histogram becomes smooth and we can see how each individual class contributes to the total distribution
- The means are clearly different, but a simple threshold classifier won't perform well since there is a lot of overlap

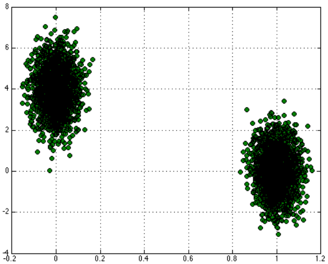


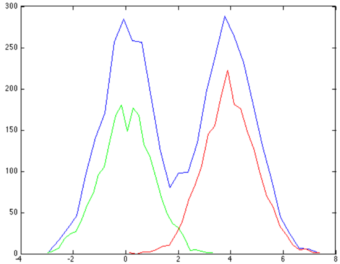
Receiver Operating Characteristic

- The Receiver Operating Characteristic (ROC) summarizes all possible threshold tests and presents in a space representing sensitivity and specificity
- The *area under the curve*, in this case ≈ 0.77 is another measure of the relationship between the variate and the class label
- What would a perfect or near-perfect test look like?

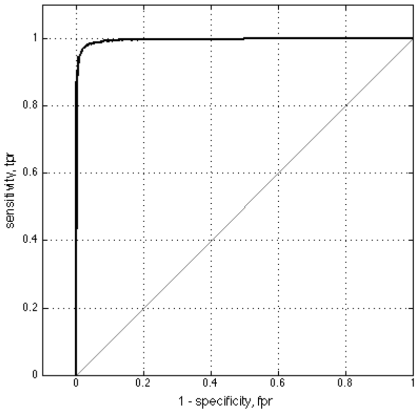


More Favorable Data





More Favorable Data ROC



Steps for Performing LDA

- Linear Discriminant Analysis (Binary)
  - Compute the within class means μ_1 and μ_2
  - Compute sample covariances in each class S_1 and S_2
  - Compute the within class scatter matrix $S_W = S_1 + S_2$
  - Find the discriminant projection w according to the equation (5) below
  - Choose a value for the threshold

$$\hat{w} = S_W^{-1}(\mu_1 - \mu_2) \tag{11}$$

$$\text{threshold} > \hat{w}^T x \tag{12}$$

Calculations

- Suppose we want to find the covariances for the datasets below
- Use a library or do calculation manually

$$\mathbf{A} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad (13)$$

General Solution

- Suppose we want to find the covariances for the datasets below
- Use a library or do calculation manually

$$R_{ij} = \sigma_{ij}^2 = E[(X_i - \mu_{x_i})(X_j - \mu_{x_j})] \tag{14}$$

$$\frac{1}{5}\left(X - \begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}[\mu_1, \mu_2]\right)^T \left(X - \begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}[\mu_1, \mu_2]\right) \tag{15}$$

Cases A and B

Case A:

$$\frac{1}{5}\left(\begin{bmatrix}0&0\\0&0\\0&0\\0&0\\0&0\\0&0\end{bmatrix}-\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}[0,0]\right)^T\left(\begin{bmatrix}0&0\\0&0\\0&0\\0&0\\0&0\\0&0\end{bmatrix}-\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}[0,0]\right)=R_{A,ij}=\begin{bmatrix}0&0\\0&0\end{bmatrix}\tag{16}$$

Case B:

$$\frac{1}{5}\left(\begin{bmatrix}0&0\\0&0\\0&0\\0&1\\0&0\\0&0\end{bmatrix}-\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}[0,1/6]\right)^T\left(\begin{bmatrix}0&0\\0&0\\0&0\\0&1\\0&0\\0&0\end{bmatrix}-\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}[0,1/6]\right)=R_{B,ij}=\begin{bmatrix}0&0\\0&1/6\end{bmatrix}\tag{17}$$

Cases C and D

- Sample covariances for cases C and D

Case C:

$$R_{C,ij} = \begin{bmatrix} 2/3 & 0 \\ 0 & 0 \end{bmatrix} \tag{18}$$

Case D:

$$R_{D,ij} = \begin{bmatrix} 3/10 & -3/10 \\ -3/10 & 3/10 \end{bmatrix} \tag{19}$$

Task

- Write a python script to verify these results
- Calculate covariance matrices by hand and also with `numpy.cov`

Matrix Calculus

- Gradient descent for machine learning is most often carried out over *vector* arguments
- In this case we just differentiate component by component and sometimes this allows us to collect terms in a compact notation
- See some familiar derivative identities compared to single-variable calculus

$$\frac{\partial \mathbf{x}^T A \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T) \tag{20}$$

Matrix Calculus: Differentiating a scalar by a vector

- For some scalar quantity $u(\mathbf{x})$ that depends on a vector $\mathbf{x}$ the derivative is another vector of the partials with respect to each component

$$\frac{\partial u}{\partial \mathbf{x}} = \left[\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}, \ldots, \frac{\partial u}{\partial x_n}\right] \tag{21}$$

- What is the derivative of a vector with respect to another vector?

Matrix Calculus: chain and product rules

- Same basic rules of differentiation apply when we differentiate with respect to matrices or vectors

$$\frac{\partial \mathbf{u} \cdot \mathbf{v}}{\partial \mathbf{x}} = ? \tag{22}$$

- Derivative of a dot product of two vectors u and v that may depend on a third x? What rule might we want to apply here?

Matrix notation for large

- A practical machine learning problem may involve vectors and matrices of large dimension
- Matrix notation allows us a convenient method of keeping track of all of the independent parameters and express the update in a convenient fashion
- For example to do least squares for a simple matrix problem we find $|\mathbf{Ax} - \mathbf{b}|^2$

$$\nabla \text{ w.r.t } \mathbf{x} = 2A^T(Ax - b) \tag{23}$$