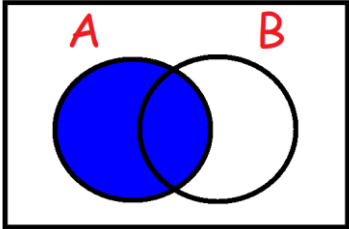
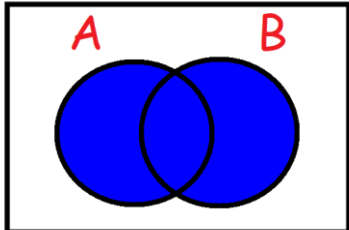
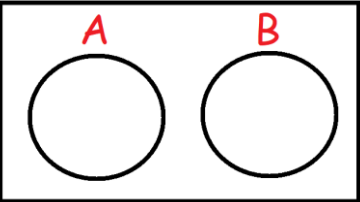
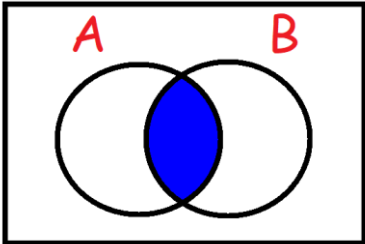
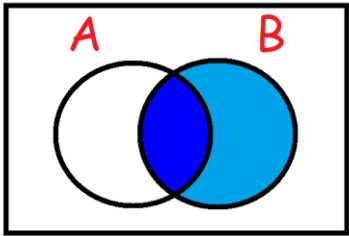
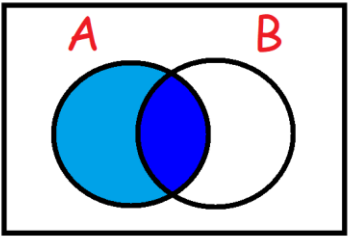
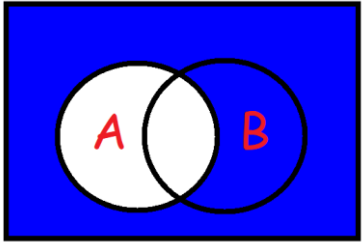
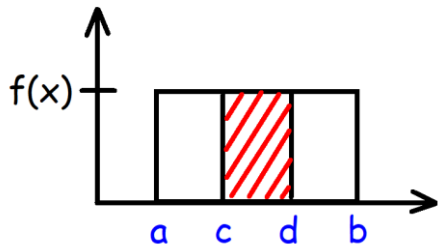


Probability Formula Sheet:

<p style="text-align: center;">$P(A)$</p> 	<p>Marginal Probability:</p> $P(A) = \frac{\text{Number of Successful Outcomes}}{\text{Total Possible Outcomes}}$ $0 \leq P(A) \leq 1$ $P(A') = 1 - P(A)$
<p style="text-align: center;">$P(A \cup B)$</p> 	<p>Union Probability: $P(A \cup B) = P(A \text{ or } B)$</p> <p>Addition Rule:</p> $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$
<p style="text-align: center;">$P(A \cap B) = 0$</p> 	<p>Mutually Exclusive Events:</p> $P(A \text{ or } B) = P(A) + P(B)$
<p style="text-align: center;">$P(A \cap B)$</p> 	<p>Joint Probability: $P(A \cap B) = P(A \text{ and } B)$</p> <p>Multiplication Rule:</p> $P(A \text{ and } B) = P(A/B) \times P(B)$ $P(A \text{ and } B) = P(B/A) \times P(A)$ <p>Note: $P(A/B) \times P(B) = P(B/A) \times P(A)$</p> <p>Independent Events:</p> $P(A \text{ and } B) = P(A) \times P(B)$ $P(A/B) = P(A) \quad \text{and} \quad P(B/A) = P(B)$

<p style="text-align: center;">$P(A/B)$</p> 	<p>Conditional Probability:</p> $P(A/B) = \frac{P(A \text{ and } B)}{P(B)}$ $P(B/A) = \frac{P(A \text{ and } B)}{P(A)}$
<p style="text-align: center;">$P(B/A)$</p> 	<p>Bayes Theorem:</p> $P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}$ $P(B/A) = \frac{P(A/B) \times P(B)}{P(A)}$
<p style="text-align: center;">$P(A')$</p> 	<p>The Complement / Negation:</p> $P(A') = 1 - P(A)$
<p>Things to Know:</p> <p>$P(AB)$ → 1st A, then B $P(BA)$ → 1st B, then A</p> <p>$P(A \text{ and } B)$: A and B occur simultaneously.</p> $P(A \text{ and } B) = P(B \text{ and } A)$	<p>Compound Probability:</p> <p>“Independent Events – With Replacement”</p> $P(AB) = P(A) \times P(B)$ <p>Dependent Events: “Without Replacement”</p> $P(AB) \neq P(BA)$
	<p>Expected Value:</p> $E(X) = X_1P_1 + X_2P_2$ <p> X_1 → Positive Value of Winning X_2 → Negative Value of Losing P_1 → Probability of Winning (decimal) P_2 → Probability of Losing (decimal) </p>

<p>Binomial Distribution:</p> <p>$P(x) \rightarrow$ Probability of 'x' successes in 'n' trials.</p> <p>$p \rightarrow$ Probability of a successful event.</p> <p>$q \rightarrow$ Probability that the event will fail.</p>	<p>Probability:</p> $P(x) = \binom{n}{x} p^x q^{n-x}$ $P(x) = \frac{n!}{(n-x)! x!} p^x q^{n-x}$ $u = np \quad \sigma = \sqrt{npq} \quad q = 1 - p$
<p>Geometric Distribution:</p> <p>$P(x) \rightarrow$ Probability that the nth event will succeed.</p> <p>$n \rightarrow$ number of 1st successful trial.</p> <p>$P(4) \rightarrow$ Probability that the 4th event will be successful.</p>	<p>Probability:</p> $P(X = n) = q^{n-1} * p$ $P(X > n) = q^n \quad P(X \geq n) = q^{n-1}$ $P(X \leq n) = 1 - q^n \quad P(X < n) = 1 - q^{n-1}$ $\sigma^2 = \frac{1}{p} \left(\frac{1}{p} - 1 \right) \quad \sigma = \frac{\sqrt{1-p}}{p}$ $u = 1/p \quad q = 1 - p$
<p>Geometric Probability:</p>	$P = \frac{\text{Shaded Area}}{\text{Total Area}}$
<p>Poisson Distribution:</p> <p>Mean: $u = \lambda = np$</p> <p>Variance: $\sigma^2 = np$</p> <p>Standard Deviation: $\sigma = \sqrt{np} = \sqrt{\lambda}$</p>	<p>Probability:</p> $P(X = n) = \frac{u^n e^{-u}}{n!} \quad \text{OR} \quad P(X = n) = \frac{\lambda^n e^{-\lambda}}{n!}$ $P(X > n) = 1 - e^{-u} \left[\sum_{x=0}^n \frac{u^x}{x!} \right]$ $P(X \leq n) = e^{-u} \left[\sum_{x=0}^n \frac{u^x}{x!} \right]$

Uniform Distribution: (Area = 1)

$$\text{Area} = f(x)(b - a)$$

$$f(x) = \frac{1}{b - a}$$

Probability:

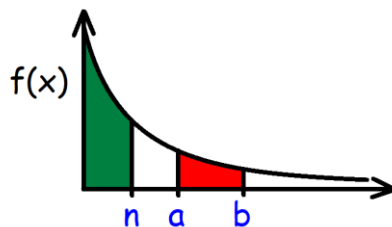
$$P(c \leq x \leq d) = \frac{d - c}{b - a}$$

$$P(a \leq x \leq c) = \frac{c - a}{b - a}$$

$$P(c \leq x \leq b) = \frac{b - c}{b - a}$$

$$u = \frac{a + b}{2}$$

$$\sigma = \frac{b - a}{\sqrt{12}}$$

Exponential Distribution: (Area = 1)

$\lambda \rightarrow$ Rate Parameter

$u \rightarrow$ Average time between occurrences

$$u = \frac{1}{\lambda} \quad \sigma^2 = \frac{1}{\lambda^2} \quad f(x) = \lambda e^{-\lambda x}$$

Probability:

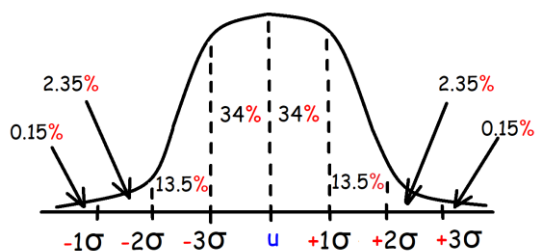
$$A_L = P(X \leq n) = 1 - e^{-\lambda n}$$

$$A_R = P(X \geq n) = e^{-\lambda n}$$

$$P(a \leq x \leq b) = e^{-\lambda a} - e^{-\lambda b}$$

$$A_L = \int_0^n \lambda e^{-\lambda x} dx = 1 - e^{-\lambda n} \quad A_R = \int_n^\infty \lambda e^{-\lambda x} dx = e^{-\lambda n}$$

$$P(a \leq x \leq b) = P(a < x < b) \quad \text{and} \quad P(X = a) = 0$$

Standard Normal Distribution:

$$\text{Area} = \int_a^b f(x) dx$$

Probability:

$$z = \frac{x - u}{\sigma} \quad x = u + z\sigma$$

$$u = np \quad \sigma = \sqrt{npq}$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2}$$

$$P(a \leq x \leq b) = \int_a^b \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2} dx = \text{Area}$$

Statistics Formula Sheet:

Mean:	Sample Mean: $\bar{X} = \frac{\sum X}{n}$	Population Mean: $u = \frac{\sum X}{N}$
Median: (Q2)	If n is odd: $M = \left(\frac{n+1}{2}\right)^{th} \text{ Term}$	If n is even: $M = \frac{\left(\frac{n}{2}\right)^{th} \text{ Term} + \left(\frac{n}{2} + 1\right)^{th} \text{ Term}}{2}$
Mode:	The number with the highest frequency.	
Range: H → Highest Value L → Lowest Value	$\text{Range} = H - L$	$\text{MidRange} = \frac{H + L}{2}$
Standard Deviation:	Sample: $s = \sqrt{\frac{\sum (X - \bar{X})^2}{n - 1}}$	Population: $\sigma = \sqrt{\frac{\sum (X - u)^2}{N}}$
Variance:	Sample Variance: $s^2 = \frac{\sum (X - \bar{X})^2}{n - 1}$	Population Variance: $\sigma^2 = \frac{\sum (X - u)^2}{N}$
Coefficient of Variation:	Sample CV: $CV = \frac{s}{\bar{X}} \times 100\%$	Population CV: $CV = \frac{\sigma}{u} \times 100\%$
Mean Absolute Deviation:	Sample MD (Mean): $MD = \frac{\sum X - \bar{X} }{n}$	Population MD (Mean): $MD = \frac{\sum X - u }{N}$
Average Deviation:	Sample AD: $AD = \frac{\sum (X - \bar{X})}{n}$	Population AD: $AD = \frac{\sum (X - u)}{N}$

Quartile:	$Q_k = k \left(\frac{n+1}{4} \right)^{th} Term$ $Q_1 = 1 \left(\frac{n+1}{4} \right)^{th} Term \quad Q_3 = 3 \left(\frac{n+1}{4} \right)^{th} Term$
Percentile:	$P_k = k \left(\frac{n+1}{100} \right)^{th} Term$ $P_{30} = 30 \left(\frac{n+1}{100} \right)^{th} Term \quad P_{70} = 70 \left(\frac{n+1}{100} \right)^{th} Term$
Decile:	$D_k = k \left(\frac{n+1}{10} \right)^{th} Term$
Octile:	$O_k = k \left(\frac{n+1}{8} \right)^{th} Term$
Interquartile Range:	$IQR = Q_3 - Q_1$
Quartile Deviation:	$QD = \frac{Q_3 - Q_1}{2} = \frac{1}{2}(IQR)$
Coefficient of Quartile Deviation:	$CQD = \frac{Q_3 - Q_1}{Q_3 + Q_1}$
Range of Outliers:	$[Q_1 - 1.5 IQR, Q_3 + 1.5 IQR]$ <p>Note: Any data point that exists outside of the range shown above is considered an outlier.</p>
Coefficient of Range:	$CR = \frac{H - L}{H + L}$

	General Formula:	Expanded Form:	2 Numbers:
Arithmetic Mean:	$\bar{X} = \frac{\sum X}{n}$	$\bar{X} = \frac{X_1 + X_2 + X_3 + \dots X_n}{n}$	$\textcolor{red}{AM} = \frac{a + b}{2}$
Geometric Mean:	$\bar{X}_G = \left(\prod_{i=1}^n X_i \right)^{\frac{1}{n}}$	$\bar{X}_G = (X_1 * X_2 * X_3 \dots X_n)^{1/n}$	$\textcolor{red}{GM} = \sqrt{ab}$
	$\bar{X}_G = 10^{\left(\frac{\sum \log(X)}{n} \right)}$	$\bar{X}_G = 10^{\left(\frac{\log(X_1) + \log(X_2) + \dots + \log(X_n)}{n} \right)}$	$\textcolor{red}{GM} = 10^{\frac{\log(a) + \log(b)}{2}}$
Weighted Mean:	$\bar{X}_W = \frac{\sum WX}{W}$	$\bar{X}_W = \frac{W_1X_1 + W_2X_2 + \dots + W_nX_n}{W_1 + W_2 + \dots + W_n}$	$\textcolor{red}{WM} = \frac{W_1a + W_2b}{W_1 + W_2}$
Harmonic Mean:	$\bar{X}_H = \frac{n}{\sum \left(\frac{1}{X} \right)}$	$\bar{X}_H = \frac{n}{\frac{1}{X_1} + \frac{1}{X_2} + \frac{1}{X_3} + \dots + \frac{1}{X_n}}$	$\textcolor{red}{HM} = \frac{2}{\frac{1}{a} + \frac{1}{b}} = \frac{2ab}{a + b}$
Root Mean Square:	$X_{rms} = \sqrt{\frac{\sum (X^2)}{n}}$	$X_{rms} = \sqrt{\frac{X_1^2 + X_2^2 + X_3^2 + \dots + X_n^2}{n}}$	$\textcolor{red}{X}_{rms} = \sqrt{\frac{a^2 + b^2}{2}}$
Mean Relationship:	$\textcolor{red}{GM} = \sqrt{(\textcolor{red}{AM})(\textcolor{red}{HM})} \quad \text{For 2 Numbers}$ $\sqrt{ab} = \sqrt{\left(\frac{\textcolor{red}{a} + \textcolor{red}{b}}{2} \right) \left(\frac{2ab}{\textcolor{red}{a} + \textcolor{red}{b}} \right)}$		

Statistics Formulas for Grouped Data:

Mean:	$\bar{X} = \frac{\sum fX_m}{\sum f} = \frac{\sum fX_m}{n}$
Midpoint of Range:	$X_m = \frac{X_1 + X_2}{2}$
Standard Deviation:	$s = \sqrt{\frac{\sum f(X_m - \bar{X})^2}{n - 1}} = \sqrt{\frac{\sum fX_m^2 - \frac{(\sum fX_m)^2}{n}}{n - 1}}$
Variance:	$s^2 = \frac{\sum f(X_m - \bar{X})^2}{n - 1} = \frac{\sum fX_m^2 - \frac{(\sum fX_m)^2}{n}}{n - 1}$
1st Quartile:	$Q_1 = L_1 + \frac{w_1}{f_1} \left(\frac{n}{4} - C_1 \right)$ <p><i>L → Lower Class Boundary w → Width of Class Interval</i></p>
Median – 2nd Quartile:	$\text{Median} = Q_2 = L_2 + \frac{w_2}{f_2} \left(\frac{n}{2} - C_2 \right)$ <p><i>f → frequency of quartile class n → total frequency</i></p>
3rd Quartile:	$Q_3 = L_3 + \frac{w_3}{f_3} \left(\frac{3n}{4} - C_3 \right)$ <p><i>C → Cumulative frequency of preceding quartile class.</i></p>
Mode:	$\text{Mode} = L + h \left(\frac{f_m - f_1}{2f_m - f_1 - f_2} \right)$ <p> <i>L → Lower boundary of modal class</i> <i>h → Size of class interval</i> <i>f_m → frequency of modal class</i> <i>f₁ → frequency of preceding class</i> <i>f₂ → frequency of succeeding class</i> </p>

MACHINE LEARNING CHEATSHEET

Summary of Machine Learning Algorithms descriptions, advantages and use cases. Inspired by the very good book and articles of *MachineLearningMastery*, with added math, and *ML Pros & Cons* of *HackingNote*. Design inspired by *The Probability Cheatsheet* of W. Chen. Written by Rémi Canard.

General

Definition

We want to learn a target function f that maps input variables X to output variable Y , with an error e :

$$Y = f(X) + e$$

Linear, Nonlinear

Different algorithms make different assumptions about the shape and structure of f , thus the need of testing several methods. Any algorithm can be either:

- **Parametric** (or **Linear**): simplify the mapping to a known linear combination form and learning its coefficients.
- **Non parametric** (or **Nonlinear**): free to learn any functional form from the training data, while maintaining some ability to generalize.

Linear algorithms are usually simpler, faster and requires less data, while Nonlinear can be are more flexible, more powerful and more performant.

Supervised, Unsupervised

Supervised learning methods learn to predict Y from X given that the data is labeled.

Unsupervised learning methods learn to find the inherent structure of the unlabeled data.

Bias-Variance trade-off

In supervised learning, the prediction error e is composed of the **bias**, the **variance** and the **irreducible** part.

Bias refers to **simplifying assumptions** made to learn the target function easily.

Variance refers to sensitivity of the model to changes in the training data.

The **goal of parameterization** is to achieve a **low bias** (underlying pattern not too simplified) and **low variance** (not sensitive to specificities of the training data) **tradeoff**.

Underfitting, Overfitting

In statistics, **fit** refers to how well the target function is approximated.

Underfitting refers to poor inductive learning from training data and poor generalization.

Overfitting refers to learning the training data detail and noise which leads to poor generalization. It can be **limited** by using resampling and defining a validation dataset.

Optimization

Almost every machine learning method has an optimization algorithm at its core.

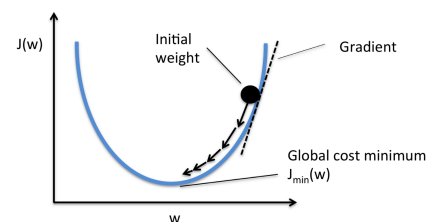
Gradient Descent

Gradient Descent is used to **find the coefficients** of f that **minimizes a cost function** (for example MSE, SSR).

Procedure:

- Initialization $\theta = 0$ (coefficients to 0 or random)
- Calculate cost $J(\theta) = \text{evaluate}(f(\text{coefficients}))$
- Gradient of cost $\frac{\partial}{\partial \theta_j} J(\theta)$ we know the uphill direction
- Update coeff $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ we go downhill

The cost updating process is repeated until convergence (minimum found).



Batch Gradient Descent does summing/averaging of the cost over all the observations.

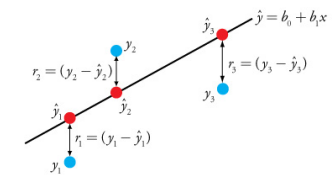
Stochastic Gradient Descent apply the procedure of parameter updating for each observation.

Tips:

- Change **learning rate** α ("size of jump" at each iteration)
- Plot **Cost vs Time** to assess learning rate performance
- Rescaling the input variables
- Reduce passes through training set with SGD
- Average over 10 or more updated to observe the learning trend while using SGD

Ordinary Least Squares

OLS is used to find the estimator $\hat{\beta}$ that **minimizes the sum of squared residuals**: $\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 = y - X \hat{\beta}$



Using linear algebra such that we have $\hat{\beta} = (X^T X)^{-1} X^T y$

Maximum Likelihood Estimation

MLE is used to find the estimators that **minimizes the likelihood function**:

$$\mathcal{L}(\theta|x) = f_{\theta}(x) \quad \text{density function of the data distribution}$$

Linear Algorithms

All linear Algorithms assume a linear relationship between the input variables X and the output variable Y .

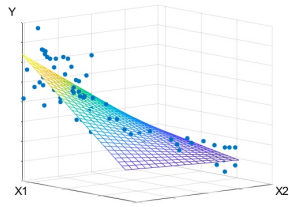
Linear Regression

Representation:

A LR model representation is a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_i x_i$$

β_0 is usually called intercept or **bias** coefficient. The dimension of the hyperplane of the regression is its **complexity**.



Learning:

Learning a LR means estimating the coefficients from the training data. Common methods include **Gradient Descent** or **Ordinary Least Squares**.

Variations:

There are extensions of LR training called **regularization** methods, that aim to **reduce the complexity** of the model:

- **Lasso Regression:** where OLS is modified to minimize the sum of the coefficients (L1 regularization)

$$\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \sum_{j=1}^p |\beta_j|$$

- **Ridge Regression:** where OLS is modified to minimize the squared sum of the coefficients (L2 regularization)

$$\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2$$

where $\lambda \geq 0$ is a tuning parameter to be determined.

Data preparation:

- Transform data for linear relationship (ex: log transform for exponential relationship)
- Remove noise such as outliers
- Rescale inputs using standardization or normalization

Advantages:

- + Good regression baseline considering simplicity
- + Lasso/Ridge can be used to avoid overfitting
- + Lasso/Ridge permit feature selection in case of collinearity

Usecase examples:

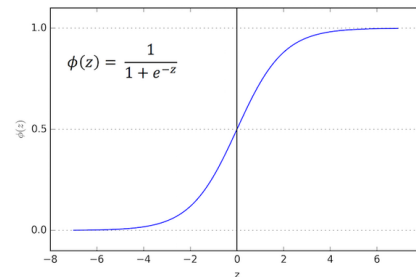
- Product sales prediction according to prices or promotions
- Call-center waiting-time prediction according to the number of complaints and the number of working agents

Logistic Regression

It is the go-to for **binary classification**.

Representation:

Logistic regression a linear method but predictions are transformed using the **logistic function** (or sigmoid):



ϕ is S-shaped and map real-valued number in (0,1).

The representation is an equation with binary output:

$$y = \frac{e^{\beta_0 + \beta_1 x_1 + \dots + \beta_i x_i}}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_i x_i}}$$

Which actually models the probability of default class:

$$p(X) = \frac{e^{\beta_0 + \beta_1 x_1 + \dots + \beta_i x_i}}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_i x_i}} = p(Y = 1|X)$$

Learning:

Learning the Logistic regression coefficients is done using **maximum-likelihood estimation**, to predict values close to 1 for default class and close to 0 for the other class.

Data preparation:

- Probability transformation to binary for classification
- Remove noise such as outliers

Advantages:

- + Good classification baseline considering simplicity
- + Possibility to change cutoff for precision/recall tradeoff
- + Robust to noise/overfitting with L1/L2 regularization
- + Probability output can be used for ranking

Usecase examples:

- Customer scoring with probability of purchase
- Classification of loan defaults according to profile

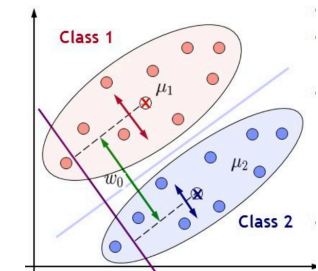
Linear Discriminant Analysis

For **multiclass classification**, LDA is the preferred linear technique.

Representation:

LDA representation consists of **statistical properties** calculated for **each class**: **means** and the **covariance matrix**:

$$\mu_k = \frac{1}{n_k} \sum_{i=1}^n x_i \text{ and } \sigma^2 = \frac{1}{n-K} \sum_{i=1}^n (x_i - \mu_k)^2$$



LDA assumes **Gaussian** data and attributes of **same σ^2** .

Predictions are made using **Bayes Theorem**:

$$P(Y = k|X = x) = \frac{P(k) \times P(x|k)}{\sum_{l=1}^K P(l) \times P(x|l)}$$

to obtain a discriminate function (latent variable) for each class k , estimating $P(x|k)$ with a Gaussian distribution:

$$D_k(x) = x \times \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \ln(P(k))$$

The class with largest **discriminant value** is the **output class**.

Variations:

- **Quadratic DA:** Each class uses its own variance estimate
- **Regularized DA:** Regularization into the variance estimate

Data preparation:

- Review and modify univariate distributions to be Gaussian
- Standardize data to $\mu = 0$, $\sigma = 1$ to have same variance
- Remove noise such as outliers

Advantages:

- + Can be used for dimensionality reduction by keeping the latent variables as new variables

Usecase example:

- Prediction of customer churn

Nonlinear Algorithms

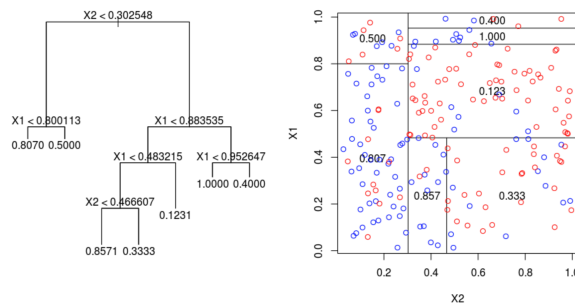
All Nonlinear Algorithms are non-parametric and more flexible. They are not sensible to outliers and do not require any shape of distribution.

Classification and Regression Trees

Also referred as CART or Decision Trees, this algorithm is the foundation of Random Forest and Boosted Trees.

Representation:

The model representation is a **binary tree**, where each **node** is an **input variable** x with a split point and each **leaf** contain an **output variable** y for prediction.



The model actually **split the input space** into (hyper) rectangles, and predictions are made according to the **area** observations *fall* into.

Learning:

Learning of a CART is done by a greedy approach called **recursive binary splitting** of the input space:

At each step, the best **predictor** X_j and the best **cutpoint** s are selected such that $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ **minimizes the cost**.

- For **regression** the cost is the **Sum of Squared Error**:

$$\sum_{i=1}^n (y_i - \hat{y})^2$$

- For **classification** the cost function is the **Gini index**:

$$G = \sum_{k=1}^n p_k(1 - p_k)$$

The Gini index is **an indication of how pure are the leaves**, if all observations are the same type $G=0$ (perfect purity), while a 50-50 split for binary would be $G=0.5$ (worst purity).

The most common **Stopping Criterion** for splitting is a minimum of **training observations per node**.

The simplest form of pruning is **Reduced Error Pruning**: Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected, then the change is kept

Advantages:

- + Easy to interpret and no overfitting with pruning
- + Works for both regression and classification problems
- + Can take any type of variables without modifications, and do not require any data preparation

Usecase examples:

- Fraudulent transaction classification
- Predict human resource allocation in companies

Naive Bayes Classifier

Naive Bayes is a **classification** algorithm interested in selecting the **best hypothesis h given data d** assuming there is no interaction between features.

Representation:

The representation is the based on Bayes Theorem:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)}$$

with naïve hypothesis $P(h|d) = P(x_1|h) \times \dots \times P(x_i|h)$

The prediction is the **Maximum A posteriori Hypothesis**:

$$MAP(h) = \max(P(h|d)) = \max(P(d|h) \times P(h))$$

The denominator is not kept as it is only for normalization.

Learning:

Training is **fast** because only **probabilities** need to be calculated:

$$P(h) = \frac{\text{instances}_h}{\text{all instances}} \quad \text{and} \quad P(x|h) = \frac{\text{count}(x \wedge h)}{\text{instances}_h}$$

Variations:

Gaussian Naive Bayes can extend to numerical attributes by assuming a Gaussian distribution.

Instead of $P(x|h)$ are calculated with $P(h)$ during **learning**:

$$\mu(x) = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{and} \quad \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu(x))^2}$$

and MAP for **prediction** is calculated using Gaussian PDF

$$f(x|\mu(x), \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Data preparation:

- Change numerical inputs to categorical (binning) or near-Gaussian inputs (remove outliers, log & boxcox transform)
- Other distributions can be used instead of Gaussian
- Log-transform of the probabilities can avoid overflow
- Probabilities can be updated as data becomes available

Advantages:

- + Fast because of the calculations
- + If the naive assumptions works can converge quicker than other models. Can be used on smaller training data.
- + Good for few categories variables

Usecase examples:

- Article classification using binary word presence
- Email spam detection using a similar technique

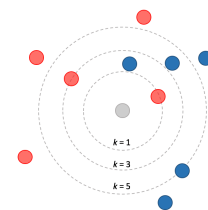
K-Nearest Neighbors

If you are similar to your neighbors, you are one of them.

Representation:

KNN uses the **entire training set**, **no training** is required.

Predictions are made by searching the **k similar instances**, according to a **distance**, and **summarizing the output**.



For **regression** the output can be the **mean**, while for **classification** the output can be the **most common class**.

Various distances can be used, for example:

- **Euclidean** Distance, good for **similar** type of variables

$$d(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

- **Manhattan** Distance, good for **different** type of variables

$$d(a, b) = \sum_{i=1}^n |a_i - b_i|$$

The **best value of k** must be found by **testing**, and the algorithm is **sensible** to the **Curse of dimensionality**.

Data preparation:

- Rescale inputs using standardization or normalization
- Address missing data for distance calculations
- Dimensionality reduction or feature selection for **COD**

Advantages:

- + Effective if the training data is large
- + No learning phase
- + Robust to noisy data, no need to filter outliers

Usecase examples:

- Recommending products based on similar customers
- Anomaly detection in customer behavior

Support Vector Machines

SVM is a go-to for high performance with little tuning

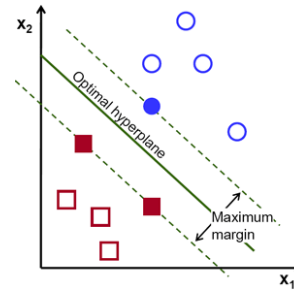
Representation:

In SVM, a **hyperplane** is selected to **separate the points** in the input variable space by their class, with the **largest margin**.

The closest datapoints (defining the margin) are called the **support vectors**.

But real **data cannot be perfectly separated**, that is why a **C** defines the amount of **violation** of the margin allowed.

The lower C, the more sensitive SVM is to training data.



The **prediction** function is the signed **distance** of the new input x to the separating hyperplane w :

$$f(x) = \langle w, x \rangle + \rho = w^T x + \rho \text{ with } \rho \text{ the bias}$$

Which gives for **linear kernel**, with x_i the support vectors:

$$f(x) = \sum_{i=1}^n a_i \times (x \times x_i) + \rho$$

Learning:

The hyperplane learning is done by transforming the problem using linear algebra, and minimizing:

$$\left[\frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i (\vec{w} \cdot \vec{x}_i - b)) \right] + \lambda \|\vec{w}\|^2$$

Variations:

SVM is implemented using various kernels, which define the measure between new data and support vectors:

- **Linear** (dot-product): $K(x, x_i) = \sum (x \times x_i)$
- **Polynomial**: $K(x, x_i) = 1 + \sum (x \times x_i)^d$
- **Radial**: $K(x, x_i) = e^{-\gamma \sum ((x - x_i)^2)}$

Data preparation:

- SVM assumes numeric inputs, may require dummy transformation of categorical features

Advantages:

- + Allow nonlinear separation with nonlinear Kernels
- + Works good in high dimensional space
- + Robust to multicollinearity and overfitting

Usecase examples:

- Face detection from images
- Target Audience Classification from tweets

Ensemble Algorithms

Ensemble methods use multiple, simpler algorithms combined to obtain better performance.

Bagging and Random Forest

Random Forest is part of a bigger type of ensemble methods called **Bootstrap Aggregation** or **Bagging**. Bagging can **reduce the variance** of high-variance models.

It uses the **Bootstrap statistical procedure**: estimate a quantity from a sample by creating many random subsamples with replacement, and computing the mean of each subsample.



Representation:

For **bagged decision trees**, the steps would be:

- Create many subsamples of the training dataset
- Train a CART model on each sample
- Given a new dataset, calculate the average prediction

However, combining models works best if submodels are **weakly correlated** at best.

Random Forest is a tweaked version of bagged decision trees to **reduce tree correlation**.

Learning:

During learning, each **sub-tree** can only access a random **sample of features** when **selecting the split points**. The size of the feature sample at each split is a parameter m .

A good default is \sqrt{p} for classification and $\frac{p}{3}$ for regression.

The **OOB** estimate is the performance of **each model on its Out-Of-Bag** (not selected) samples. It is a reliable estimate of test error.

Bagged method can provide **feature importance**, by calculating and averaging the **error function drop for individual variables** (depending of samples where a variable is selected or not).

Advantages:

In addition to the advantages of the CART algorithm

- + Robust to overfitting and missing variables
- + Can be parallelized for distributed computing
- + Performance as good as SVM but easier to interpret

Usecase examples:

- Predictive machine maintenance
- Optimizing line decision for credit cards

Boosting and AdaBoost

AdaBoost was the first successful boosting algorithm developed for binary classification.

Representation:

A boost classifier is of the form

$$F_T(x) = \sum_{t=1}^T f_t(x)$$

where each f_t is a **weak learner correcting the errors** of the previous one.

Adaboost is commonly used with decision trees with one level (**decision stumps**).

Predictions are made using the weighted average of the weak classifiers.

Learning:

Each training set instance is initially weighted $w(x_i) = \frac{1}{n}$

One *decision stump* is prepared using the weighted samples, and a **misclassification rate** is calculated:

$$\epsilon = \frac{\sum_{i=1}^n (w_i \times p_{error\ i})}{\sum_{i=1}^n w}$$

Which is the weighted sum of the misclassification rates, where w is the training instance i weight and $p_{error\ i}$ its prediction error (1 or 0).

A **stage value** is computed from the misclassification rate:

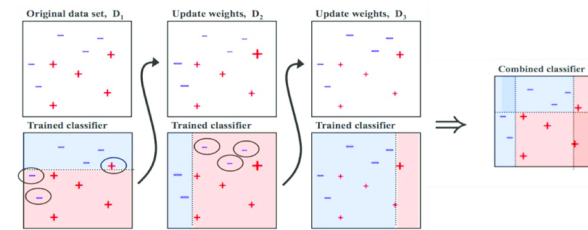
$$stage = \ln\left(\frac{1 - \epsilon}{\epsilon}\right)$$

This stage value is used to **update the instances weights**:

$$w = w \times e^{stage \times \epsilon}$$

The **incorrectly** predicted instance are given **more** weight.

Weak models are added sequentially using the training weights, until no improvement can be made or the number of rounds has been attained.



Data preparation:

- Outliers should be removed for AdaBoost

Advantages:

- + High performance with no tuning (only number of rounds)

Interesting Resources

Machine Learning Mastery website

- > <https://machinelearningmastery.com/>

Scikit-learn website, for python implementation

- > <http://scikit-learn.org/>

W.Chen probability cheatsheet

- > https://github.com/wzchen/probability_cheatsheet

HackingNote, for interesting, condensed insights

- > <https://www.hackingnote.com/>

Seattle Data Guy blog, for business oriented articles

- > <https://www.theseattledataguy.com/>

Explained visually, making hard ideas intuitive

- > <http://setosa.io/ev/>

This Machine Learning Cheatsheet

- > https://github.com/remicnrd/ml_cheatsheet

Eigenvalues and eigenvectors

The subject of eigenvalues and eigenvectors will take up most of the rest of the course. We will again be working with square matrices. Eigenvalues are special numbers associated with a matrix and eigenvectors are special vectors.

Eigenvectors and eigenvalues

A matrix A acts on vectors \mathbf{x} like a function does, with input \mathbf{x} and output $A\mathbf{x}$. *Eigenvectors* are vectors for which $A\mathbf{x}$ is parallel to \mathbf{x} . In other words:

$$A\mathbf{x} = \lambda\mathbf{x}.$$

In this equation, \mathbf{x} is an eigenvector of A and λ is an *eigenvalue* of A .

Eigenvalue 0

If the eigenvalue λ equals 0 then $A\mathbf{x} = 0\mathbf{x} = \mathbf{0}$. Vectors with eigenvalue 0 make up the nullspace of A ; if A is singular, then $\lambda = 0$ is an eigenvalue of A .

Examples

Suppose P is the matrix of a projection onto a plane. For any \mathbf{x} in the plane $P\mathbf{x} = \mathbf{x}$, so \mathbf{x} is an eigenvector with eigenvalue 1. A vector \mathbf{x} perpendicular to the plane has $P\mathbf{x} = \mathbf{0}$, so this is an eigenvector with eigenvalue $\lambda = 0$. The eigenvectors of P span the whole space (but this is not true for every matrix).

The matrix $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ has an eigenvector $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ with eigenvalue 1 and another eigenvector $\mathbf{x} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ with eigenvalue -1 . These eigenvectors span the space. They are perpendicular because $B = B^T$ (as we will prove).

$$\det(A - \lambda I) = 0$$

An n by n matrix will have n eigenvalues, and their sum will be the sum of the diagonal entries of the matrix: $a_{11} + a_{22} + \cdots + a_{nn}$. This sum is the *trace* of the matrix. For a two by two matrix, if we know one eigenvalue we can use this fact to find the second.

Can we solve $A\mathbf{x} = \lambda\mathbf{x}$ for the eigenvalues and eigenvectors of A ? Both λ and \mathbf{x} are unknown; we need to be clever to solve this problem:

$$\begin{aligned} A\mathbf{x} &= \lambda\mathbf{x} \\ (A - \lambda I)\mathbf{x} &= \mathbf{0} \end{aligned}$$

In order for λ to be an eigenvalue, $A - \lambda I$ must be singular. In other words, $\det(A - \lambda I) = 0$. We can solve this *characteristic equation* for λ to get n solutions.

If we're lucky, the solutions are distinct. If not, we have one or more *repeated eigenvalues*.

Once we've found an eigenvalue λ , we can use elimination to find the nullspace of $A - \lambda I$. The vectors in that nullspace are eigenvectors of A with eigenvalue λ .

Calculating eigenvalues and eigenvectors

Let $A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$. Then:

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} 3 - \lambda & 1 \\ 1 & 3 - \lambda \end{vmatrix} \\ &= (3 - \lambda)^2 - 1 \\ &= \lambda^2 - 6\lambda + 8. \end{aligned}$$

Note that the coefficient 6 is the trace (sum of diagonal entries) and 8 is the determinant of A . In general, the eigenvalues of a two by two matrix are the solutions to:

$$\lambda^2 - \text{trace}(A) \cdot \lambda + \det A = 0.$$

Just as the trace is the sum of the eigenvalues of a matrix, the product of the eigenvalues of any matrix equals its determinant.

For $A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$, the eigenvalues are $\lambda_1 = 4$ and $\lambda_2 = 2$. We find the eigenvector $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ for $\lambda_1 = 4$ in the nullspace of $A - \lambda_1 I = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$.

x_2 will be in the nullspace of $A - 2I = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. The nullspace is an entire line; x_2 could be any vector on that line. A natural choice is $x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

Note that these eigenvectors are the same as those of $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Adding $3I$ to the matrix $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ added 3 to each of its eigenvalues and did not change its eigenvectors, because $A\mathbf{x} = (B + 3I)\mathbf{x} = \lambda\mathbf{x} + 3\mathbf{x} = (\lambda + 3)\mathbf{x}$.

A caution

Similarly, if $A\mathbf{x} = \lambda\mathbf{x}$ and $B\mathbf{x} = \alpha\mathbf{x}$, $(A + B)\mathbf{x} = (\lambda + \alpha)\mathbf{x}$. It would be nice if the eigenvalues of a matrix sum were always the sums of the eigenvalues, but this is only true if A and B have the same eigenvectors. The eigenvalues of the product AB aren't usually equal to the products $\lambda(A)\lambda(B)$, either.

Complex eigenvalues

The matrix $Q = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ rotates every vector in the plane by 90° . It has trace $0 = \lambda_1 + \lambda_2$ and determinant $1 = \lambda_1 \cdot \lambda_2$. Its only real eigenvector is the zero vector; any other vector's direction changes when it is multiplied by Q . How will this affect our eigenvalue calculation?

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} -\lambda & -1 \\ 1 & -\lambda \end{vmatrix} \\ &= \lambda^2 + 1. \end{aligned}$$

$\det(A - \lambda I) = 0$ has solutions $\lambda_1 = i$ and $\lambda_2 = -i$. If a matrix has a complex eigenvalue $a + bi$ then the *complex conjugate* $a - bi$ is also an eigenvalue of that matrix.

Symmetric matrices have real eigenvalues. For *antisymmetric* matrices like Q , for which $A^T = -A$, all eigenvalues are imaginary ($\lambda = bi$).

Triangular matrices and repeated eigenvalues

For triangular matrices such as $A = \begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}$, the eigenvalues are exactly the entries on the diagonal. In this case, the eigenvalues are 3 and 3:

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} 3 - \lambda & 1 \\ 0 & 3 - \lambda \end{vmatrix} \\ &= (3 - \lambda)(3 - \lambda) \quad \left(= (a_{11} - \lambda)(a_{22} - \lambda) \right) \\ &= 0, \end{aligned}$$

so $\lambda_1 = 3$ and $\lambda_2 = 3$. To find the eigenvectors, solve:

$$(A - \lambda I)\mathbf{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} = \mathbf{0}$$

to get $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. There is no independent eigenvector \mathbf{x}_2 .

MIT OpenCourseWare
<http://ocw.mit.edu>

18.06SC Linear Algebra
Fall 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.

Diagonalization and powers of A

We know how to find eigenvalues and eigenvectors. In this lecture we learn to *diagonalize* any matrix that has n independent eigenvectors and see how diagonalization simplifies calculations. The lecture concludes by using eigenvalues and eigenvectors to solve *difference equations*.

Diagonalizing a matrix $S^{-1}AS = \Lambda$

If A has n linearly independent eigenvectors, we can put those vectors in the columns of a (square, invertible) matrix S . Then

$$\begin{aligned} AS &= A \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 \mathbf{x}_1 & \lambda_2 \mathbf{x}_2 & \cdots & \lambda_n \mathbf{x}_n \end{bmatrix} \\ &= S \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} = S\Lambda. \end{aligned}$$

Note that Λ is a diagonal matrix whose non-zero entries are the eigenvalues of A . Because the columns of S are independent, S^{-1} exists and we can multiply both sides of $AS = S\Lambda$ by S^{-1} :

$$S^{-1}AS = \Lambda.$$

Equivalently, $A = S\Lambda S^{-1}$.

Powers of A

What are the eigenvalues and eigenvectors of A^2 ?

$$\begin{aligned} \text{If } A\mathbf{x} &= \lambda\mathbf{x}, \\ \text{then } A^2\mathbf{x} &= \lambda A\mathbf{x} = \lambda^2\mathbf{x}. \end{aligned}$$

The eigenvalues of A^2 are the squares of the eigenvalues of A . The eigenvectors of A^2 are the same as the eigenvectors of A . If we write $A = S\Lambda S^{-1}$ then:

$$A^2 = S\Lambda S^{-1}S\Lambda S^{-1} = S\Lambda^2 S^{-1}.$$

Similarly, $A^k = S\Lambda^k S^{-1}$ tells us that raising the eigenvalues of A to the k th power gives us the eigenvalues of A^k , and that the eigenvectors of A^k are the same as those of A .

Theorem: If A has n independent eigenvectors with eigenvalues λ_i , then $A^k \rightarrow 0$ as $k \rightarrow \infty$ if and only if all $|\lambda_i| < 1$.

A is guaranteed to have n independent eigenvectors (and be *diagonalizable*) if all its eigenvalues are different. Most matrices do have distinct eigenvalues.

Repeated eigenvalues

If A has repeated eigenvalues, it may or may not have n independent eigenvectors. For example, the eigenvalues of the identity matrix are all 1, but that matrix still has n independent eigenvectors.

If A is the triangular matrix $\begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$ its eigenvalues are 2 and 2. Its eigenvectors are in the nullspace of $A - \lambda I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ which is spanned by $\mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. This particular A does not have two independent eigenvectors.

Difference equations $\mathbf{u}_{k+1} = A\mathbf{u}_k$

Start with a given vector \mathbf{u}_0 . We can create a sequence of vectors in which each new vector is A times the previous vector: $\mathbf{u}_{k+1} = A\mathbf{u}_k$. $\mathbf{u}_{k+1} = A\mathbf{u}_k$ is a *first order difference equation*, and $\mathbf{u}_k = A^k\mathbf{u}_0$ is a solution to this system.

We get a more satisfying solution if we write \mathbf{u}_0 as a combination of eigenvectors of A :

$$\mathbf{u}_0 = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_n\mathbf{x}_n = S\mathbf{c}.$$

Then:

$$A\mathbf{u}_0 = c_1\lambda_1\mathbf{x}_1 + c_2\lambda_2\mathbf{x}_2 + \cdots + c_n\lambda_n\mathbf{x}_n$$

and:

$$\mathbf{u}_k = A^k\mathbf{u}_0 = c_1\lambda_1^k\mathbf{x}_1 + c_2\lambda_2^k\mathbf{x}_2 + \cdots + c_n\lambda_n^k\mathbf{x}_n = \Lambda^k S\mathbf{c}.$$

Fibonacci sequence

The Fibonacci sequence is 0, 1, 1, 2, 3, 5, 8, 13, In general, $F_{k+2} = F_{k+1} + F_k$. If we could understand this in terms of matrices, the eigenvalues of the matrices would tell us how fast the numbers in the sequence are increasing.

$\mathbf{u}_{k+1} = A\mathbf{u}_k$ was a first order system. $F_{k+2} = F_{k+1} + F_k$ is a second order scalar equation, but we can convert it to first order linear system by using a clever trick. If $\mathbf{u}_k = \begin{bmatrix} F_{k+1} \\ F_k \end{bmatrix}$, then:

$$F_{k+2} = F_{k+1} + F_k \quad (1)$$

$$F_{k+1} = F_{k+1} \quad (2)$$

is equivalent to the first order system $\mathbf{u}_{k+1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{u}_k$.

What are the eigenvalues and eigenvectors of $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$? Because A is symmetric, its eigenvalues will be real and its eigenvectors will be orthogonal.

Because A is a two by two matrix we know its eigenvalues sum to 1 (the trace) and their product is -1 (the determinant).

$$|A - \lambda I| = \begin{vmatrix} 1 - \lambda & 1 \\ 1 & -\lambda \end{vmatrix} = \lambda^2 - \lambda - 1$$

Setting this to zero we find $\lambda = \frac{1 \pm \sqrt{1+4}}{2}$; i.e. $\lambda_1 = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$ and $\lambda_2 = \frac{1}{2}(1 - \sqrt{5}) \approx -.618$. The growth rate of the F_k is controlled by λ_1 , the only eigenvalue with absolute value greater than 1. This tells us that for large k , $F_k \approx c_1 \left(\frac{1+\sqrt{5}}{2}\right)^k$ for some constant c_1 . (Remember $\mathbf{u}_k = A^k \mathbf{u}_0 = c_1 \lambda_1^k \mathbf{x}_1 + c_2 \lambda_2^k \mathbf{x}_2$, and here λ_2^k goes to zero since $|\lambda_2| < 1$.)

To find the eigenvectors of A note that:

$$(A - \lambda I)\mathbf{x} = \begin{bmatrix} 1 - \lambda & 1 \\ 1 & -\lambda \end{bmatrix} \mathbf{x}$$

equals $\mathbf{0}$ when $\mathbf{x} = \begin{bmatrix} \lambda \\ 1 \end{bmatrix}$, so $\mathbf{x}_1 = \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix}$ and $\mathbf{x}_2 = \begin{bmatrix} \lambda_2 \\ 1 \end{bmatrix}$.

Finally, $\mathbf{u}_0 = \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = c_1 \mathbf{x}_1 + c_2 \mathbf{x}_2$ tells us that $c_1 = -c_2 = \frac{1}{\sqrt{5}}$.

Because $\begin{bmatrix} F_{k+1} \\ F_k \end{bmatrix} = \mathbf{u}_k = c_1 \lambda_1^k \mathbf{x}_1 + c_2 \lambda_2^k \mathbf{x}_2$, we get:

$$F_k = \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^k - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^k.$$

Using eigenvalues and eigenvectors, we have found a *closed form expression* for the Fibonacci numbers.

Summary: When a sequence evolves over time according to the rules of a first order system, the eigenvalues of the matrix of that system determine the long term behavior of the series. To get an exact formula for the series we find the eigenvectors of the matrix and then solve for the coefficients c_1, c_2, \dots

MIT OpenCourseWare
<http://ocw.mit.edu>

18.06SC Linear Algebra
Fall 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.

Orthogonal matrices and Gram-Schmidt

In this lecture we finish introducing orthogonality. Using an orthonormal basis or a matrix with orthonormal columns makes calculations much easier. The Gram-Schmidt process starts with any basis and produces an orthonormal basis that spans the same space as the original basis.

Orthonormal vectors

The vectors $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ are *orthonormal* if:

$$\mathbf{q}_i^T \mathbf{q}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases}$$

In other words, they all have (normal) length 1 and are perpendicular (ortho) to each other. Orthonormal vectors are always independent.

Orthonormal matrix

If the columns of $Q = [\mathbf{q}_1 \dots \mathbf{q}_n]$ are orthonormal, then $Q^T Q = I$ is the identity.

Matrices with orthonormal columns are a new class of important matrices to add to those on our list: triangular, diagonal, permutation, symmetric, reduced row echelon, and projection matrices. We'll call them "orthonormal matrices".

A square orthonormal matrix Q is called an *orthogonal matrix*. If Q is square, then $Q^T Q = I$ tells us that $Q^T = Q^{-1}$.

For example, if $Q = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ then $Q^T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$. Both Q and Q^T

are orthogonal matrices, and their product is the identity.

The matrix $Q = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$ is orthogonal. The matrix $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ is

not, but we can adjust that matrix to get the orthogonal matrix $Q = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

We can use the same tactic to find some larger orthogonal matrices called *Hadamard matrices*:

$$Q = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

An example of a rectangular matrix with orthonormal columns is:

$$Q = \frac{1}{3} \begin{bmatrix} 1 & -2 \\ 2 & -1 \\ 2 & 2 \end{bmatrix}.$$

We can extend this to a (square) orthogonal matrix:

$$\frac{1}{3} \begin{bmatrix} 1 & -2 & 2 \\ 2 & -1 & -2 \\ 2 & 2 & 1 \end{bmatrix}.$$

These examples are particularly nice because they don't include complicated square roots.

Orthonormal columns are good

Suppose Q has orthonormal columns. The matrix that projects onto the column space of Q is:

$$P = Q^T(Q^T Q)^{-1}Q^T.$$

If the columns of Q are orthonormal, then $Q^T Q = I$ and $P = QQ^T$. If Q is square, then $P = I$ because the columns of Q span the entire space.

Many equations become trivial when using a matrix with orthonormal columns. If our basis is orthonormal, the projection component \hat{x}_i is just $\mathbf{q}_i^T \mathbf{b}$ because $A^T A \hat{\mathbf{x}} = A^T \mathbf{b}$ becomes $\hat{\mathbf{x}} = Q^T \mathbf{b}$.

Gram-Schmidt

With elimination, our goal was "make the matrix triangular". Now our goal is "make the matrix orthonormal".

We start with two independent vectors \mathbf{a} and \mathbf{b} and want to find orthonormal vectors \mathbf{q}_1 and \mathbf{q}_2 that span the same plane. We start by finding orthogonal vectors \mathbf{A} and \mathbf{B} that span the same space as \mathbf{a} and \mathbf{b} . Then the unit vectors $\mathbf{q}_1 = \frac{\mathbf{A}}{\|\mathbf{A}\|}$ and $\mathbf{q}_2 = \frac{\mathbf{B}}{\|\mathbf{B}\|}$ form the desired orthonormal basis.

Let $\mathbf{A} = \mathbf{a}$. We get a vector orthogonal to \mathbf{A} in the space spanned by \mathbf{a} and \mathbf{b} by projecting \mathbf{b} onto \mathbf{a} and letting $\mathbf{B} = \mathbf{b} - \mathbf{p}$. (\mathbf{B} is what we previously called \mathbf{e} .)

$$\mathbf{B} = \mathbf{b} - \frac{\mathbf{A}^T \mathbf{b}}{\mathbf{A}^T \mathbf{A}} \mathbf{A}.$$

If we multiply both sides of this equation by \mathbf{A}^T , we see that $\mathbf{A}^T \mathbf{B} = 0$.

What if we had started with three independent vectors, \mathbf{a} , \mathbf{b} and \mathbf{c} ? Then we'd find a vector \mathbf{C} orthogonal to both \mathbf{A} and \mathbf{B} by subtracting from \mathbf{c} its components in the \mathbf{A} and \mathbf{B} directions:

$$\mathbf{C} = \mathbf{c} - \frac{\mathbf{A}^T \mathbf{c}}{\mathbf{A}^T \mathbf{A}} \mathbf{A} - \frac{\mathbf{B}^T \mathbf{c}}{\mathbf{B}^T \mathbf{B}} \mathbf{B}.$$

For example, suppose $\mathbf{a} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$. Then $\mathbf{A} = \mathbf{a}$ and:

$$\begin{aligned} \mathbf{B} &= \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - \frac{\mathbf{A}^T \mathbf{b}}{\mathbf{A}^T \mathbf{A}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - \frac{3}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}. \end{aligned}$$

Normalizing, we get:

$$Q = [\mathbf{q}_1 \quad \mathbf{q}_2] = \begin{bmatrix} 1/\sqrt{3} & 0 \\ 1/\sqrt{3} & -1/\sqrt{2} \\ 1/\sqrt{3} & 1/\sqrt{2} \end{bmatrix}.$$

The column space of Q is the plane spanned by \mathbf{a} and \mathbf{b} .

When we studied elimination, we wrote the process in terms of matrices and found $A = LU$. A similar equation $A = QR$ relates our starting matrix A to the result Q of the Gram-Schmidt process. Where L was lower triangular, R is upper triangular.

Suppose $A = [\mathbf{a}_1 \quad \mathbf{a}_2]$. Then:

$$\begin{matrix} A \\ [\mathbf{a}_1 \quad \mathbf{a}_2] \end{matrix} = \begin{matrix} Q \\ [\mathbf{q}_1 \quad \mathbf{q}_2] \end{matrix} \begin{matrix} R \\ \begin{bmatrix} \mathbf{a}_1^T \mathbf{q}_1 & \mathbf{a}_2^T \mathbf{q}_1 \\ \mathbf{a}_1^T \mathbf{q}_2 & \mathbf{a}_2^T \mathbf{q}_2 \end{bmatrix} \end{matrix}.$$

If R is upper triangular, then it should be true that $\mathbf{a}_1^T \mathbf{q}_2 = 0$. This must be true because we chose \mathbf{q}_1 to be a unit vector in the direction of \mathbf{a}_1 . All the later \mathbf{q}_i were chosen to be perpendicular to the earlier ones.

Notice that $R = Q^T A$. This makes sense; $Q^T Q = I$.

MIT OpenCourseWare
<http://ocw.mit.edu>

18.06SC Linear Algebra
Fall 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.

Singular value decomposition

The *singular value decomposition* of a matrix is usually referred to as the *SVD*. This is the final and best factorization of a matrix:

$$A = U\Sigma V^T$$

where U is orthogonal, Σ is diagonal, and V is orthogonal.

In the decomposition $A = U\Sigma V^T$, A can be *any* matrix. We know that if A is symmetric positive definite its eigenvectors are orthogonal and we can write $A = Q\Lambda Q^T$. This is a special case of a SVD, with $U = V = Q$. For more general A , the SVD requires two different matrices U and V .

We've also learned how to write $A = S\Lambda S^{-1}$, where S is the matrix of n distinct eigenvectors of A . However, S may not be orthogonal; the matrices U and V in the SVD will be.

How it works

We can think of A as a linear transformation taking a vector \mathbf{v}_1 in its row space to a vector $\mathbf{u}_1 = A\mathbf{v}_1$ in its column space. The SVD arises from finding an orthogonal basis for the row space that gets transformed into an orthogonal basis for the column space: $A\mathbf{v}_i = \sigma_i\mathbf{u}_i$.

It's not hard to find an orthogonal basis for the row space – the Gram-Schmidt process gives us one right away. But in general, there's no reason to expect A to transform that basis to another orthogonal basis.

You may be wondering about the vectors in the nullspaces of A and A^T . These are no problem – zeros on the diagonal of Σ will take care of them.

Matrix language

The heart of the problem is to find an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r$ for the row space of A for which

$$\begin{aligned} A \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_r \end{bmatrix} &= \begin{bmatrix} \sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & \cdots & \sigma_r\mathbf{u}_r \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix}, \end{aligned}$$

with $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$ an orthonormal basis for the column space of A . Once we add in the nullspaces, this equation will become $AV = U\Sigma$. (We can complete the orthonormal bases $\mathbf{v}_1, \dots, \mathbf{v}_r$ and $\mathbf{u}_1, \dots, \mathbf{u}_r$ to orthonormal bases for the entire space any way we want. Since $\mathbf{v}_{r+1}, \dots, \mathbf{v}_n$ will be in the nullspace of A , the diagonal entries $\sigma_{r+1}, \dots, \sigma_n$ will be 0.)

The columns of U and V are bases for the row and column spaces, respectively. Usually $U \neq V$, but if A is positive definite we can use the *same* basis for its row and column space!

Calculation

Suppose A is the invertible matrix $\begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix}$. We want to find vectors \mathbf{v}_1 and \mathbf{v}_2 in the row space \mathbb{R}^2 , \mathbf{u}_1 and \mathbf{u}_2 in the column space \mathbb{R}^2 , and positive numbers σ_1 and σ_2 so that the \mathbf{v}_i are orthonormal, the \mathbf{u}_i are orthonormal, and the σ_i are the scaling factors for which $A\mathbf{v}_i = \sigma_i\mathbf{u}_i$.

This is a big step toward finding orthonormal matrices V and U and a diagonal matrix Σ for which:

$$AV = U\Sigma.$$

Since V is orthogonal, we can multiply both sides by $V^{-1} = V^T$ to get:

$$A = U\Sigma V^T.$$

Rather than solving for U , V and Σ simultaneously, we multiply both sides by $A^T = V\Sigma^T U^T$ to get:

$$\begin{aligned} A^T A &= V\Sigma U^{-1} U\Sigma V^T \\ &= V\Sigma^2 V^T \\ &= V \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{bmatrix} V^T. \end{aligned}$$

This is in the form $Q\Lambda Q^T$; we can now find V by diagonalizing the symmetric positive definite (or semidefinite) matrix $A^T A$. The columns of V are eigenvectors of $A^T A$ and the eigenvalues of $A^T A$ are the values σ_i^2 . (We choose σ_i to be the positive square root of λ_i .)

To find U , we do the same thing with AA^T .

SVD example

We return to our matrix $A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix}$. We start by computing

$$\begin{aligned} A^T A &= \begin{bmatrix} 4 & -3 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 25 & 7 \\ 7 & 25 \end{bmatrix}. \end{aligned}$$

The eigenvectors of this matrix will give us the vectors \mathbf{v}_i , and the eigenvalues will give us the numbers σ_i .

Two orthogonal eigenvectors of $A^T A$ are $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. To get an orthonormal basis, let $\mathbf{v}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$. These have eigenvalues $\sigma_1^2 = 32$ and $\sigma_2^2 = 18$. We now have:

$$\begin{array}{c} A \\ \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} \end{array} = \begin{array}{c} U \\ \begin{bmatrix} \\ \end{bmatrix} \end{array} \begin{array}{c} \Sigma \\ \begin{bmatrix} 4\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix} \end{array} \begin{array}{c} V^T \\ \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \end{array}.$$

We could solve this for U , but for practice we'll find U by finding orthonormal eigenvectors \mathbf{u}_1 and \mathbf{u}_2 for $AA^T = U\Sigma^2U^T$.

$$\begin{aligned} AA^T &= \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} \begin{bmatrix} 4 & -3 \\ 4 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 32 & 0 \\ 0 & 18 \end{bmatrix}. \end{aligned}$$

Luckily, AA^T happens to be diagonal. It's tempting to let $\mathbf{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{u}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, as Professor Strang did in the lecture, but because $A\mathbf{v}_2 = \begin{bmatrix} 0 \\ -3\sqrt{2} \end{bmatrix}$ we instead have $\mathbf{u}_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ and $U = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Note that this also gives us a chance to double check our calculation of σ_1 and σ_2 .

Thus, the SVD of A is:

$$\begin{array}{c} A \\ \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} \end{array} = \begin{array}{c} U \\ \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{array} \begin{array}{c} \Sigma \\ \begin{bmatrix} 4\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix} \end{array} \begin{array}{c} V^T \\ \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \end{array}.$$

Example with a nullspace

Now let $A = \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix}$. This has a one dimensional nullspace and one dimensional row and column spaces.

The row space of A consists of the multiples of $\begin{bmatrix} 4 \\ 3 \end{bmatrix}$. The column space of A is made up of multiples of $\begin{bmatrix} 4 \\ 8 \end{bmatrix}$. The nullspace and left nullspace are perpendicular to the row and column spaces, respectively.

Unit basis vectors of the row and column spaces are $\mathbf{v}_1 = \begin{bmatrix} .8 \\ .6 \end{bmatrix}$ and $\mathbf{u}_1 =$

$\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$. To compute σ_1 we find the nonzero eigenvalue of $A^T A$.

$$\begin{aligned} A^T A &= \begin{bmatrix} 4 & 8 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix} \\ &= \begin{bmatrix} 80 & 60 \\ 60 & 45 \end{bmatrix}. \end{aligned}$$

Because this is a rank 1 matrix, one eigenvalue must be 0. The other must equal the trace, so $\sigma_1^2 = 125$. After finding unit vectors perpendicular to \mathbf{u}_1 and \mathbf{v}_1 (basis vectors for the left nullspace and nullspace, respectively) we see that the SVD of A is:

$$\begin{array}{ccc} \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix} & = & \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} \sqrt{125} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} .8 & .6 \\ .6 & -.8 \end{bmatrix} \\ A & & U \qquad \qquad \Sigma \qquad \qquad V^T \end{array}$$

The singular value decomposition combines topics in linear algebra ranging from positive definite matrices to the four fundamental subspaces.

- $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r$ is an orthonormal basis for the row space.
- $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$ is an orthonormal basis for the column space.
- $\mathbf{v}_{r+1}, \dots, \mathbf{v}_n$ is an orthonormal basis for the nullspace.
- $\mathbf{u}_{r+1}, \dots, \mathbf{u}_m$ is an orthonormal basis for the left nullspace.

These are the “right” bases to use, because $A\mathbf{v}_i = \sigma_i\mathbf{u}_i$.

MIT OpenCourseWare
<http://ocw.mit.edu>

18.06SC Linear Algebra
Fall 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.

Symmetric matrices and positive definiteness

Symmetric matrices are good – their eigenvalues are real and each has a complete set of orthonormal eigenvectors. Positive definite matrices are even better.

Symmetric matrices

A *symmetric matrix* is one for which $A = A^T$. If a matrix has some special property (e.g. it's a Markov matrix), its eigenvalues and eigenvectors are likely to have special properties as well. For a symmetric matrix with real number entries, the eigenvalues are real numbers and it's possible to choose a complete set of eigenvectors that are perpendicular (or even orthonormal).

If A has n independent eigenvectors we can write $A = SAS^{-1}$. If A is symmetric we can write $A = Q\Lambda Q^{-1} = Q\Lambda Q^T$, where Q is an orthogonal matrix. Mathematicians call this the *spectral theorem* and think of the eigenvalues as the “spectrum” of the matrix. In mechanics it's called the *principal axis theorem*.

In addition, any matrix of the form $Q\Lambda Q^T$ will be symmetric.

Real eigenvalues

Why are the eigenvalues of a symmetric matrix real? Suppose A is symmetric and $A\mathbf{x} = \lambda\mathbf{x}$. Then we can conjugate to get $\overline{A\mathbf{x}} = \overline{\lambda\mathbf{x}}$. If the entries of A are real, this becomes $A\overline{\mathbf{x}} = \overline{\lambda}\overline{\mathbf{x}}$. (This proves that complex eigenvalues of real valued matrices come in conjugate pairs.)

Now transpose to get $\overline{\mathbf{x}}^T A^T = \overline{\mathbf{x}}^T \overline{\lambda}$. Because A is symmetric we now have $\overline{\mathbf{x}}^T A = \overline{\mathbf{x}}^T \overline{\lambda}$. Multiplying both sides of this equation on the right by \mathbf{x} gives:

$$\overline{\mathbf{x}}^T A\mathbf{x} = \overline{\mathbf{x}}^T \overline{\lambda}\mathbf{x}.$$

On the other hand, we can multiply $A\mathbf{x} = \lambda\mathbf{x}$ on the left by $\overline{\mathbf{x}}^T$ to get:

$$\overline{\mathbf{x}}^T A\mathbf{x} = \overline{\mathbf{x}}^T \lambda\mathbf{x}.$$

Comparing the two equations we see that $\overline{\mathbf{x}}^T \overline{\lambda}\mathbf{x} = \overline{\mathbf{x}}^T \lambda\mathbf{x}$ and, unless $\overline{\mathbf{x}}^T \mathbf{x}$ is zero, we can conclude $\lambda = \overline{\lambda}$ is real.

How do we know $\overline{\mathbf{x}}^T \mathbf{x} \neq 0$?

$$\overline{\mathbf{x}}^T \mathbf{x} = \begin{bmatrix} \overline{x}_1 & \overline{x}_2 & \cdots & \overline{x}_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = |x_1|^2 + |x_2|^2 + \cdots + |x_n|^2.$$

If $\mathbf{x} \neq \mathbf{0}$ then $\overline{\mathbf{x}}^T \mathbf{x} \neq 0$.

With complex vectors, as with complex numbers, multiplying by the conjugate is often helpful.

Symmetric matrices with real entries have $A = A^T$, real eigenvalues, and perpendicular eigenvectors. If A has complex entries, then it will have real eigenvalues and perpendicular eigenvectors if and only if $A = \overline{A}^T$. (The proof of this follows the same pattern.)

Projection onto eigenvectors

If $A = A^T$, we can write:

$$\begin{aligned} A &= Q\Lambda Q^T \\ &= \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_n^T \end{bmatrix} \\ &= \lambda_1 \mathbf{q}_1 \mathbf{q}_1^T + \lambda_2 \mathbf{q}_2 \mathbf{q}_2^T + \cdots + \lambda_n \mathbf{q}_n \mathbf{q}_n^T \end{aligned}$$

The matrix $\mathbf{q}_k \mathbf{q}_k^T$ is the projection matrix onto \mathbf{q}_k , so every symmetric matrix is a combination of perpendicular projection matrices.

Information about eigenvalues

If we know that eigenvalues are real, we can ask whether they are positive or negative. (Remember that the signs of the eigenvalues are important in solving systems of differential equations.)

For very large matrices A , it's impractical to compute eigenvalues by solving $|A - \lambda I| = 0$. However, it's not hard to compute the pivots, and the signs of the pivots of a symmetric matrix are the same as the signs of the eigenvalues:

$$\text{number of positive pivots} = \text{number of positive eigenvalues.}$$

Because the eigenvalues of $A + bI$ are just b more than the eigenvalues of A , we can use this fact to find which eigenvalues of a symmetric matrix are greater or less than any real number b . This tells us a lot about the eigenvalues of A even if we can't compute them directly.

Positive definite matrices

A *positive definite matrix* is a symmetric matrix A for which all eigenvalues are positive. A good way to tell if a matrix is positive definite is to check that all its pivots are positive.

Let $A = \begin{bmatrix} 5 & 2 \\ 2 & 3 \end{bmatrix}$. The pivots of this matrix are 5 and $(\det A)/5 = 11/5$. The matrix is symmetric and its pivots (and therefore eigenvalues) are positive, so A is a positive definite matrix. Its eigenvalues are the solutions to:

$$|A - \lambda I| = \lambda^2 - 8\lambda + 11 = 0,$$

i.e. $4 \pm \sqrt{5}$.

The determinant of a positive definite matrix is always positive but the determinant of $\begin{bmatrix} -1 & 0 \\ 0 & -3 \end{bmatrix}$ is also positive, and that matrix isn't positive definite. If all of the subdeterminants of A are positive (determinants of the k by k matrices in the upper left corner of A , where $1 \leq k \leq n$), then A is positive definite.

The subject of positive definite matrices brings together what we've learned about pivots, determinants and eigenvalues of square matrices. Soon we'll have a chance to bring together what we've learned in this course and apply it to non-square matrices.

MIT OpenCourseWare
<http://ocw.mit.edu>

18.06SC Linear Algebra
Fall 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.