Probability Review

Jeremy Irvin and
Daniel Spokoyny

Created from Maleki and Do's Probability Review for Stanford CS229

Consider flipping a coin twice.

Elements of Probability

- Sample Space (Ω): Set of all outcomes
- $\Omega = \{HH, HT, TH, TT\}$

Elements of Probability

- Event (E): A subset E of Ω, ie, a subset of outcomes

- E = {HH, HT}

Elements of Probability

- Event Space (F): Set of all possible events, ie, set of all subsets of Ω

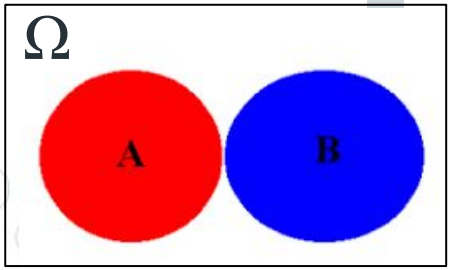
- F= {∅, {HH}, {TT}, {TH}, {TT}, {HH, HT} , …}

Elements of Probability

- Probability Measure (P): A function $P : \mathcal{F} \to \mathbb{R}$ satisfying:
  - (i) $P(A) \geq 0$, for all $A \in \mathcal{F}$
  - (ii) $P(\Omega) = 1$

  - (iii) If $A_1, A_2, \ldots$ are disjoint events ($A_i \cap A_j = \emptyset$ when i≠j), then

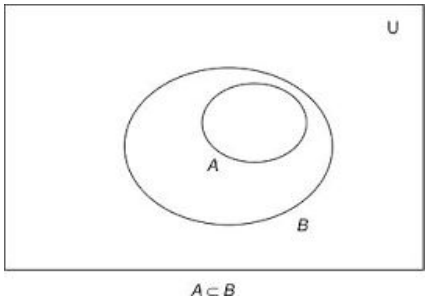
$$P(\cup_i A_i) = \sum_i P(A_i)$$

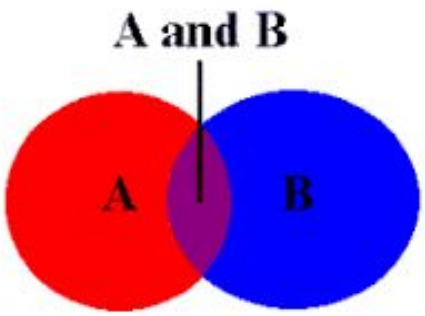


Simple Example

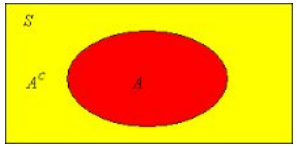
- P({HH}) = ¼ , P({HT}) = ¼, P({HH, HT}) = ½
- Notice {HH} and {HT} are disjoint events, and
  - P({HH, HT}) = P({HH}) + P({HT})

Properties





- If $A \subseteq B \Longrightarrow P(A) \leq P(B)$.
- $P(A \cap B) \leq \min(P(A), P(B))$.
- (Union Bound) $P(A \cup B) \leq P(A) + P(B)$.
- $P(\Omega \setminus A) = 1 - P(A)$.

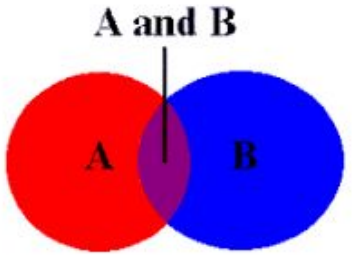


- If $A_1, \ldots, A_k$ are disjoint events with

$$\cup_{i=1}^{k} A_i = \Omega,$$

then $\sum_{i=1}^{k} P(A_k) = 1.$

Conditional Probability



- If B is an event with non-zero probability ($P(B) \neq 0$) then the conditional probability of A given B is

$$P(A|B) \triangleq \frac{P(A \cap B)}{P(B)}$$

	23	2	25
	12	3	15
	35	5	40

- In other words, $P(A|B)$ is the probability of event A after observing event B.

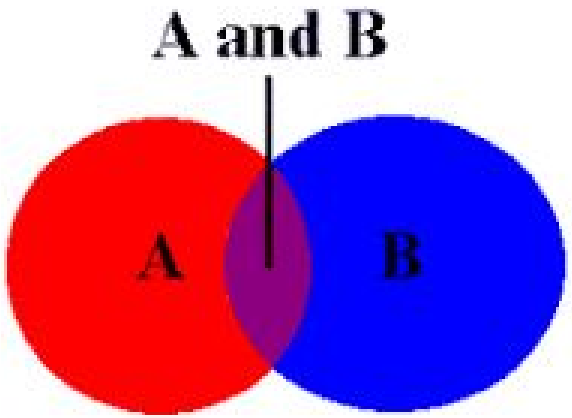
Independence

- A and B are independent if

$$P(A \cap B) = P(A)P(B)$$

or equivalently,

$$P(A \mid B) = P(A)$$



HH	HT
TH	TT

Bayes Theorem!!!

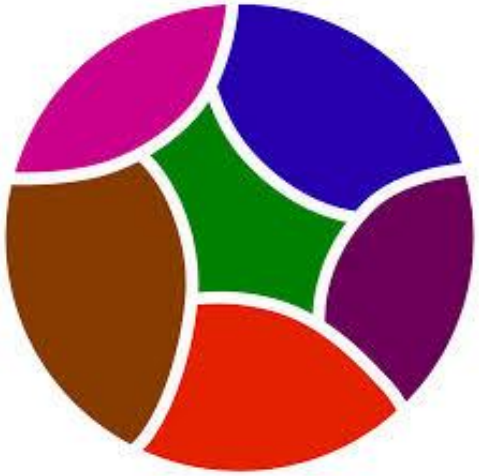
- If A and B are any two events, then

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

- If {*A_j*} is a partition of the sample space, then

$$P(B) = \sum_j P(B \mid A_j) P(A_j),$$

$$\Rightarrow P(A_i \mid B) = \frac{P(B \mid A_i)\,P(A_i)}{\sum_j P(B \mid A_j)\,P(A_j)}.$$



Random Variables

- Suppose we flip 10 coins and want to know the number of coins which come up heads.

- Maybe we get the sequence:

{HHTHTTTHTH}

Random Variables

- Real-valued functions of outcomes (such as the number of heads that appear among our 10 tosses) are known as random variables.

- More formally, a random variable X is a function $X : \Omega \longrightarrow \mathbb{R}$.

Random Variable Example

- Suppose we are flipping a coin 10 times.
- For any outcome $w \in \Omega$, let X(w) be the number of heads which occur in w.
- X is discrete since it can only take on a countable amount of values {0,1…,10} (a random variable is continuous if it takes on an uncountable number of values) and

$$P(X = k) = P(\{w:X(w) = k\}) = 10Ck/2^{10}$$

Probability Mass Function (pmf)

- A probability mass function (pmf) corresponding to a discrete random variable X is a function $p_X : Val(X) \to [0, 1]$ where

$$p_X(x) := P(X = x) = P(\{\omega \in \Omega : X(\omega) = x\})$$

- $$\sum_{x \in Val(X)} p_X(x) = 1$$

$$A \subseteq Val(X)$$

- $$\sum_{x \in A} p_X(x) = P(X \in A) = P(\{\omega \in \Omega : X(\omega) \in A\})$$

Cumulative Distribution Function (cdf)

- A cumulative distribution function (cdf) corresponding to a random variable X is a function $F_X : \mathbb{R} \to [0, 1]$ which specifies a probability measure as

$$F_X(x) \triangleq P(X \leq x).$$

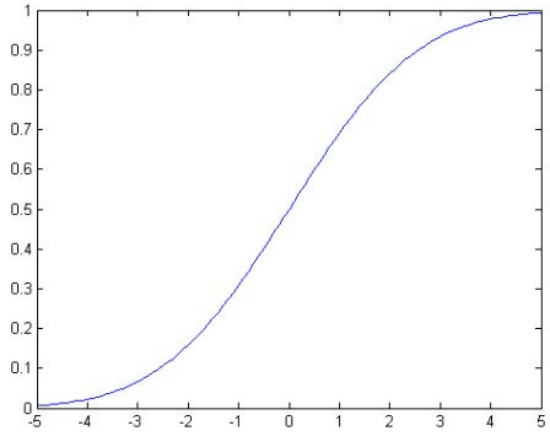


Figure 1: A cumulative distribution function (CDF).

cdf Properties

- $0 \leq F_X(x) \leq 1.$
- $\lim_{x \to -\infty} F_X(x) = 0.$
- $\lim_{x \to \infty} F_X(x) = 1.$
- $x \leq y \Longrightarrow F_X(x) \leq F_X(y).$

Probability Density Function (pdf)

- A probability density function (pdf) corresponding to a continuous random variable X with differentiable cdf F_X is a function $f_X : \Omega \to \mathbb{R}$ where

$$f_X(x) \triangleq \frac{dF_X(x)}{dx}.$$

- $f_X(x) \geq 0$.
- $\int_{-\infty}^{\infty} f_X(x) = 1.$
- $\int_{x \in A} f_X(x) dx = P(X \in A).$

Expected Value

- If X is a random variable with pmf $p_X(x)$ the expected value of *X* is defined as

$$\mathbb{E}[X] := \sum_x xP(X = x)$$

- Think of $\mathbb{E}[X]$ as a weighted average of the values *x* that X can take on with weights $p_X(x)$.

Expected Value

- E[X] is called the mean of X
- E[a] = a for any constant $a \in \mathbb{R}$
- E[X + Y] = E[X] + E[Y] (linearity)

Variance

- The variance of a random variable X is a measure of how concentrated the distribution of X is around its mean $E[X]$

- Formally, the variance of X is defined

$$Var[X] \triangleq E[(X - E(X))^2] = E[X^2] - E[X]^2$$

- $Var[a] = 0$ for any constant $a \in \mathbb{R}$.

- $Var(aX)=a^2Var(X)$

Common Discrete Distributions

- X~Bernoulli(p)

$$p(x) = \begin{cases} p & x = 1 \\ 1 - p & x = 0 \end{cases}$$

A single coin flip, with heads probability p.

- X~Binomial(n,p)

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

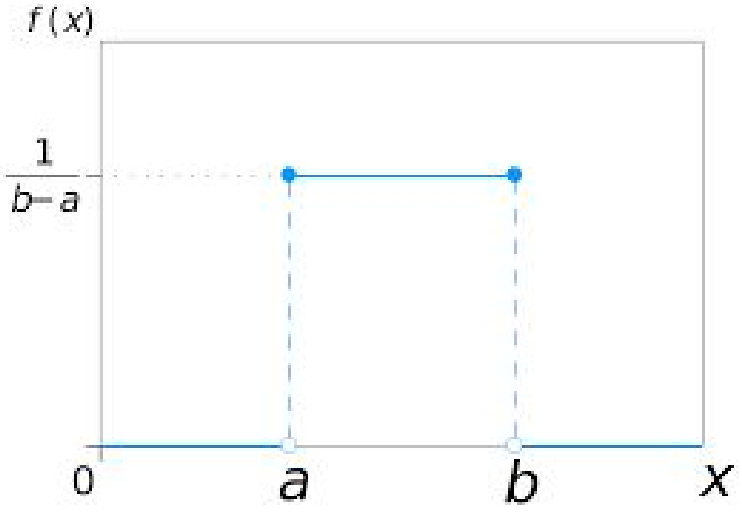
The number of heads in n independent flips of a coin with heads probability p.

Common Continuous Distributions

- X~Uniform(a,b)

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

Any equal-sized interval occurs with equal probability:

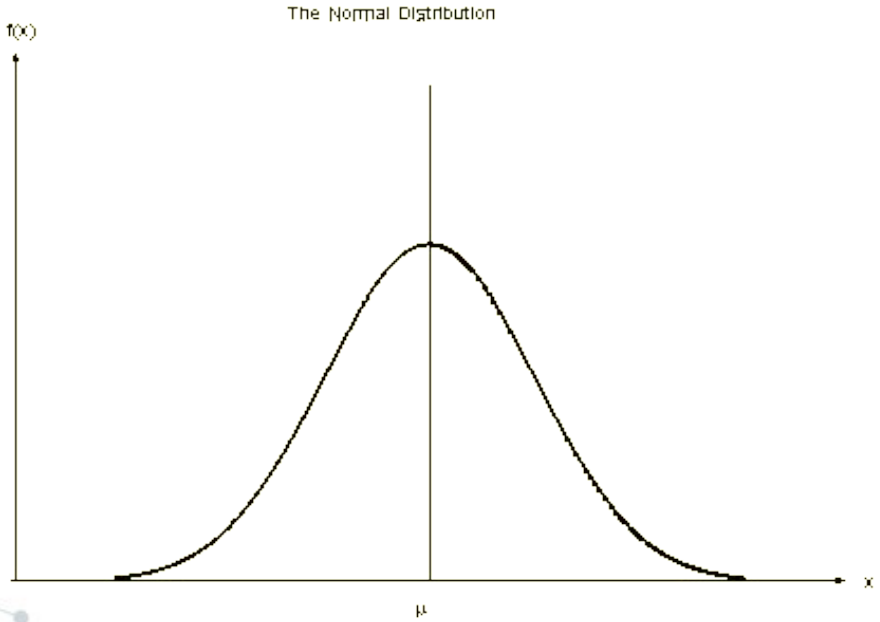


Common Continuous Distributions

- $X \sim \text{Normal}(\mu, \sigma^2)$

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

Also known as Gaussian, the typical 'bell-curve':



Expected Value and Variance Example

- Calculate the mean and the variance of the uniform random variable X with pdf $f_X(x) = 1$ for $x \in [0, 1]$ and $f_X(x) = 0$ elsewhere.
- Answer:

  E[X]=½ , Var(X) = 1/12

What Just Happened?

- Axioms of Probability
- Bayes Theorem
- Random Variables
- Common Distributions

Python demo

Gaussian Distribution Sampling

Done with probability!

Linear Algebra Review

Jeremy Irvin and
Daniel Spokoyny

Created from Kolter's Linear Algebra Review for Stanford CS229

Linear Equations, Notation, Matrix Multiplication

See blackboard

Identity/ Diagonal Matrix, Transpose, Symmetric Matrices, Trace, Norms

See blackboard

Linear Independence

- A set of vectors $\{x_1, x_2, ..., x_n\} \subset \mathbb{R}^m$ is linearly independent if no vector can be written as a linear combination of the other vectors.

Linear Independence

- Conversely, the set is linearly dependent if *one vector can* be written as a linear combination of the remaining vectors, ie. if

$$x_n = \sum_{i=1}^{n-1} \alpha_i x_i$$

for some scalar values $\alpha_1, \ldots, \alpha_{n-1} \in \mathbb{R}$.

- Example on board

Rank

- Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$.
  - The column rank of A is the number of linearly independent columns of A.
  - The row rank of a A is the number of linearly independent rows of A.
  - The row rank of A equals the column rank of A, and is called the rank of A.

Properties of Rank

- Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$.
  - $\text{rank}(A) \leq \min(m,n)$
  - If $\text{rank}(A) = \min(m,n)$, A has <u>full rank.</u>
  - $\text{rank}(A) = \text{rank}(A^T)$
  - $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$
  - $\text{rank}(A+B) \leq \text{rank}(A) + \text{rank}(B)$

Inverse

- The inverse of a square matrix $A \in \mathbb{R}^{n \times n}$ is denoted by A^{-1} and is the unique matrix such that

$$A^{-1}A = I = AA^{-1}$$

- Not all matrices have inverses.

- A is invertible (or non-singular) if A^{-1} exists and is called non-invertible or singular otherwise.

Inverse

- *A* is invertible iff it is square and has full rank.

- $(A^{-1})^{-1} = A$
- $(AB)^{-1} = B^{-1}A^{-1}$
- $(A^{-1})^T = (A^T)^{-1}$

- *A* is non-singular iff $Ax=b$ has a unique solution for any b ($x=A^{-1}b$).

Orthogonality and Normalization

- Recall two vectors $x, y \in \mathbb{R}^n$ are orthogonal if

$$x^T y = 0$$

- A vector $x \in \mathbb{R}^n$ is normalized if

$$\|x\|_2 = 1$$

Orthogonal Matrices

- A square matrix $U \in \mathbb{R}^{n \times n}$ is orthogonal if all of its columns are orthogonal to each other and are normalized, or equivalently,

$$U^T U = I = U U^T$$

- Orthogonal matrices are norm (or distance) preserving, ie,

$$\|Ux\|_2 = \|x\|_2$$

for any $x \in \mathbb{R}^n$, $U \in \mathbb{R}^{n \times n}$.

Span and Basis

- The span of a set of vectors $\{x_1, x_2, \ldots x_n\} \subset \mathbb{R}^m$ is the set of all vectors that can be expressed as a linear combination of $\{x_1, \ldots, x_n\}$, ie,

$$\text{span}(\{x_1, \ldots x_n\}) = \left\{ v : v = \sum_{i=1}^{n} \alpha_i x_i, \quad \alpha_i \in \mathbb{R} \right\}$$

- If $\{x_1, \ldots, x_n\}$ is a set of n linear independent vectors then .

$$\text{span}(\{x_1, \ldots x_n\}) = \mathbb{R}^n$$

- $\{x_1, \ldots, x_n\}$ is called a basis of $\mathbb{R}^n$.

Range and Nullspace

- The range of $A \in \mathbb{R}^{m \times n}$ is the span of the columns of A, ie,

$$\mathcal{R}(A) = \{v \in \mathbb{R}^m : v = Ax, x \in \mathbb{R}^n\}$$

- The nullspace of a matrix $A \in \mathbb{R}^{m \times n}$ is the set of all vectors that equal 0 when multiplied by A, ie,

$$\mathcal{N}(A) = \{x \in \mathbb{R}^n : Ax = 0\}$$

Range and Nullspace

- The range of A^T and the nullspace of A are orthogonal complements in $\mathbb{R}^n$.

- This means:
  - they are disjoint subsets which span the ent
    space, and
  - every vector in the range of A^T is orthogona
    every vector in the nullspace of A.

Projection

- The projection of a vector $y \in \mathbb{R}^m$ onto the span of $\{x_1, \ldots, x_n\}$ (with $x_i \in \mathbb{R}^m$) is the vector $v \in \text{span}(\{x_1, \ldots x_n\})$ closest to y as measured by Euclidean Distance (ie $||v - y||_2$).

- Formally,

$$\text{Proj}(y; \{x_1, ..., x_n\}) = \underset{v \in \text{span}(\{x_1, ..., x_n\})}{\text{argmin}} ||y - v||_2$$

Projection

- Assuming A is full rank and $n<m$, the projection of $y \in \mathbb{R}^m$ onto the range of A is

$$\text{Proj}(y; A) = \underset{v \in \mathcal{R}(A)}{\text{argmin}} ||y - v||_2 = A(A^T A)^{-1} A^T y$$

- When A contains a single column, ie, $A = a \in \mathbb{R}^m$, we see the 'familiar' case of a projection of a vector onto a line:

$$\text{Proj}(y; a) = \frac{aa^T}{a^T a} y$$

Determinant

- The determinant of a square matrix is a function $\det : \mathbb{R}^{n \times n} \to \mathbb{R}$, denoted $|A|$ or $\det A$.

- Let $a_1, \ldots, a_n \in \mathbb{R}^n$ denote the columns of A. Consider S the the restriction of $\operatorname{span}(\{a_1, \ldots, a_n\})$ to linear combinations whose coefficients satisfy

$$0 \leq \alpha_i \leq 1, \, i = 1, \ldots, n.$$

Determinant

- The absolute value of the determinant of A is a measure of the 'volume' of S.

- For example, for 2 x 2 matrices A, $|\det A|$ is a measure of the area enclosed by the parallelogram with edges the columns of A.

Properties of Determinant

- $|\mathrm{I}| = 1$.

- If B is formed by multiplying a single row of A by a real number t, then

$$|B| = \mathrm{t}|A|.$$

- If B is formed by switching ay two rows of A, then

$$|B| = -|A|.$$

Properties of Determinant

- $|A| = |A^{\mathrm{T}}|$.
- $|AB| = |A||B|$
- $|A| = 0$ if and only if A is singular (non-invertible).
- If A is invertible, then $|A^{-1}| = 1/|A|$.

Eigenvalues and Eigenvectors

- Given a square matrix $A \in \mathbb{R}^{n \times n}$, we say $\lambda \in \mathbb{C}$ is an eigenvalue of A and $x \in \mathbb{C}^n$ is the corresponding eigenvector if

$$Ax = \lambda x, \quad x \neq 0.$$

- Intuitively, this means multiplying A by x results in a new vector in the same direction as x but scaled by λ.

Eigenvalues and Eigenvectors

- Note that if x is an eigenvector, then cx is an eigenvector for any complex c.

- We can rewrite the equation above as

$$(\lambda I - A)x = 0, \quad x \neq 0.$$

- $(\lambda I - A)x = 0$ has a non-zero solution iff $(\lambda I - A)$ has a non-empty nullspace, which only happens if $(\lambda I - A)$ is singular, ie,

$$|(\lambda I - A)| = 0.$$

E-values and E-vectors Properties

- The trace of A is equal to the sum of its eigenvalues,

$$\operatorname{tr} A = \sum_{i=1}^{n} \lambda_i.$$

- The determinant of A is equal to the product of its eigenvalues

$$|A| = \prod_{i=1}^{n} \lambda_i.$$

- The rank of A is equal to the number of non-zero eigenvalues of A.
- The eigenvalues of a diagonal matrix are just the diagonal entries.

Diagonalization

- We can write all the eigenvector equations simultaneously as

$$AX = X\Lambda\,.$$

with $X \in \mathbb{R}^{n\times n}$ the eigenvectors of A and a diagonal matrix Λ whose entries are the eigenvalues A, ie,

$$X \in \mathbb{R}^{n\times n} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n).$$

Diagonalization

- If the eigenvectors of A are linearly independent, then X will be invertible, so

$$A = X\Lambda X^{-1}.$$

We say that A is diagonalizable.

What Just Happened?

- Linear Equations
- Matrices / Special Matrices
- Subspaces and Projection
- Eigenvalues and Eigenvectors

Halfway done with linear algebra!

Linear Algebra and Calculus!

Jeremy Irvin and Daniel Spokoyny

Eigenvalues and Eigenvectors

- Given a square matrix $A \in \mathbb{R}^{n \times n}$, we say $\lambda \in \mathbb{C}$ is an eigenvalue of A and $x \in \mathbb{C}^n$ is the corresponding eigenvector if

$$Ax = \lambda x, \quad x \neq 0.$$

- Intuitively, this means multiplying A by x results in a new vector in the same direction as x but scaled by λ.

Eigenvalues and Eigenvectors

- Note that if x is an eigenvector, then cx is an eigenvector for any complex c.

- We can rewrite the equation above as

$$(\lambda I - A)x = 0, \quad x \neq 0.$$

- $(\lambda I - A)x = 0$ has a non-zero solution iff $(\lambda I - A)$ has a non-empty nullspace, which only happens if $(\lambda I - A)$ is singular, ie,

$$|(\lambda I - A)| = 0.$$

E-values and E-vectors Properties

- The trace of A is equal to the sum of its eigenvalues,

$$\operatorname{tr} A = \sum_{i=1}^{n} \lambda_i.$$

- The determinant of A is equal to the product of its eigenvalues

$$|A| = \prod_{i=1}^{n} \lambda_i.$$

- The rank of A is equal to the number of non-zero eigenvalues of A.
- The eigenvalues of a diagonal matrix are just the diagonal entries.

Diagonalization

- We can write all the eigenvector equations simultaneously as

$$AX = X\Lambda\,.$$

with $X \in \mathbb{R}^{n\times n}$ the eigenvectors of A and a diagonal matrix Λ whose entries are the eigenvalues A, ie,

$$X \in \mathbb{R}^{n\times n} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n).$$

Diagonalization

- If the eigenvectors of A are linearly independent, then X will be invertible, so

$$A = X\Lambda X^{-1}.$$

We say that A is diagonalizable.

Quadratic Forms

- Given any symmetric matrix $A \in \mathbb{R}^{n \times n}$ and vector $x \in \mathbb{R}^n$, the scalar value is called a $x^T Ax$ quadratic form.

- Explicitly, we have

$$x^T Ax = \sum_{i=1}^{n} x_i (Ax)_i = \sum_{i=1}^{n} x_i \left(\sum_{j=1}^{n} A_{ij} x_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j \quad .$$

Definite Matrices

- A is positive definite if for all non-zero vectors $x \in \mathbb{R}^n$

$$x^T Ax > 0.$$

- A is negative definite if for all non-zero vectors $x \in \mathbb{R}^n$

$$x^T Ax < 0.$$

- Positive and negative definite matrices are full rank and thus invertible.

- For any matrix $A \in \mathbb{R}^{m \times n}$, $A^T A$ is positive semidefinite.

E-values and E-vectors of Symmetric Matrices

- Let $A \in \mathbb{R}^{n \times n}$ be any symmetric matrix:
  - All eigenvalues of A are real.
  - The non-collinear eigenvectors of A are orthonormal.
  - Thus we can decompose A:

$$A = U \Lambda U^T$$

where U is an orthogonal matrix.

E-values and E-vectors of Symmetric Matrices

- We can use this to show that definiteness only depends on sign of eigenvalues:

$$x^T A x = x^T U \Lambda U^T x = y^T \Lambda y = \sum_{i=1}^{n} \lambda_i y_i^2$$

  where $y = U^T x$.
- For any quadratic form $x^T A x$ subject to $\|x\|_2^2 = 1$, its maximum value is the maximum eigenvalue of A, and its minimum value is the minimum eigenvalue of A.

Singular Value Decomposition (SVD)

- Goal: Given any matrix $A \in \mathbb{R}^{m \times n}$, find orthogonal matrices U and V such that

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

- If A diagonalizable,

$$A = X \Lambda X^{-1}.$$

- If A positive semidefinite,

$$A = U \Lambda U^T$$

Singular Value Decomposition (SVD)

- See blackboard pictures.
- We know we can find an orthonormal basis for the rowspace of A.
- Can we find one that is mapped into an orthonormal basis for the column space of A?

Singular Value Decomposition (SVD)

- Goal: Find an orthonormal basis $v_1, \ldots, v_r$ for the row space of A such that

$$Av_1 = \sigma_1 u_1, \ldots, Av_r = \sigma_r u_r$$

where $u_1, \ldots, u_r$ is an orthonormal basis for the column space of A.

Singular Value Decomposition (SVD)

- In matrix notation,

$$A \begin{bmatrix} | & & | & | & & | \\ v_1 & \cdots & v_r & v_{r+1} & \cdots & v_m \\ | & & | & | & & | \end{bmatrix} = \begin{bmatrix} | & & | & | & & | \\ u_1 & \cdots & u_r & u_{r+1} & \cdots & u_n \\ | & & | & | & & | \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ \hline & 0 & & 0 \end{array}\right]$$

where $v_{r+1}, \ldots, v_m$ orthonormal basis for the null space of A, and $u_{r+1}, \ldots, u_n$ an orthonormal basis for the null space of A^T .

- Or

$$AV = U\Sigma$$

Singular Value Decomposition (SVD)

- But because $\mathcal{N}(A)$ and $\mathcal{R}(A)$ are orthogonal complements, V is orthogonal.
- Similarly, U is orthogonal.
- Therefore

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

Singular Value Decomposition (SVD)

- How do we find U and V?
- Trick:

$$A^T A = V \Sigma^T U^T U \Sigma V^T = V \left[\begin{array}{ccc|c} \sigma_1^2 & & & \\ & \ddots & & 0 \\ & & \sigma_r^2 & \\ \hline & 0 & & 0 \end{array}\right] V^T$$

- Since $A^T A$ is positive semidefinite, V is the orthogonal matrix of eigenvectors of $A^T A$, and its eigenvalues are the squares of the diagonal entries of Σ.

Singular Value Decomposition (SVD)

- Similarly,

$$AA^T = U\Sigma^T V^T V \Sigma U^T = U \left[\begin{array}{ccc|c} \sigma_1^2 & & & \\ & \ddots & & 0 \\ & & \sigma_r^2 & \\ \hline & 0 & & 0 \end{array}\right] U^T$$

- So to find U, simply find the eigenvectors of AA^T .

SVD Example

$$A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix}$$

Find orthonormal v_1, v_2 in the row space of A ($\mathbb{R}^2$) and orthonormal u_1, u_2 in the column space of A ($\mathbb{R}^2$), and $\sigma_1, \sigma_2 > 0$ such that

$$Av_1 = \sigma_1 u_1, Av_2 = \sigma_2 u_2$$

SVD Example

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

Therefore the eigenvectors of $A^T A$ are

$$\begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

with eigenvalues *32* and *18* respectively.

SVD Example

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

Therefore the eigenvectors of $A^T A$ are

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

again with eigenvalues *32* and *18* respectively (is this surprising?)

SVD Example

Therefore:

$$A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{32} & 0 \\ 0 & \sqrt{18} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = U\Sigma V^T$$

SVD Example 2

$$A = \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix}$$

See blackboard for geometric intuition.

$$A^T A = \begin{bmatrix} 80 & 60 \\ 60 & 45 \end{bmatrix}$$

SVD Example 2

- Therefore the eigenvalues of $A^T A$ are *0* and *125*. Hence:

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

So who cares?

- Why is SVD even useful?

- SVD can be used for **dimensionality reduction -** given high dimensional data, one can use SVD to represent the data using less dimensions, while still capturing the most significant (largest eigenvalues) features.

- We will see important applications later.

Multivariable Calculus

Jeremy Irvin and Daniel Spokoyny

Derivative

- Let $E \subseteq \mathbb{R}$ be open. A function $f : E \to \mathbb{R}^m$ is differentiable at $x \in E$ if

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

exists. We call this value *f'(x)*.

- Intuitively, a function is differentiable if it is locally approximated by a line.

Derivative

- If f is differentiable, we can rewrite this as

$$\lim_{h \to 0} \frac{f(x+h) - f(x) - f'(x)h}{h} = 0$$

- Or equivalently,

$$\lim_{h \to 0} \frac{|f(x+h) - f(x) - f'(x)h|}{|h|} = 0$$

Derivative

- Now let $E \subseteq \mathbb{R}^n$ be open, and $f : E \to \mathbb{R}^m$. Then f is differentiable at $x \in E$ if $\exists f'(x) \in \mathcal{L}(\mathbb{R}^n, \mathbb{R}^m)$ such that

$$\lim_{h \to 0} \frac{||f(x+h) - f(x) - f'(x)h||_{\mathbb{R}^m}}{||h||_{\mathbb{R}^n}} = 0$$

- Intuitively, f is differentiable if it is locally approximated by a linear function.

Partial Derivative

$E \subseteq \mathbb{R}^n$

- Let $f : E \to \mathbb{R}$. The jth partial derivative of f at $x \in E$ is

$$\lim_{t \to 0} \frac{f(x + te_j) - f(x)}{t} = D_j f(x) = \frac{\partial f}{\partial x_j}$$

provided this limit exists.

Total and Partial Derivative

- Now let $E \subseteq \mathbb{R}$ and $f : E \to \mathbb{R}^m$. We can write

$$f(x) = (f_1(x), ..., f_m(x))$$

- And if $E \subseteq \mathbb{R}^n$ and $f : E \to \mathbb{R}^m$, writing the component of *x* out explicitly,

$$f(x_1, ..., x_n) = (f_1(x_1, ..., x_n), ..., f_m(x_1, ..., x_n))$$

Total and Partial Derivative

- So we can define

$$\frac{\partial f_i}{\partial x_j} = D_j f_i(x) = \lim_{t \to 0} \frac{f_i(x + te_j) - f_i(x)}{t}$$

- Because *f'(x)* is linear, it can be represented as a matrix, call it $A \in \mathbb{R}^{m \times n}$. In fact,

$$A_{ij} = \frac{\partial f_i}{\partial x_j} = D_j f_i(x)$$

Total and Partial Derivative

- If *f* is real-valued ($f : E \to \mathbb{R}$), then the matrix representation of *f'(x)* is called the gradient of *f* at *x*, denoted

$$\nabla f(x_1, \ldots, x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

- Think of the gradient as a direction in which the parameters move so that the function *f* increases the fastest.

Convexity

- Let $E \subseteq \mathbb{R}$. A function $f : E \to \mathbb{R}$ is convex if for all $x_1, x_2 \in E, t \in [0, 1]$,

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

- This means that the line between any two points on the graph of *f* lies above the graph of *f* (see blackboard).

- **Convex functions have a single global optimum.**

Lagrange Multipliers

Jeremy Irvin and Daniel Spokoyny

Lagrange Multipliers

- Suppose $g : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, and let M be the set of points $x \in \mathbb{R}^n$ such that $g(x) = 0$ and $\nabla g(x) \neq 0$.
  If the differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ attains its maximum or minimum on M at the point $a \in M$, then

$$\nabla f(a) = \lambda \nabla g(a)$$

λ is called the “Lagrange multiplier”.

Lagrange Multiplier Example

- Find the rectangular box of volume 1000 which has the least total surface area A.

- Let $A = f(x, y, z) = 2xy + 2xz + 2yz$ and $g(x, y, z) = xyz - 1000\,.$

- We want to minimize f on the set of points which satisfy $g(x, y, z) = 0\,.$

- Sounds like Lagrange Multipliers!

Lagrange Multiplier Example

- $\nabla f = (2y + 2z, 2x + 2z, 2x + 2y)$
- $\nabla g = (yz, xz, xy)$
- We want to solve

$$2y + 2z = \lambda yz$$

$$2x + 2z = \lambda xz$$

$$2x + 2y = \lambda xy$$

$$xyz = 1000$$

- It is easily seen that the unique solution to this set of equations is $x=y=z=10$.

Generalized Lagrange Multipliers

- Informally, given some constraints $g_1(x) = 0, \ldots, g_m(x) = 0$, and denoting M the set of points which satisfy them, if (under some conditions on these constraints) the differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ attains a local maximum or minimum on M at $a \in M$, then

$$\nabla f(a) = \lambda_1 \nabla g_1(a) + \cdots + \lambda_m \nabla g_m(a)$$

- So to find points which optimize *f* given some constraints, simply solve the set of equations above.

Matrix Calculus

Matrix Gradient

- Let $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ (input matrix, output real value). The gradient of f with respect to some input $A \in \mathbb{R}^{m \times n}$ is the matrix of partial derivatives:

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

- More compactly,

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}$$

Matrix Derivative Properties

$$
\begin{aligned}
\nabla_A \mathrm{tr} AB &= B^T \\
\nabla_{A^T} f(A) &= (\nabla_A f(A))^T \\
\nabla_A \mathrm{tr} ABA^T C &= CAB + C^T AB^T \\
\nabla_A |A| &= |A|(A^{-1})^T.
\end{aligned}
$$

Hessian

- Suppose $f : \mathbb{R}^n \to \mathbb{R}$. The Hessian matrix with respect to x is the n x n matrix of partial derivatives:

$$\nabla_x^2 f(x) \in \mathbb{R}^{n \times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

- More compactly,

$$(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

Gradients/ Hessians of Quadratic/ Linear Functions

- $\nabla_x b^T x = b$
- $\nabla_x x^T A x = 2Ax$ (if A symmetric)
- $\nabla_x^2 x^T A x = 2A$ (if A symmetric)

What Just Happened?

- Multivariable Derivative
- Convexity
- Lagrange Multipliers
- Matrix Calculus

Done with math!! (kinda)

Linear Regression

Jeremy Irvin and
Daniel Spokoyny

Created from Andrew Ng's Stanford CS229 Notes, MIT Linear Algebra Lecture Video 16

Supervised vs. Unsupervised

- The ultimate goal of a machine learning algorithm is to allow a machine to learn from data and make predictions/ inferences from that data automatically (without hand-made rules).

- There are two main different types of learning algorithms.

- Unsupervised learning algorithms learn from *unlabeled* data, whereas supervised learning algorithms learn from *labeled* data.

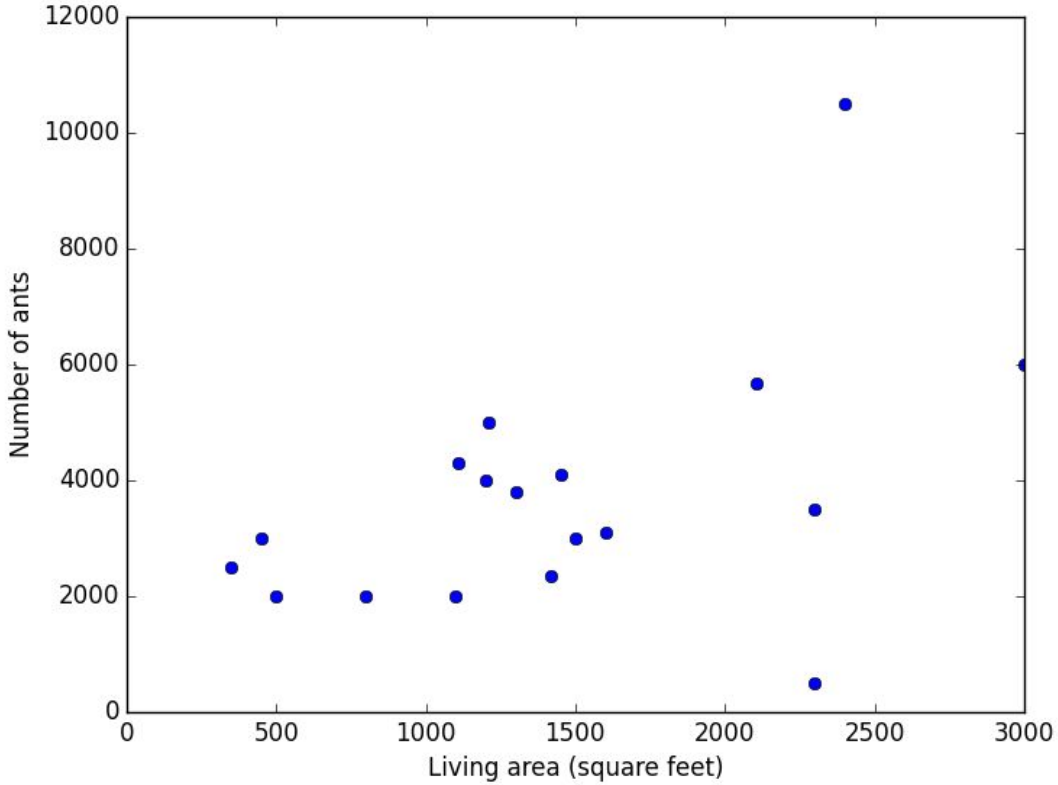
Supervised Learning Example (Linear Regression)

- Suppose we are given some data from Isla Vista residences:

Living area (feet^2)	Number of ants
2104	5678
1600	100
2400	10500
1416	234
3000	50000
⋮	⋮

Supervised Learning

- We can plot this data:



- We want to predict the number of ants in other residences from the size of their living areas.

Supervised Learning

- Maybe we have more relevant features in the data to help us predict:

Living area (feet^2)	year built	Number of residents	Number of ants
2104	1950	4	5678
1600	1975	2	100
2400	50	15	10500
1416	1915	5	234
3000	2010	3	50000
⋮	⋮	⋮	⋮

Supervised Learning Notation

- $x^{(i)}$ will denote the "input" variables, called input features (living area, year built, number of residents in our example).

- $y^{(i)}$ will denote the "output" variable, or target variable that we are trying to predict (the number of ants).

- $(x^{(i)}, y^{(i)})$ will denote a training example.

- $\{(x^{(i)}, y^{(i)}) | i = 1, ..., m\}$ will denote a training set.

Supervised Learning Notation

- $\mathcal{X}$ will denote the space input values and $\mathcal{Y}$ will denote the space of output values.

- We want to learn a function $h : \mathcal{X} \to \mathcal{Y}$ so that *h(x)* is a good predictor of the corresponding value of *y*.

- *h* is called the hypothesis.

- When the target variable is continuous, the learning problem is called regression. If it is discrete, it is called classification.

Linear Regression

- In linear regression, we want to find a *best* fit line to our data.

- In our example, we restrict *h* to functions of the form:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

- The $\theta_i' s$ are the parameters (also called weights).

- We want to choose the $\theta_i' s$ so that *h* is the *best* line.

Linear Regression

- We can generalize this to arbitrary (n) numbers of features, and write (letting $x_0 = 1$):

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

- So what does *best* fit line mean?

- We define the cost function J which measures how close the $h(x^{(i)})$'s are to the $y^{(i)}$'s

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient Descent

- We want to choose θ to minimize the error $J(\theta)$.
- Calculus? We will see this later.
- What we can do is use gradient descent, we update θ by repeatedly taking steps in the steepest decrease of J, ie, the opposite direction of the gradient.

Gradient Descent

- Specifically, we want to perform the update

$$\theta := \theta - \alpha \nabla J(\theta)$$

- Componentwise, for *j=0,...,n*,

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- α is called the learning rate.

Gradient Descent

- So what is $\frac{\partial}{\partial \theta_j} J(\theta)$? Let's compute it when we only have one training example *(x,y)*:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y \right) \\
&= \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y \right) x_j
\end{aligned}
$$

Gradient Descent

- This gives the update rule:

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

for each individual training example $(x^{(i)}, y^{(i)}), i = 1, ..., m$.

- This is the “least mean squares” (LMS) update rule.

- We can iterate over the examples in our training set and update every time until *convergence* - this is called stochastic gradient descent.

Gradient Descent

- We could also perform the following update rule until convergence:

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

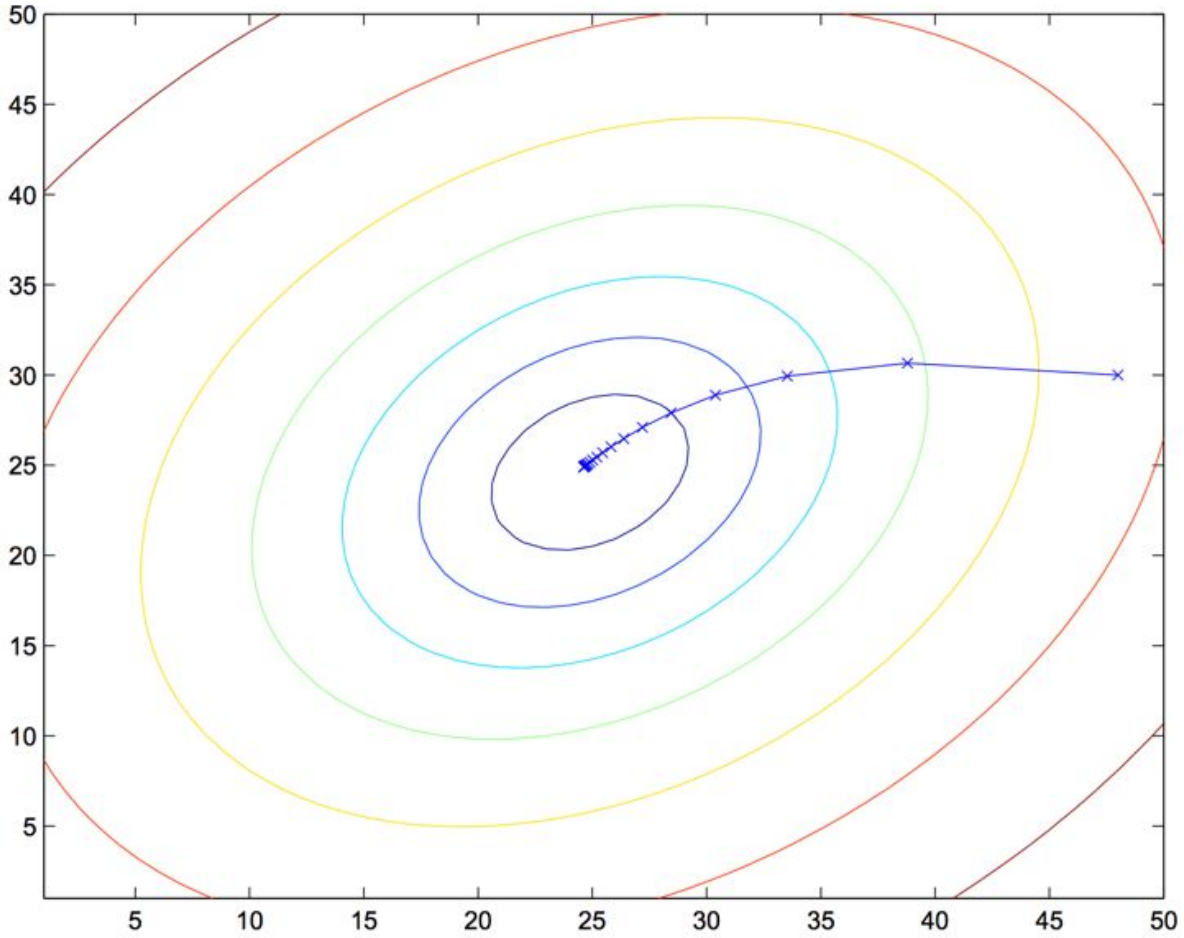
- The right term in the sum is just $\frac{\partial J(\theta)}{\partial \theta_j}$ for the original *J* with all training examples.

- This algorithm is known as batch gradient descent.

- *J* is a convex function, so batch gradient descent 'always' converges (approximately) to the *global* minimum.

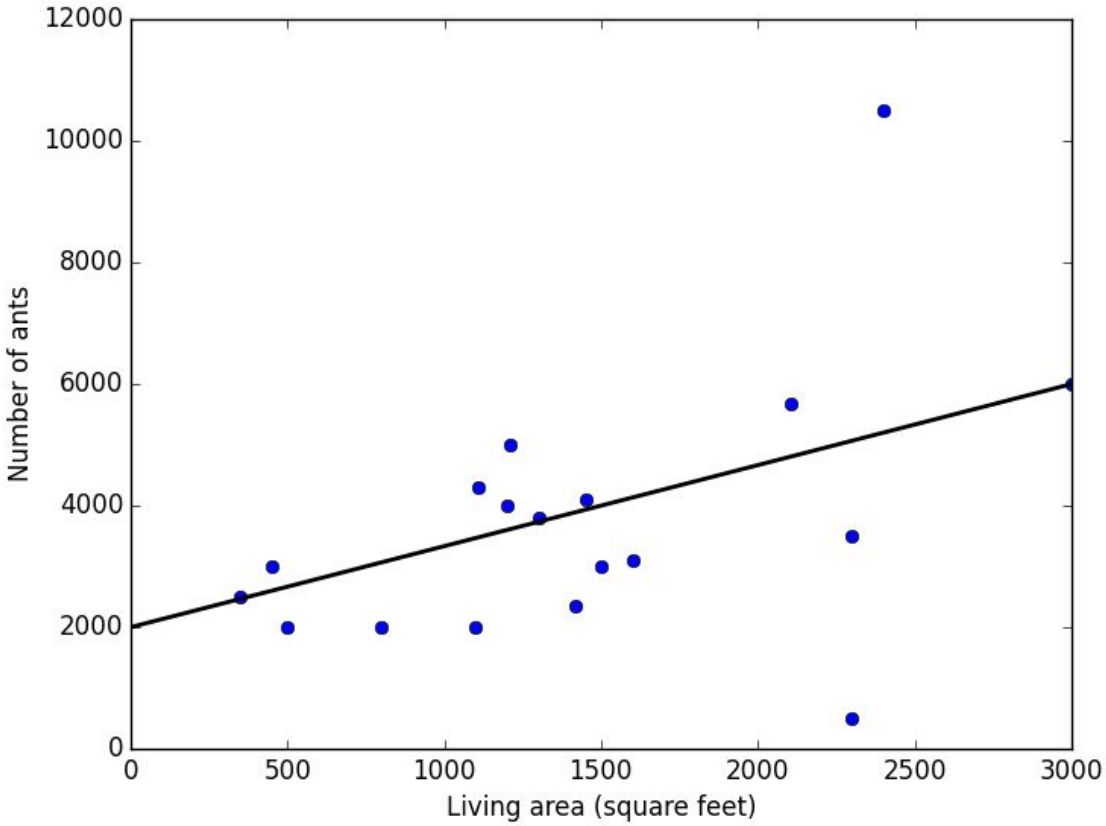
Gradient Descent

- Here is an example of batch gradient descent:



Gradient Descent

- Applying this algorithm to our Isla Vista data:



Batch vs. Stochastic

- Batch has to scan through the whole dataset before taking a step - costly if m is large.
- Stochastic takes a step after every training example, and thus approaches the minimum much faster.
- However, batch always converges, but stochastic may oscillate around the minimum (in practice these are still good approximations of the true minimum)
- Hence stochastic gradient descent is preferred when the training set is large.

Linear Algebra
Recall: Projection

- Assuming A is full rank and $n<m$, the projection of $y \in \mathbb{R}^m$ onto the range (column space) of A is

$$\text{Proj}(y; A) = \text{argmin}_{v \in \mathcal{R}(A)} \|v - y\|_2 = A(A^T A)^{-1} A^T y$$

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

Projection

- If $b \in R(A)$, then $Pb = b$.
  - $b = Ax$ thus

$$Pb = A(A^T A)^{-1} A^T Ax = Ax = b$$

- If $b \in \mathcal{N}(A^T)$, then $Pb = 0$.
  - $A^T b = 0$ thus

$$Pb = A(A^T A)^{-1} A^T b = 0$$

  - Take for example the column space of A to be a plane and b a perpendicular vector.

Projection

- See drawing on board.
- So we have that

$$Pb + (I - P)e = p + e = b$$

- Note that $I - P$ projects vectors onto the perpendicular space (check for yourself).

- Also check for yourself that if P is a projection matrix, then

$$(I - P)^2 = I - P$$

Linear Algebra

- We can actually interpret linear regression as a projection.

- For example, suppose we are given the following points in the plane:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

- See the drawing on the board.

Linear Algebra

- We want to find the best fit line, ie, find C and D for the line $y=C+Dt$.

- Equivalently, we want to solve the following systems of equations

$$C + D = 1$$
$$C + 2D = 2$$
$$C + 3D = 2$$

Linear Algebra

- We can rewrite this using matrix notation:

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$Ax = b$$

- But notice this system has no solution - b is not in the column space of A.

- We hope to find the “best” solution!

Linear Algebra

- We will have some error on the best fit line.
- We will measure this error as before, namely

$$||Ax - b||^2 = ||e||^2$$

- We want to find *x* to minimize this error.
- Notice that the error is *0* iff there exists some *x* such that *Ax=b*, ie, $b \in \mathcal{R}(A)$
- In our example, $||e||^2 = e_1^2 + e_2^2 + e_3^2$, see blackboard.

Linear Algebra

- In examples with outliers, this choice of error may not be the best. This is something to keep in mind.
- There are two pictures to keep in mind here. See the blackboard.
- We wish to find some $\hat{x} = \begin{bmatrix} \hat{C} \\ \hat{D} \end{bmatrix}$ which minimizes the squared error.
- In order to do this, we solve the normal equations:

$$A^T A\hat{x} = A^T b$$

Linear Algebra

- In our example,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \hat{C} \\ \hat{D} \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix} \begin{bmatrix} \hat{C} \\ \hat{D} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

- Observe that the matrix is symmetric, invertible, and positive semidefinite (is this always true?).

- Simplifying this yields the normal equations

$$3\hat{C} + 6\hat{D} = 5$$
$$6\hat{C} + 14\hat{D} = 11$$

Linear Algebra

- If we had used calculus instead by minimizing

$$||e||^2 = (C + D - 1) + (C + 2D - 2)^2 + (C + 3D - 2)^2$$

by taking partial derivatives and setting equal to *0*, it would yield the identical normal equations.

  - This set of equations is always linear because the error function is quadratic!

- Solving this we get

$$\hat{D} = \frac{1}{2}, \hat{C} = \frac{2}{3}$$

Linear Algebra

- So the best line with respect to squared error is

$$y = \frac{2}{3} + \frac{1}{2}t$$

- This yields the following points, as seen on the blackboard:

$$p_1 = \frac{7}{6}, p_2 = \frac{5}{3}, p_3 = \frac{13}{6}$$

$$e_1 = -\frac{1}{6}, e_2 = \frac{2}{6}, e_3 = -\frac{1}{6}$$

Linear Algebra

- So in the other picture,

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 7/6 \\ 5/3 \\ 13/6 \end{bmatrix} \quad e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} -1/6 \\ 2/6 \\ -1/6 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = p + e$$

Linear Algebra

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 7/6 \\ 5/3 \\ 13/6 \end{bmatrix} \quad e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} -1/6 \\ 2/6 \\ -1/6 \end{bmatrix}$$

- Notice that *p* and *e* are perpendicular.

- In fact, *e* is perpendicular to any vector in the column space of *A*.
  - Test each column of *A*.

- *C* and *D* is the combination of the 2 columns that give *p*.

Linear Algebra

- So given a set of points, here is the algorithm to find the best fit line:

1. Construct the matrix A as we did in the example.

2. Solve the normal equations

$$A^T A\hat{x} = A^T b$$

for $\hat{x}$.

3. To find the predicted values, compute

$$p = A\hat{x}$$

Probabilistic Interpretation

- Assume that the target variables and inputs are related via the equation

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

where $\epsilon^{(i)}$ is an error term representing either unmodeled effects or random noise.

- Also assume that $\epsilon^{(i)}$ are IID (independently and identically distributed) from a Gaussian distribution with mean 0 and variance σ^2.

Probabilistic Interpretation

- This means that

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

which means that

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

where $p(y^{(i)}|x^{(i)};\theta)$ is the distribution of $y^{(i)}$ given $x^{(i)}$ and *parameterized* by θ.

Design Matrix

- Given a training set, define the design matrix X to be the the matrix whose rows are the training examples:

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ - & (x^{(3)})^T & - \end{bmatrix}$$

- Also let

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Likelihood Function

- Given the design matrix X and θ, what is the distribution of the $y^{(i)}$'s?

- The probability of the data is given by $p(\vec{y}|X;\theta)$. This is typically viewed as a function of $\vec{y}$ for a fixed θ.

- When view as a function of θ, it is called the likelihood function:

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta)$$

Likelihood Function

- Due to the independence of the $\epsilon^{(i)}$'s (and thus the $y^{(i)}$'s given the $x^{(i)}$'s), then

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right)
\end{aligned}
$$

Maximum Likelihood

- So how do we choose θ?
- We want to choose θ to maximize the probability of our data, ie, to maximize $L(\theta)$.
- But $L(\theta)$ is ugly to maximize - the trick is that any monotone increasing function of $L(\theta)$ will yield the same parameter.
- We will maximize the log likelihood:

$$\ell(\theta) = \log L(\theta)$$

Maximum Likelihood

- Simplifying $\ell(\theta) = \log L(\theta)$ yields

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}
$$

Maximum Likelihood

- So maximizing $\ell(\theta)$ is the same as minimizing

$$\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$$

- Hence, given our probabilistic assumptions on the data, least-squares-regression corresponds to finding the maximum likelihood estimate of θ.

  Neato!

Matrix Calculus Interpretation

- Recall the design matrix X whose rows are the training example inputs, and column vector $\vec{y}$ whose entries are the training example outputs.

- Then since $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$,

$$
\begin{aligned}
X\theta - \vec{y} &= \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \\
&= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}.
\end{aligned}
$$

Matrix Calculus Interpretation

- So we're trying to minimize this function *J*. Why don't we just take the derivative and set to 0?

- We can actually do that! We will derive this method using matrix calculus.

- Turns out that to optimize some function F, setting derivatives to 0 and solving is only useful when $\nabla F(x) = 0$ happens to be a linear system (or at least a system in which *x* can be isolated).

Matrix Calculus Interpretation

- Then since for any vector z,

$$z^T z = \sum_i z_i^2$$

we have

$$\begin{aligned} \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) &= \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= J(\theta) \end{aligned}$$

Matrix Calculus Interpretation

- Then

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta \operatorname{tr}\left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta \left(\operatorname{tr}\theta^T X^T X\theta - 2\operatorname{tr}\vec{y}^T X\theta\right) \\
&= \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T\vec{y}\right) \\
&= X^T X\theta - X^T\vec{y}
\end{aligned}
$$

Matrix Calculus Interpretation

where:

- the third equality uses the fact that the trace of a real number is the real number,
- the fourth equality uses the fact that the trace of a matrix is the trace of its transpose,
- the fifth equality uses

$$\nabla_{A^T} \mathrm{tr} ABA^T C = B^T A^T C^T + BA^T C$$

with $A^T = \theta, B - B^T = X^T X, C = I$, and that $\nabla_A \mathrm{tr} AB = B^T$.

Matrix Calculus Interpretation

So to minimize *J*, we set its derivatives to zero, and we get the normal equations:

$$X^T X\theta = X^T \vec{y}$$

Solving for θ, if *X* has full column rank, we have

$$\theta = (X^T X)^{-1} X^T \vec{y}$$

Hey, the same as the linear algebra interpretation!

Matrix Calculus vs. Gradient Descent

- So solving for the maximal θ reduces to computing the matrix product above (which involves computing an inverse of a very large matrix).

- However, in practice, this inverse is never computed. Instead, the system is posed in the form $(X^T X)\theta = X^T \vec{y}$ and solved using a linear solver.

- This method is cheaper, and allows exploitation of the coefficient matrix (using bandedness, symmetry, sparsity) and other methods.

Matrix Calculus vs. Gradient Descent

- Bottom-line:

- when the first order derivative system is linear, solving it directly is much more computationally efficient than gradient descent (which can have slow convergence).

- Otherwise, other strategies (including gradient descent) may be better.

- Note: people like to use gradient descent for convex optimization because it is easy to implement and relatively cheap computationally.

What Just Happened?

- Linear Regression Interpretations:

1. Least Squares
2. Linear Algebra
3. Probabilistic
4. Matrix Calculus

Logistic Regression

Jeremy Irvin and
Daniel Spokoyny

Created from Andrew Ng's
Stanford CS229 Notes

Classification

- Recall that we we're trying to predict continuous values using regression.

- If we're trying to predict the values *y* which only take on a small amount of discrete values, it is called classification.

- For now we will focus on binary classification, ie, predicting either a **0** or a **1**.

- **0** will be called the negative class, and
  1 will be called the positive class.

Logistic Regression

- We could attempt to tackle this classification problem with the linear regression algorithm.

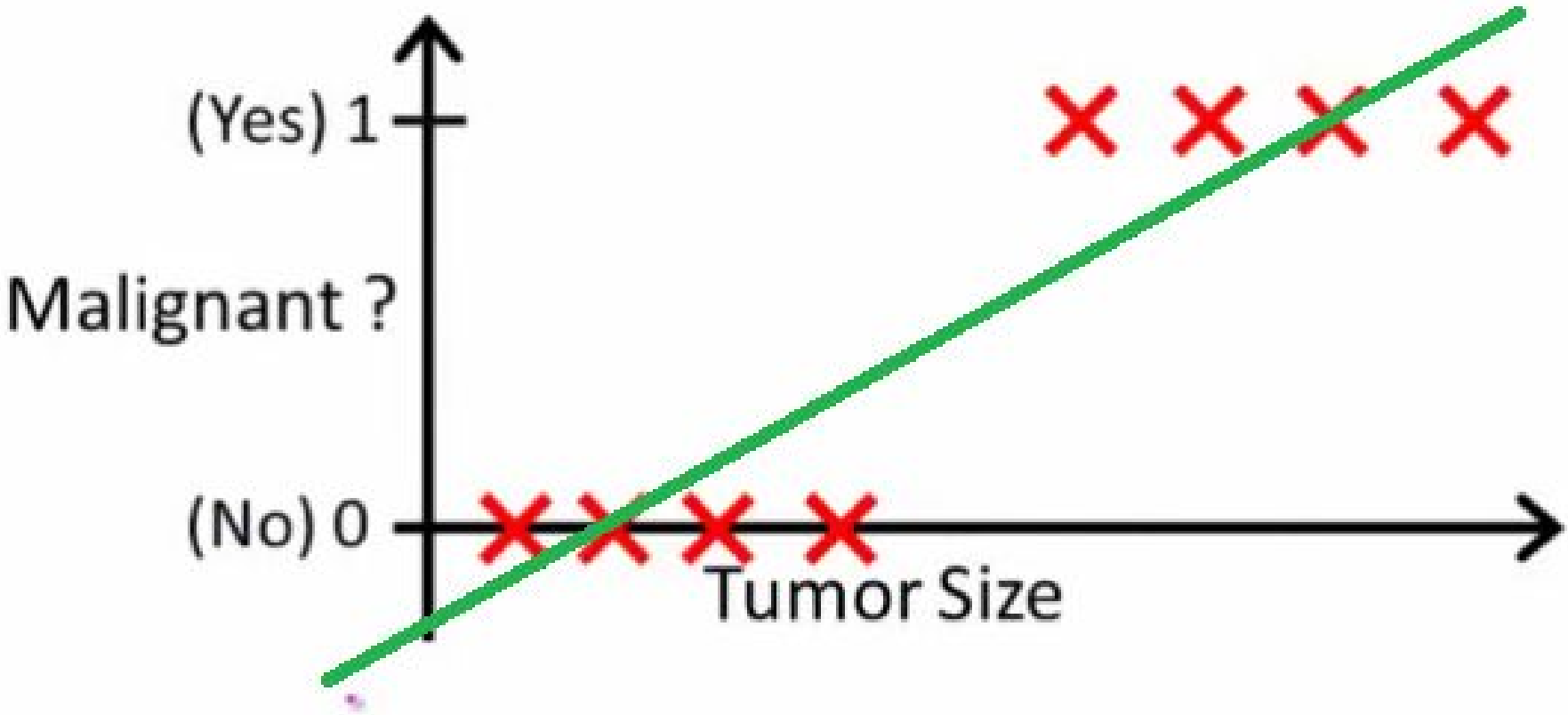
- However, it is easy to construct an example where this performs poorly, as we will see on the next slide.

- For initial intuition, it does not make sense for the hypothesis function to output values greater than 1 or less than 0 when $y \in \{0, 1\}$.

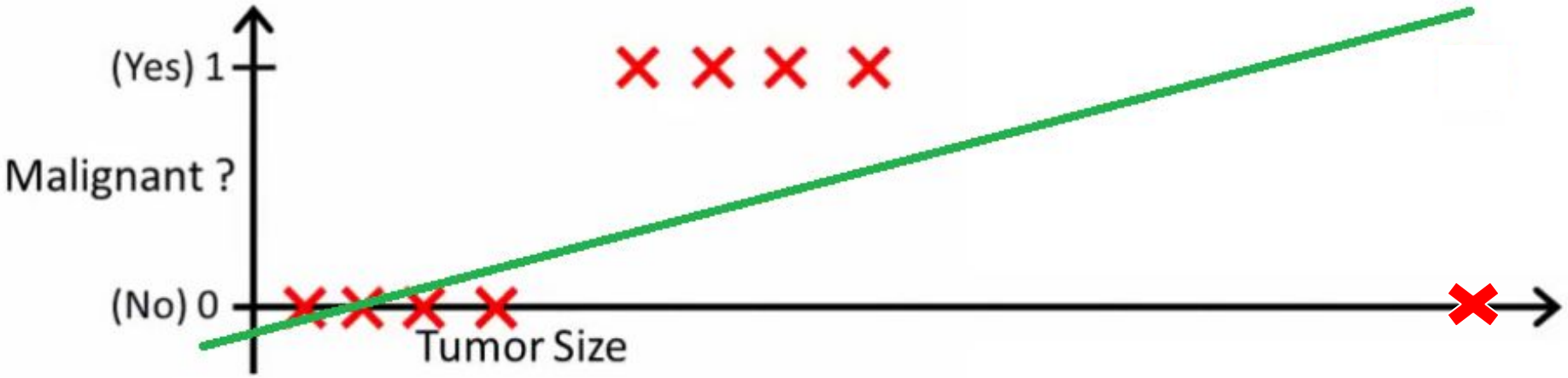
Linear Regression Binary Classification Example

- Suppose we are trying to predict whether a tumor is malignant based on its size.
- Malignant tumors are labeled **1**, and benign tumors are labeled **0**.
- To make predictions using linear regression, we could say if *h(x)* outputs a value larger than 0.5, predict malignant, otherwise predict benign.

Example Cont.



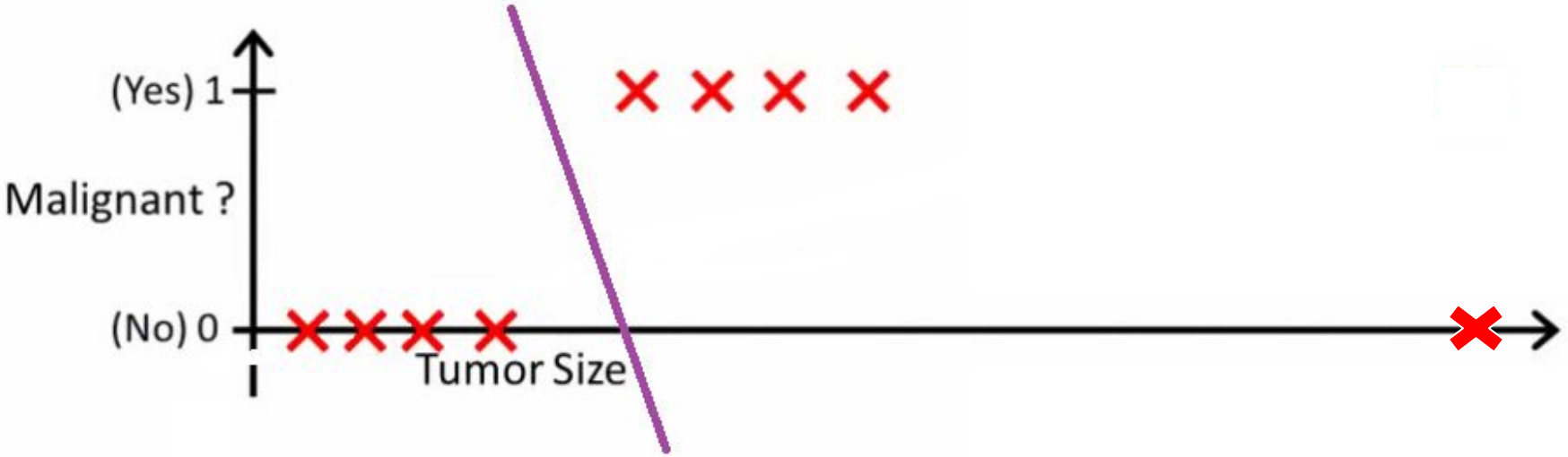
Example Cont.



- Now notice that $h(x) > 0.5 \Rightarrow$ malignant does not work anymore. We would have to alter our h.

- But we can't just change h every time a new sample arrives - it should be fixed after training.

Example Cont.



- Both linear and logistic predict straight lines.
- Linear interpolates the output and predicts the value for *x* we haven't seen.
- Logistic says all points sitting to the right of the classifier line belong to one class, and the left belong to the other.
  - In this case, *h(x)* represents the probability that *x* belongs to the positive class.

Sigmoid Function

- We will change the form of $h_\theta(x)$:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

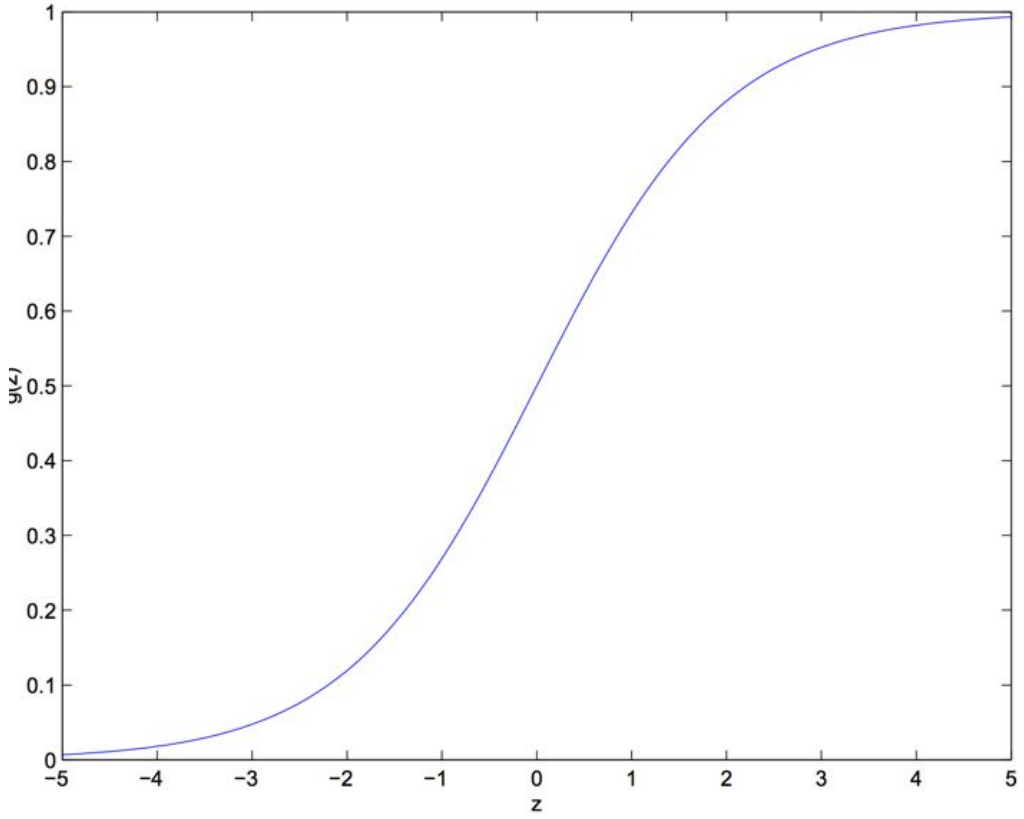
where

$$g(z) = \frac{1}{1 + e^{-z}}$$

is called the logistic or sigmoid function.

Sigmoid Function

- Here is a plot of *g(z)*: (visualize scaling)



$$\lim_{z \to \infty} g(z) = 1, \qquad \lim_{z \to -\infty} g(z) = 0$$

Sigmoid Function

- So we kind of arbitrarily chose this g due to the fact that it increases from 0 to 1.

- In fact, there are many reasons why we use the logistic function, the first being its *smoothness* and easily computable derivative:

$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

Fitting the Logistic Regression Model

- Similar to linear regression, we want to find θ to *best* fit our data for future predictions.

- Let's endow the classification problem with some probabilistic assumptions (as we did with linear regression), and then fit the parameters through maximum likelihood!

Fitting the Logistic Regression Model

- Assume that

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

- Or more compactly, that

$$p(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Fitting the Logistic Regression Model

- Assuming the m training examples were generated independently, the likelihood of the parameters is

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X; \theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

Fitting the Logistic Regression Model

- Maximizing the log likelihood will again be easier:

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))\end{aligned}$$

- We will maximize this function using gradient descent, thus we need to find its gradient. Let's do it component-wise on a single training example (x,y):

Fitting the Logistic Regression Model

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$\ell(\theta) = y \log h_\theta(x) + (1 - y) \log(1 - h_\theta(x))$$

Therefore:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x) \frac{\partial}{\partial \theta_j} \theta^T x \\
&= \left(y(1 - g(\theta^T x)) - (1 - y) g(\theta^T x) \right) x_j \\
&= (y - h_\theta(x)) \, x_j
\end{aligned}$$

since $g'(z) = g(z)(1 - g(z))$.

Fitting the Logistic Regression Model

- This gives us the stochastic gradient *ascent* rule:

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

- This looks identical to the LMS update rule!
- But this is a completely different algorithm.
- Is this a coincidence?
- No!! It is because they are both types of Generalized Linear Models (GLM's).

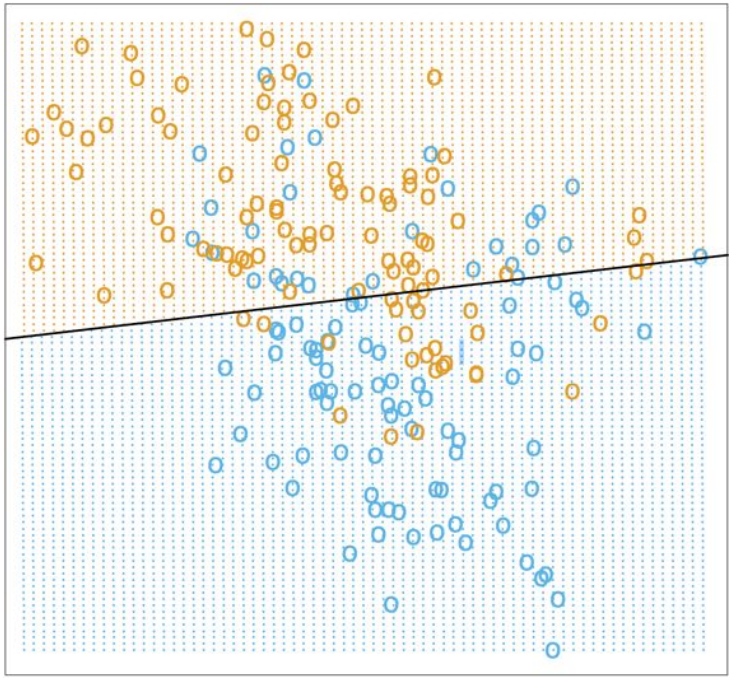
Statistical Decision Theory

Jeremy Irvin and
Daniel Spokoyny

Created from Elements of Statistical Learning (Hastie, Tibshirani, Friedman)

Two Basic Classifiers

- Just as we did in logistic regression, we can learn a linear decision boundary to perform binary classification.



- It seems like a linear assumption is too rigid. Or are errors on our predictions unavoidable?

Two Basic Classifiers

- The errors that we make by assuming a linear decision boundary of course depends on the specific training set we are using:
  - in none of these models have we specified where the data itself comes from.

- Let's examine two scenarios. The training data in each class were generated from:
  - bivariate Gaussians with uncorrelated components (variance matrix identity) and distinct means.
  - a mixture of 10 low-variance Gaussians, with the means themselves distributed as Gaussian.

Two Basic Classifiers

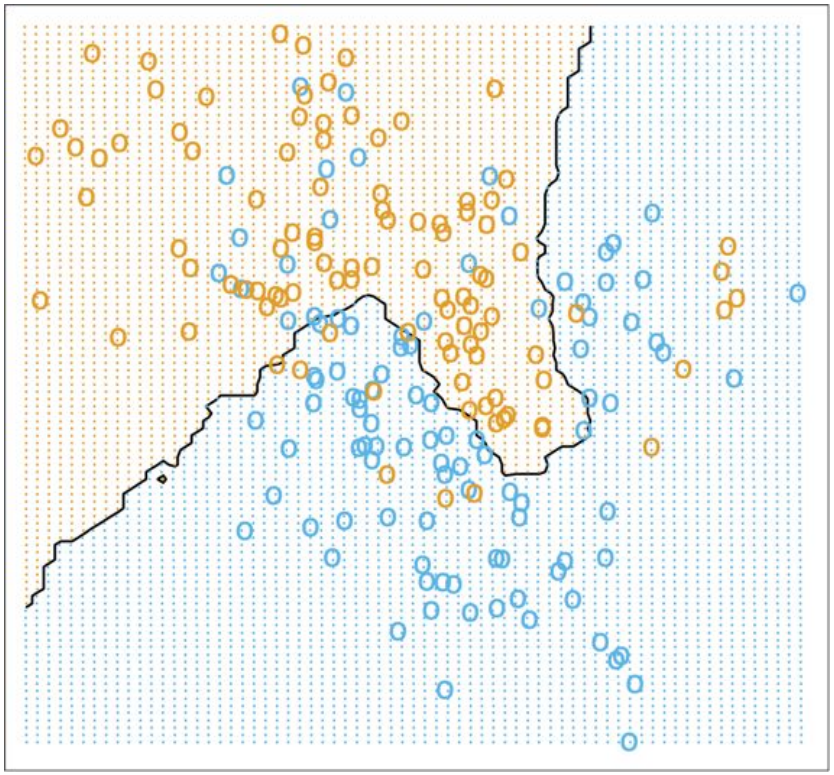
- Think of a mixture of Gaussians in the "generative" sense:
  - Generate a discrete random variable that determines which of the 10 distributions to generate (sample) from
  - Then generate from that chosen distribution

- If the data comes from one Gaussian per class, linear decision boundary is optimal.

- For tightly clustered Gaussians, a linear decision boundary is not optimal - optimal will most likely be linear and disjoint (and therefore difficult to learn).

k-Nearest Neighbors

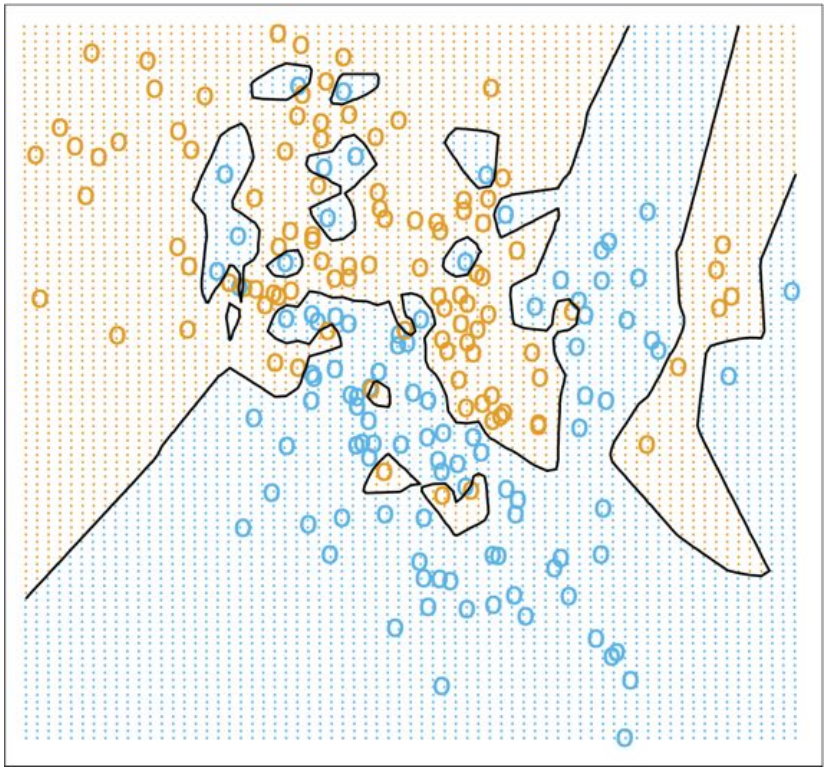
- Can do nearest neighbor methods using majority vote (15-NN):



- Seems much better - but in fact it’s not necessarily a good model. Why?

k-Nearest Neighbors

- This is the decision boundary generated using 1-NN:



- This is a perfect decision boundary for our training set. Why not always use this?

Bias and Variance

- A linear decision boundary is smooth and stable (small changes to our training set won't affect the line), but it relies heavily on the linearity assumption.
  - Low variance, high bias

- k-NN doesn't make any assumptions about the data, and can adapt to it well, but any local region is very susceptible to any change in the training set.
  - High variance, low bias

Statistical Decision Theory

- Let's generalize our original learning formulation:

- Let X denote a random variable whose distribution is the distribution of the inputs, and Y a random variable whose distribution is the distribution of the outputs.

- We want to find h to minimize the value $L(Y, h(X))$ for some loss function L (over the distribution).

- Put another way, we want to discover the joint distribution of the random variables to find the optimal h.

Statistical Decision Theory

- Take the loss function (as before) to be squared loss. Call $\mathcal{T}$ the training set. Then the expected prediction error for h over the training set $\mathcal{T}$ is

$$\mathrm{EPE}(h) = \mathbb{E}_{\mathcal{T}}[Y - h(X)]$$

- We hope to minimize this error. Turns out we can minimize it pointwise (ie, minimize it for each training example individually):

$$h(x) = \mathbb{E}[Y|X = x]$$

- This is known as the regression function.
- So the best prediction of Y at $X = x$ is the conditional mean when "best" is measured by average squared error.

Statistical Decision Theory

- k-NN in fact attempts to estimate this conditional mean.

- At any input x, the k-NN model yields

$$h(x) = \text{Ave}(y_i | x_i \in N_k(x))$$

Two estimations:

- The expectation is approximated by averaging over sample data.
- Conditioning at a single point is relaxed to condition on a region close to the point.

Statistical Decision Theory

- As the size *N* of our training set increases, these estimations become more and more accurate.
- The points in a neighborhood of x are close to x.
- As the number of neighbors *k* increases, the average will stabilize.
- In fact, it can be shown that if $N, k \to \infty$ with $k/N \to 0$ (the size of the training set increases much faster than the number of neighbors), then

$$f(x) \to \mathbb{E}[Y|X = x]$$

Statistical Decision Theory

- So it seems like we've found a universal approximator of this mean, and thus an optimal classifier in this general formulation.

- However, in practice, we often cannot get large enough samples for this approximation to yield good results.

- Additionally, if we know the structure of the data (such as linearity), models with this innate structure will be more stable (but this structure somehow needs to be discovered beforehand).

- Also, as the dimension of the input space becomes large, so does the k-NN neighborhood (the curse of dimensionality), causing the rate of convergence to greatly decrease.

Statistical Decision Theory

- Linear regression similarly approximates this conditional expectation by using the functional model assumption to pool over values of the input space.

- So least squares in this framework amounts to replacing this expectation with averages over the training data, like k-NN.

- Here's how the two models differ however:
  - least squares assumes h is well approximated by a globally linear function.
  - k-NN assumes h is well approximated by a locally constant function.

Bias-Variance Decomposition

- We can actually express the expected prediction error (using squared loss) as a decomposition into variance and squared bias (here MSE is mean squared error):

$$\begin{aligned}
\mathrm{MSE}(x_0) &= \mathbb{E}_{\mathcal{T}}[\hat{y}_0 - f(x_0)]^2 \\
&= \mathbb{E}_{\mathcal{T}}[\hat{y}_0 - \mathbb{E}_{\mathcal{T}}(\hat{y}_0)]^2 + [\mathbb{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\
&= \mathrm{Var}_{\mathcal{T}}(\hat{y}_0) + \mathrm{Bias}^2(\hat{y}_0)
\end{aligned}$$

- This is known as the bias-variance decomposition.

- This can be used to show (theoretically) the effect of bias and variance on the performance of the model.
  - See Elements of Statistical Learning for more details

Confusion Matrix

- Suppose we are performing binary classification.
- The following is known as the confusion matrix:

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)
	Predicted condition negative	**False negative** (Type II error)	**True negative**

Precision vs. Recall

- Precision is the number of true positives divided by the total number of positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall is the number of true positives divided by the total number of correctly classified points.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative.
- Recall is the ability of the classifier to find all the positive samples.

F1 Score

- One commonly used method of determining the quality of a binary classification model is to use the F1 Score, defined as the harmonic mean of precision and recall:

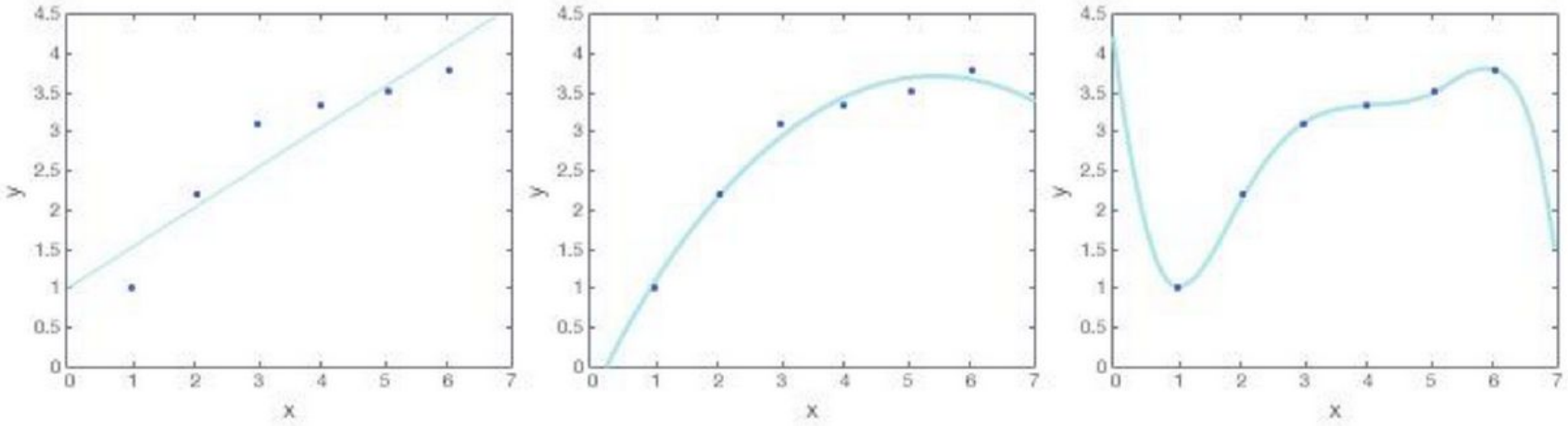
$$F_1 = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

- The best value is 1, the worst is 0.

Overfitting and Underfitting

- We have been discussing ways to evaluate your model.

- Two very common problems with a model are models which overfit and underfit.



Cross-Validation

- In practice, you are given data (let's say in the supervised setting - so data with labels).

- You hope to build a model and test the model.

- Typically, the data is split into parts - a training set, a test set, and a cross-validation set.

1. The model is first learned using the *training* set.
2. The best performing model (tuning/ choosing hyperparameters) is determined using the *validation* set.
3. The evaluation of the fully trained model is performed using the *test* set (no tuning at this point can occur).

Why separate validation and test sets? To prevent overfitting.

What Just Happened?

- Two Basic Classifiers (linear/ k-NN)
- Bias / Variance Tradeoff and Decomposition
- Confusion Table and F1 Score
- Overfitting and Underfitting
- Cross-Validation

Generative vs. Discriminative

Jeremy Irvin and
Daniel Spokoyny

Created from Andrew Ng's Stanford CS229 Notes and Coursera Stanford NLP

Introduction

- So far we've mostly talked about learning algorithms which model $p(y|x;\theta)$, the conditional distribution of *y* given *x*.

- For instance, logistic regression modeled $p(y|x;\theta)$ as $h_\theta(x) = g(\theta^T x)$, where *g* is the sigmoid function.

The Old Approach

- Consider a classification problem where we want to learn to distinguish between whether a house will fall of DP (*y=1*) or not (*y=0*), based on some features of the house (and its parties).

- Logistic regression tries to learn a straight line (a decision boundary) to separate the houses.

- Then to classify which house will fall off the cliff, it checks which side of the decision boundary the house lies, and predicts accordingly.

A New Approach

- Suppose instead we first look at the houses which have fallen off and build a model of what these houses look like.
- Then we look at the houses which are stable and build a model of what these houses look like.
- Finally, to classify a house, we can match the house against both models and see which best models it.

Discriminative vs. Generative

- Algorithms which try to learn $p(y|x)$ directly (such as logistic regression) or algorithms which try to learn mappings from the space of inputs to the labels {0,1} (such as the perceptron algorithm which we will discuss later) are called discriminative algorithms.

- Algorithms which try to model $p(x|y)$ (and $p(y)$) are called generative algorithms.

- In the our example, $p(x|y=0)$ models the distribution of stable house *features*, and $p(x|y=1)$ models the distribution of unstable house features.

Bayes Theorem!!!

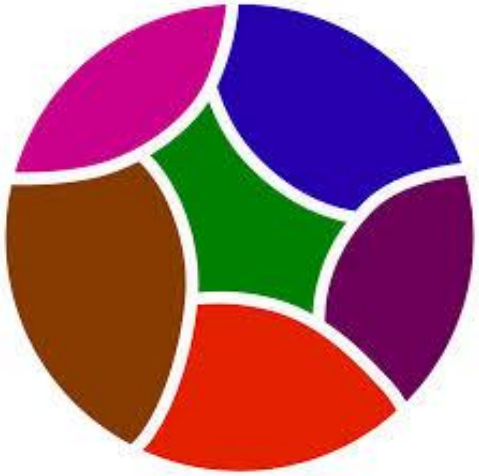
- If A and B are any two events, then

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

- If {*A_j*} is a partition of the sample space, then

$$P(B) = \sum_j P(B \mid A_j) P(A_j),$$

$$\Rightarrow P(A_i \mid B) = \frac{P(B \mid A_i)\,P(A_i)}{\sum_j P(B \mid A_j)\,P(A_j)}.$$



Discriminative vs. Generative

- After modeling $p(y)$ (called the class priors) and $p(x|y)$, we can then use Bayes rule to derive the posterior distribution of *y* given *x*:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{p(x|y=1)p(y=1) + p(x|y=0)p(y=0)}$$

- Hence everything can be computed using the quantities we have modeled. But we don't really need to compute the denominator since:

$$\begin{aligned} \arg\max_y p(y|x) &= \arg\max_y \frac{p(x|y)p(y)}{p(x)} \\ &= \arg\max_y p(x|y)p(y). \end{aligned}$$

Summary

- Generative classifiers learn a model of the joint probability $p(x, y) = p(x|y)p(y)$, of the inputs and label, and make their predictions by using Bayes rule to calculate $p(y|x)$.

- Discriminative classifiers model the posterior $p(y|x)$ directly, or learn a direct map from inputs to the labels.

- So a parametric family of probabilistic models $p(x, y)$ can be fit either to optimize the joint likelihood of the inputs and labels or the conditional likelihood.

Important Note

- A pair of classifiers, one discriminative and one generative which use the same parametric family of models, is called a Generative-Discriminative pair.

- For example, if $p(x|y)$ is Gaussian and $p(y)$ multinomial, then the Gen-Dis pair is Normal Discriminant Analysis and logistic regression.

- In the discrete case, the naive Bayes classifier and logistic regression form a Generative-Discriminative pair.

Support Vector Machines

Jeremy Irvin and
Daniel Spokoyny

Created from Andrew Ng's
Stanford CS229 Notes

Functional Margin Intuition

- Remember logistic regression, where $p(y = 1|x; \theta)$ is modeled by $h_\theta(x) = g(\theta^T x)$.

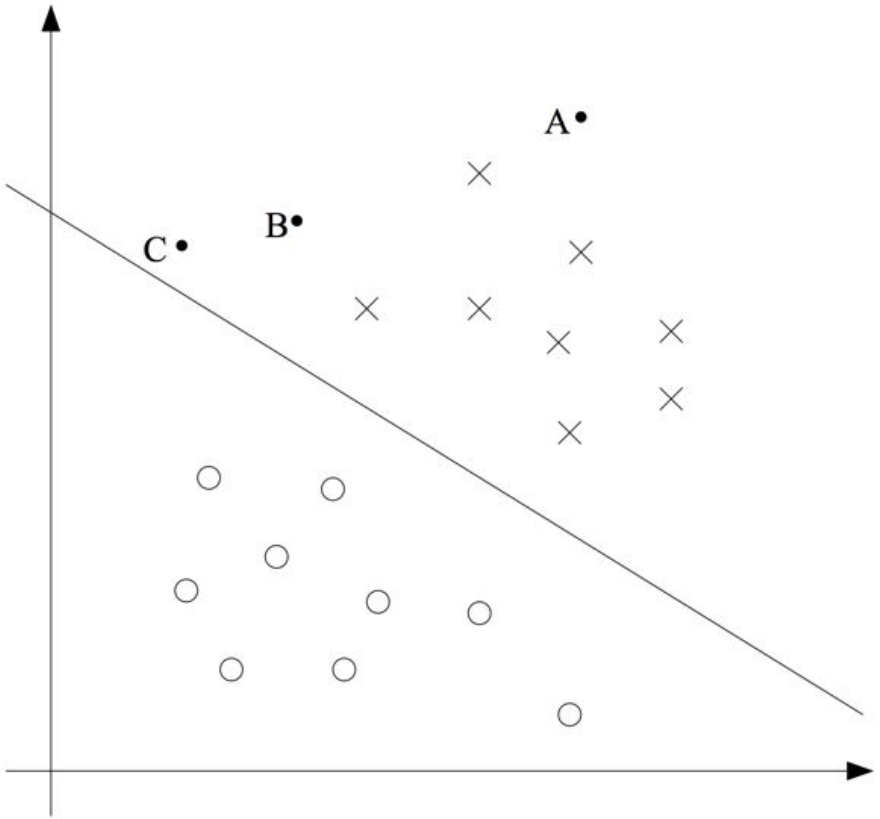
- We predict '1' if and only if $h_\theta(x) \geq 0.5$, or equivalently, $\theta^T x \geq 0$.

- So for a positive training example, the larger $\theta^T x$, the more 'confident' we are that the label is 1.

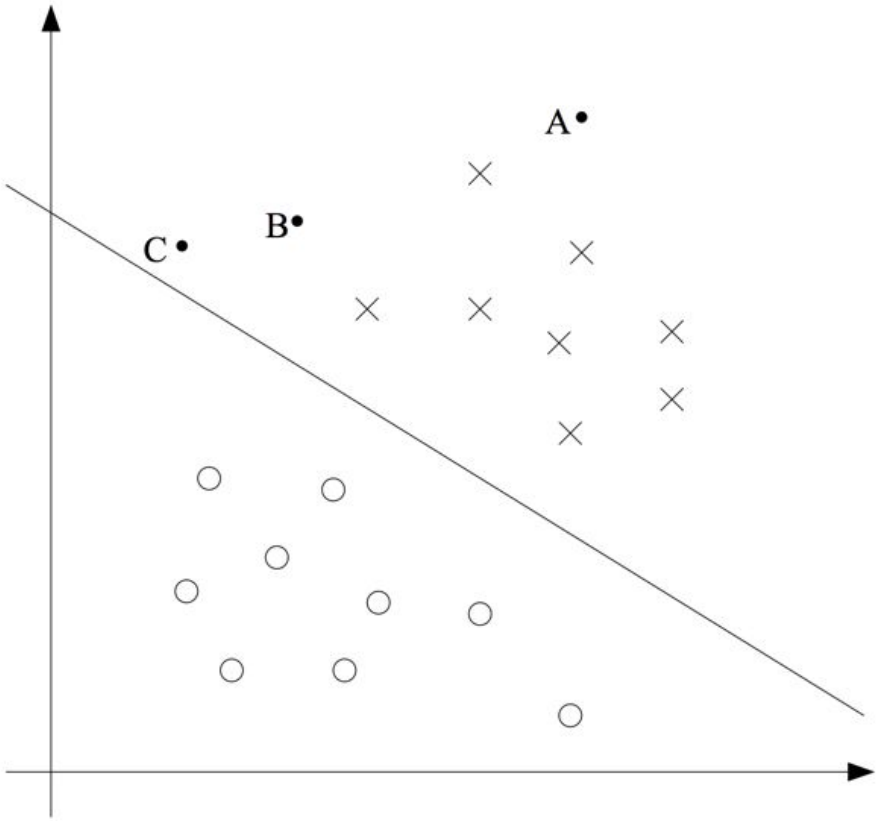
- Intuitively, our model is a good fit if we have learned a θ such that $\theta^T x^{(i)} \gg 0$ when $y^{(i)} = 1$ and $\theta^T x^{(i)} \ll 0$ when $y^{(i)} = 0$.

Geometric Margin Intuition

- Below, **x**'s represent positive training examples and **o**'s represent negative training examples, and the line is $\theta^T x = 0$ (the decision boundary, or separating hyperplane):



Geometric Margin Intuition



- Notice A is very far from the decision boundary - we should be very confident about our prediction for this point.
- However, C is very close - a small change to the boundary could change the prediction here.
- So we should be confident about our predictions for points far from the decision boundary.
- Intuitively, our model is a good fit the data is far from the learned decision boundary (high confidence).

Change of Notation

- We will be discussing a binary classification problem (now with $y \in \{-1, 1\}$) using a linear classifier with parameters w, b:

$$h_{w,b}(x) = g(w^T x + b)$$

- The g above is not the sigmoid function from before - it is any function satisfying

$$g(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$

- Notice that this classifier will directly predict either 1 or -1, unlike logistic regression which estimated to probability of y being 1.

Functional Margin

- Given a training example $(x^{(i)}, y^{(i)})$, the functional margin of (w, b) is

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

- If $y^{(i)} = 1$, then the functional margin is large when $w^T x^{(i)} + b$ is a large positive number.

- If $y^{(i)} = -1$, then the functional margin is large when $w^T x^{(i)} + b$ is a large negative number.

- Also, our prediction is correct if $y^{(i)}(w^T x^{(i)} + b) > 0$.

- So a large functional margin means a confident and correct prediction.

Functional Margin

- There is one big problem with the functional margin that makes it a poor measure of confidence!

- For any choice of g, if we replace **w** with **2w** and **b** with **2b**, then

$$g(w^T x + b) = g(2w^T + 2b)$$

and thus our prediction $h_{w,b}(x)$ would not change at all.

- But this means we can make the functional margin arbitrarily large!

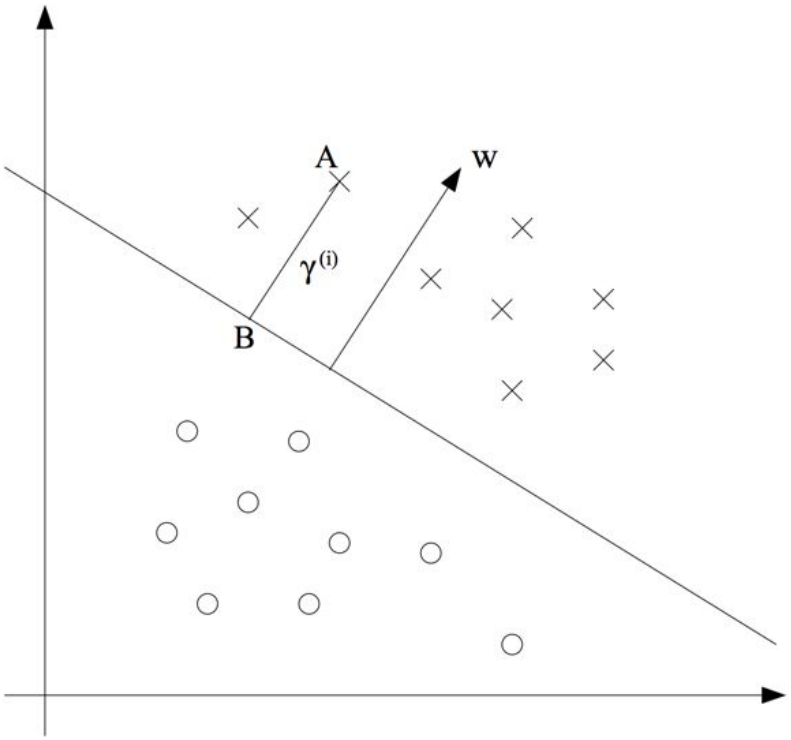
Functional Margin

- Maybe we should impose some normalization condition, like $||w||_2 = 1$ (replace (w, b) with $(w/||w||_2, b/||w||_2)$ when computing the functional margin).

- Given a training set $S = \{(x^{(i)}, y^{(i)}) : i = 1, ..., m\}$, we define the functional margin of the training set to be the smallest functional margin of each individual training example, ie,

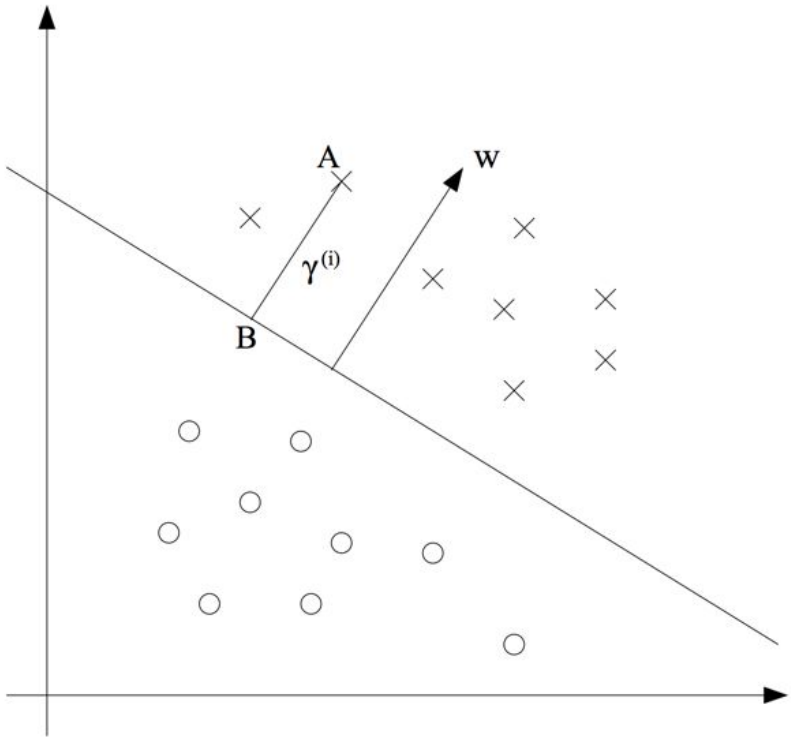
$$\hat{\gamma} = \min_{i=1,...,m} \hat{\gamma}^{(i)}$$

Geometric Margin



- The separating hyperplane corresponding to (w,b) is shown.
- The vector w is shown as well - it is orthogonal to the decision boundary - coincidence?
- A represents the input $x^{(i)}$ with label $y^{(i)} = 1$.
- The distance $\gamma^{(i)}$ to the decision boundary is the length of the line segment AB.
- How can we find $\gamma^{(i)}$?

Geometric Margin



- First note that

$$B = A - \gamma^{(i)} \frac{w}{||w||} = x^{(i)} - \gamma^{(i)} \frac{w}{||w||}$$

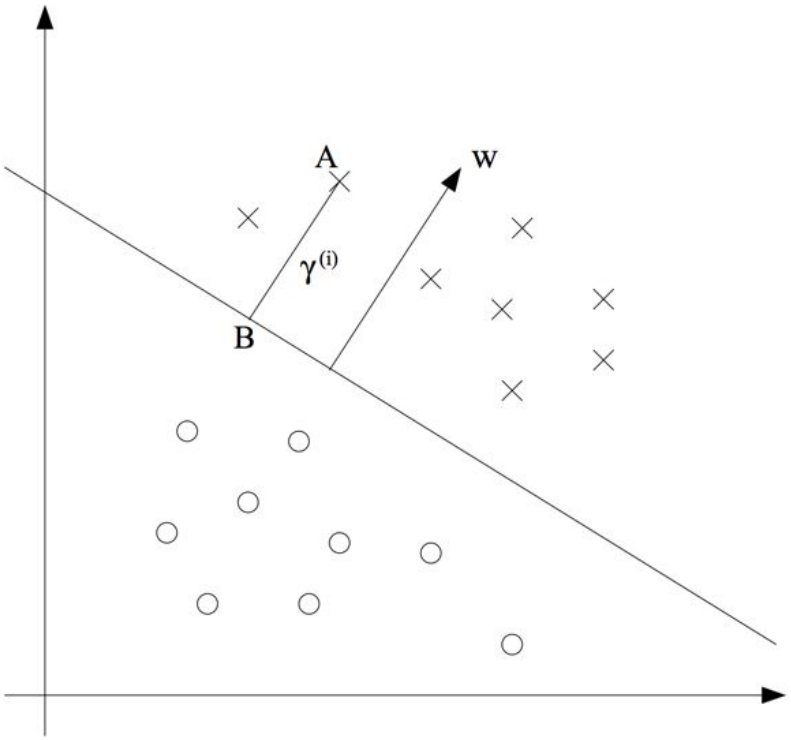
- But B lies on the decision boundary, so it satisfies $w^T B + b = 0$. Therefore

$$w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{||w||} \right) + b = 0$$

and solving for $\gamma^{(i)}$ yields

$$\gamma^{(i)} = \frac{w^T x^{(i)} + b}{||w||} = \left(\frac{w}{||w||} \right)^T x^{(i)} + \frac{b}{||w||}$$

Geometric Margin



- We can do the same thing for negative training examples to find the more general definition

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

- If $||w|| = 1$ then this is the same as the functional margin!
- Notice that this margin is invariant to scaling of the parameters.

Geometric Margin

- This means we can choose any scaling constraint without changing the value of the margin!
- Given a training set $S = \{(x^{(i)}, y^{(i)}) : i = 1, ..., m\}$, we define the functional margin of the training set to be the smallest functional margin of each individual training example, ie,

$$\hat{\gamma} = \min_{i=1,...,m} \hat{\gamma}^{(i)}$$

Maximal Margin Classifier

- Given a training set, we hope to find a decision boundary which maximizes the (geometric) margin, since this would imply a confident set of predictions and thus a good fit to the data.

- Suppose that our training set is linearly separable (we are able to separate the positive and negative examples using a hyperplane).

- How do we find the separating hyperplane which maximizes the geometric margin?

Maximal Margin Classifier

- Formally, the problem formulation becomes

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \ldots, m \\ & ||w|| = 1. \end{aligned}$$

- We want to maximize γ subject to every training example having functional margin greater than or equal to γ.

$$||w|| = 1$$

- Notice that ensures that the functional margin equals the geometric margin, so this optimization problem results in parameters **(w,b)** which maximize the geometric margin of the training set.

Maximal Margin Classifier

- But solving this problem is difficult due to the non-convex $||w|| = 1$ constraint - we cannot use any standard optimization software to solve the problem in its current form.

- We can reformulate this problem as

$$\max_{\hat{\gamma},w,b} \quad \frac{\hat{\gamma}}{||w||}$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, \quad i = 1, \ldots, m$$

- We want to maximize $\hat{\gamma}/||w||$ subject to every training example having functional margin greater than $\hat{\gamma}$.

- Since geometric and functional margins are related by $\gamma = \hat{\gamma}/||w||$, this yields the same result.

Maximal Margin Classifier

- But again solving this problem is difficult due to the non-convex objective function - still not standard software can solve the optimization problem in this form.

- Remember that we can add any constraint on **w** and **b** without changing the geometric margin!

- We will introduce the scaling constraint that the functional margin of **w,b** of the training set must be 1, ie:

$$\hat{\gamma} = 1$$

- Because multiplying **w** and **b** by some constant results in the functional margin multiplied by the same constant, this is just a scaling constraint - it can be satisfied by rescaling **w,b**.

Maximal Margin Classifier

- Plugging this into the reformulated problem above and noticing that maximizing $\hat{\gamma}/||w|| = 1/||w||$ is the same thing as minimizing $\frac{1}{2}||w||^2$, we have the optimization problem:

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m \end{aligned}$$

- This has a convex quadratic objective function and linear constraints - it can be efficiently solved using quadratic programming software!

- The solution yields the optimal (maximal) margin classifier.

Lagrange Duality

- We will not discuss the details of Lagrange duality - but it essentially allows us to reformulate this optimization problem in its dual form.

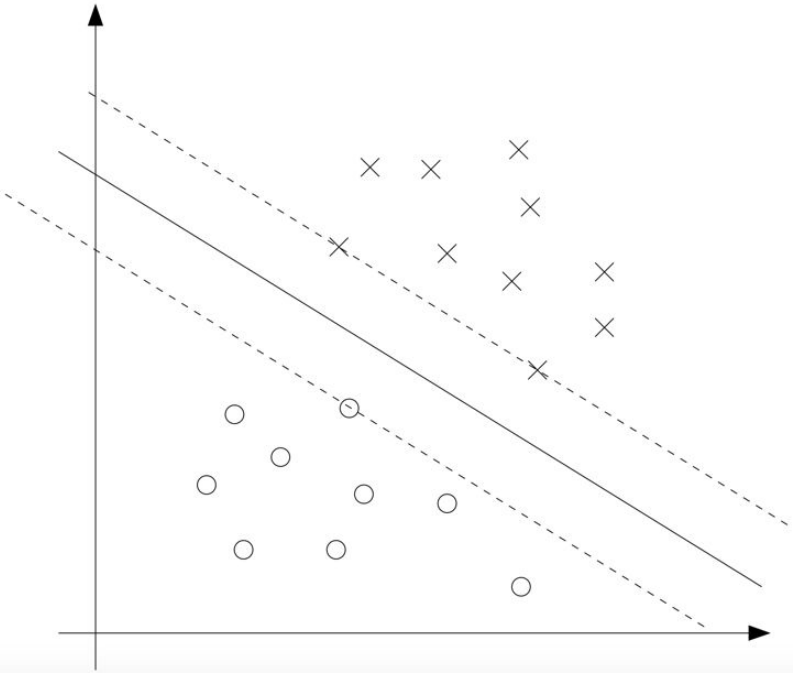
- Doing so will allow us to use kernels for efficiency in high dimensional spaces, as well as efficiency in general - much better than generic quadratic programming software.

- If you would like to learn more, the details can be found in Andrew Ng's CS229 notes!

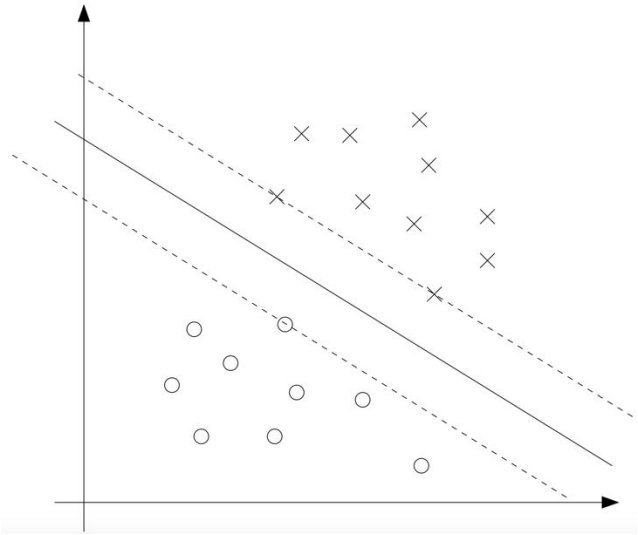
SVM Intuition

- Using Lagrange Duality, we can once again reformulate our optimization problem.

- Consider the figure below. The solid line is the maximum margin separating hyperplane:



SVM Intuition



- The three points closest to the decision boundary (on the dashed lines) are called the support vectors.
- Using Lagrange Duality, you can show that the number of support vectors can be much smaller than the training set.

SVM Intuition

- Again using Lagrange Duality, the solution to the following problem

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

  can be used to solve the original problem.

- Moreover, this problem is written only in terms of inner products between input feature vectors.

- We can exploit this property to apply kernels to the classification problem. The resulting algorithm, called support vector machines, allows for efficient learning in very high dimensional spaces.

Kernels

- Recall the initial linear regression problem.
- We had a few attributes given, like the number of ants x, the size of the house y, etc, and we're trying to make a prediction about the house.
- We could have used slightly different variations of our features instead - $x^2, x^3, \sin x, \ldots$ - and learned a much more complex function using least squares as before.

Kernels

- The original (raw) input is called the attributes, and the attributes mapped to a new set of quantities passed to the learning algorithm are called the features.

- Let ϕ denote the feature mapping, which maps from the attributes to the features.

- For example,

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \\ \sin(x) \end{bmatrix}$$

Kernels

- In any learning algorithm, rather than directly inputting our input attributes, we may want to instead learn using the features.

- We can do this easily by replacing the attributes x everywhere with $\phi(x)$.

- Since the SVM algorithm can be written entirely in terms of inner products, we can replace all of the inner product of attribute vectors with inner products of feature vectors.

Kernels

- We define the Kernel to be

$$K(x, z) = \langle \phi(x), \phi(z) \rangle = \phi(x)^T \phi(z)$$

- So we could replace $\langle x, z \rangle$ with $K(x, z)$ and the algorithm would learn using the features rather than the attributes.

- So given ϕ, we can compute $K(x, z)$ by finding $\phi(x)$ and $\phi(z)$ and taking their inner product.

- However, it is often very inexpensive to calculate $K(x, z)$ directly from the attributes when $\phi(x)$ may be very expensive to calculate (high-dim).

Kernels

- In these situations, by using an efficient way to calculate the kernel $K(x, z)$, the algorithm (SVM) can learn in high dimensional feature space (the range of ϕ), without ever explicitly finding or representing the vectors $\phi(x)$.

- Example: $x, z \in \mathbb{R}^n, \quad K(x, z) = (x^T z)^2$

$$
\begin{aligned}
K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\
&= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)
\end{aligned}
$$

Kernels

$$K(x,z) = \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)$$

- Thus we can write $K(x,z) = \phi(x)^T \phi(z)$ where ϕ is defined by (for n=3):

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

Kernels

- T

Kernels

- Intuitively, if $\phi(x)$ and $\phi(z)$ are close together, we might expect $K(x,z) = \phi(x)^T \phi(z)$ to be large.

- If they are ‘far” apart (say orthogonal) then $K(x,z) = \phi(x)^T \phi(z)$ will be small.

- So we can think of $K(x,z)$ as a similarity measure of $\phi(x)$ and $\phi(z)$ (or of x and z).

- Suppose we find some function $K(x,z)$ that we think is a good measure of similarity of x and z.

Kernels

- Maybe we choose

$$K(x,z) = e^{-\frac{||x-z||^2}{2\sigma^2}}$$

- This measure is close to *1* when *x* and *z* are close, and close to *0* when they are far apart.

- Can we use this *K* as the kernel in an SVM?

- Yes! It is called the Gaussian kernel, and corresponds to an infinite dimensional feature mapping ϕ.

Kernels

- How do we tell is a function *K* is a valid kernel (ie, that there exists some feature mapping ϕ such that $K(x,z) = \phi(x)^T \phi(z)$)?

- Suppose for now that *K* is a valid kernel, and consider some finite set of *m* points (not necessarily the training set) $\{x^{(1)}, ..., x^{(m)}\}$.

- Define a square *m*-by-*m* matrix *K* with $K_{ij} = K(x^{(i)}, x^{(j)})$. This is called the Kernel matrix.

Kernels

- If K is a valid kernel, then FIX

$$K_{ij} = K(x^{(i)}, x^{(j)}) = \langle x^{(i)}, x^{(j)} \rangle = \langle x^{(j)}, x^{(i)} \rangle = K(x^{(j)}, x^{(i)}) = K_{ji}$$

so K is symmetric.

- It can be easily shown that in fact K is positive semidefinite.

- In fact, this is also a sufficient condition:

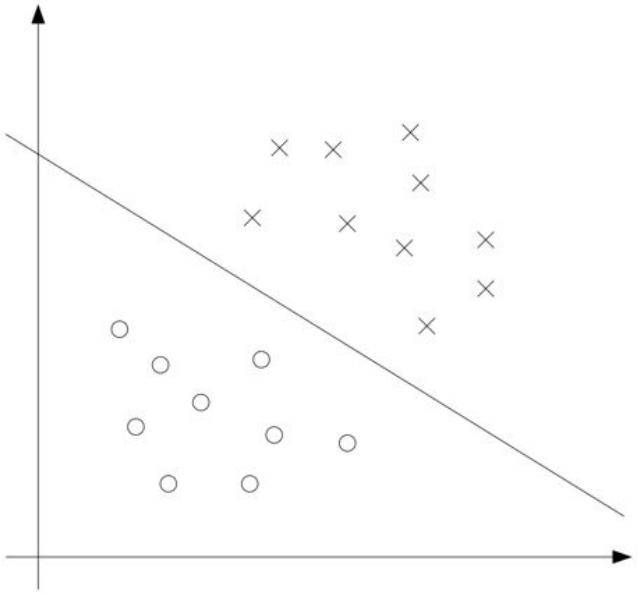
Theorem (Mercer). Let $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^{(1)}, \ldots, x^{(m)}\}$, $(m < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

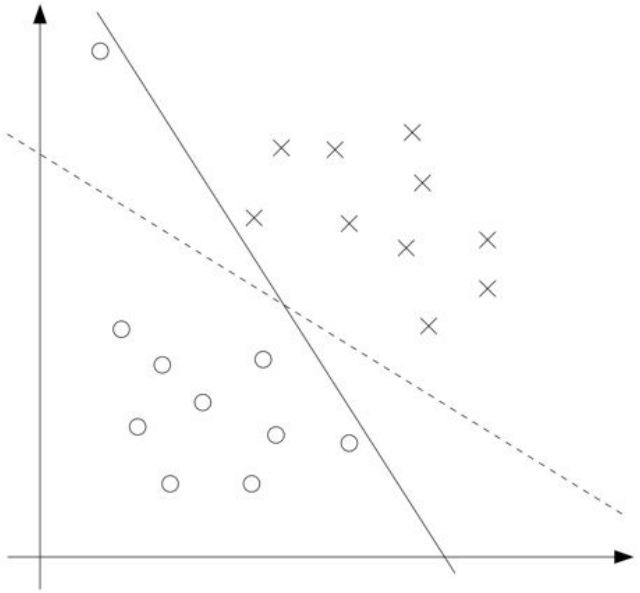
Regularization

- The SVM algorithm has thus far assumed the data is linearly separable.
- Mapping data to a high dimensional feature space via ϕ increases the likelihood the data is separable, but this is not always the case.
- In some cases it is not clear whether finding a separating hyperplane is what we want to do, since it may be susceptible to outliers.

Regularization

- For example,





- Here, the outlier causes the decision boundary to make a large rotation, causing the classifier to have a much smaller margin.

Regularization

- So to make the algorithm work for non-linearly separated datasets and simultaneously be less sensitive to outliers, we reformulate our optimization as:

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

Regularization

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m$$
$$\xi_i \geq 0, \quad i = 1, \ldots, m.$$

- The Csum term is called ℓ_1-regularization.

- Now, the training examples can have functional margin less than 1, and if one has functional margin $1 - \xi^i, \xi^i > 0$, then we pay the cost of the objective function increased by $C\xi_i$.

- *C* controls the weighting between making the $||w||^2$ term small and ensuring the examples have functional margin at least *1*.

Regularization

- Once again we can use Lagrange duality to reformulate the problem in terms of only inner products:

$$\begin{aligned}
\max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\
\text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \\
& \sum_{i=1}^{m} \alpha_i y^{(i)} = 0,
\end{aligned}$$

Coordinate Ascent

- Consider solving the *unconstrained* optimization problem:

$$\max_{\alpha} W(\alpha_1, \ldots, \alpha_m)$$

- We've seen gradient ascent as one optimization algorithm.

- Let's consider coordinate ascent:

Loop until convergence: {

For $i = 1, \ldots, m$, {

$\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$

}

}

Coordinate Ascent

Loop until convergence: {

For $i = 1, \ldots, m$, {

$\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$

}

}

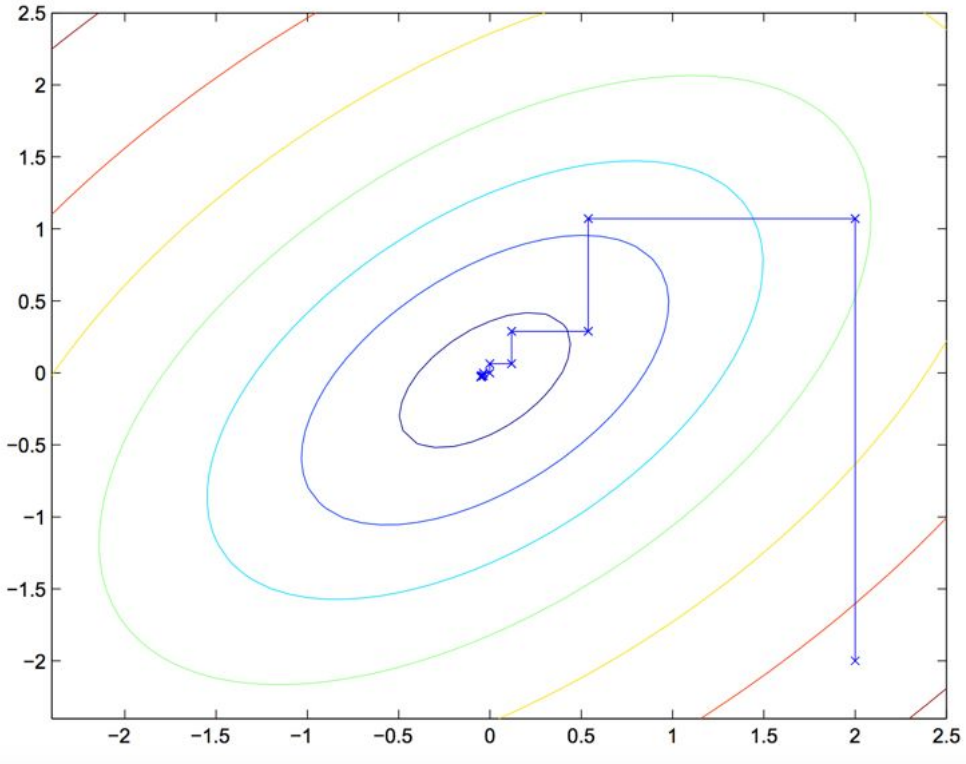
- In the innermost loop, we hold all variables except for one fixed, and reoptimize *W* with respect to that one variable.

- This algorithm reoptimizes the variables in order, but a more sophisticated algorithm may choose other orderings such as updating the variable which makes the largest increase in *W*.

Coordinate Ascent

- Coordinate ascent is fairly efficient when W is in a form that the 'argmax' in the inner loop can be performed efficiently.

- Example of coordinate ascent:



SMO Algorithm

- Recall the reformulation:

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

- Suppose we hold all but one variable fixed and reoptimize with respect to that one variable. Can we make progress?

- No! The last variables must be fixed as well:

$$\alpha_1 y^{(1)} = -\sum_{i=2}^{m} \alpha_i y^{(i)}$$

SMO Algorithm

- Thus to update some of the variables, we must update two simultaneously. This motivates the SMO algorithm:

Repeat till convergence {

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

}

- This is a very efficient algorithm because the update to the pair of variables can be computed very efficiently. See the notes.

What Just Happened?

- Maximal Margin Classifiers
- Kernels
- Regularization
- Coordinate Ascent and the SMO Algorithm

Naive Bayes

Jeremy Irvin and
Daniel Spokoyny

Andrew Ng Notes/ Stanford NLP

Instructions for use

Open this document in Google Slides (if you are at slidescarnival.com use the button below this presentation)

You have to be signed in to your Google account

EDIT IN GOOGLE SLIDES

Go to the ***File*** menu and select ***Make a copy***.

You will get a copy of this document on your Google Drive and will be able to edit, add or delete slides.

EDIT IN POWERPOINT®

Go to the ***File*** menu and select ***Download as Microsoft PowerPoint***. You will get a .pptx file that you can edit in PowerPoint.

Remember to download and install the fonts used in this presentation (you'll find the links to the font files needed in the Presentation design slide)

More info on how to use this template at www.slidescarnival.com/help-use-presentation-template

Hello!

I am Jayden Smith

I am here because I love to give presentations.

You can find me at:

@username

1.
Transition headline

Let’s start with the first set of slides

*Quotations are commonly printed as a **means of inspiration** and to invoke philosophical thoughts from the reader.*

This is a slide title

- Here you have a list of items
- And some text
- But remember not to overload your slides with content

You audience will listen to you or read the content, but won’t do both.

Big concept

Bring the attention of your audience over a key concept using icons or illustrations

You can also split your content

White

Is the color of milk and fresh snow, the color produced by the combination of all the colors of the visible spectrum.

Black

Is the color of coal, ebony, and of outer space. It is the darkest color, the result of the absence of or complete absorption of light.

In two or three columns

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

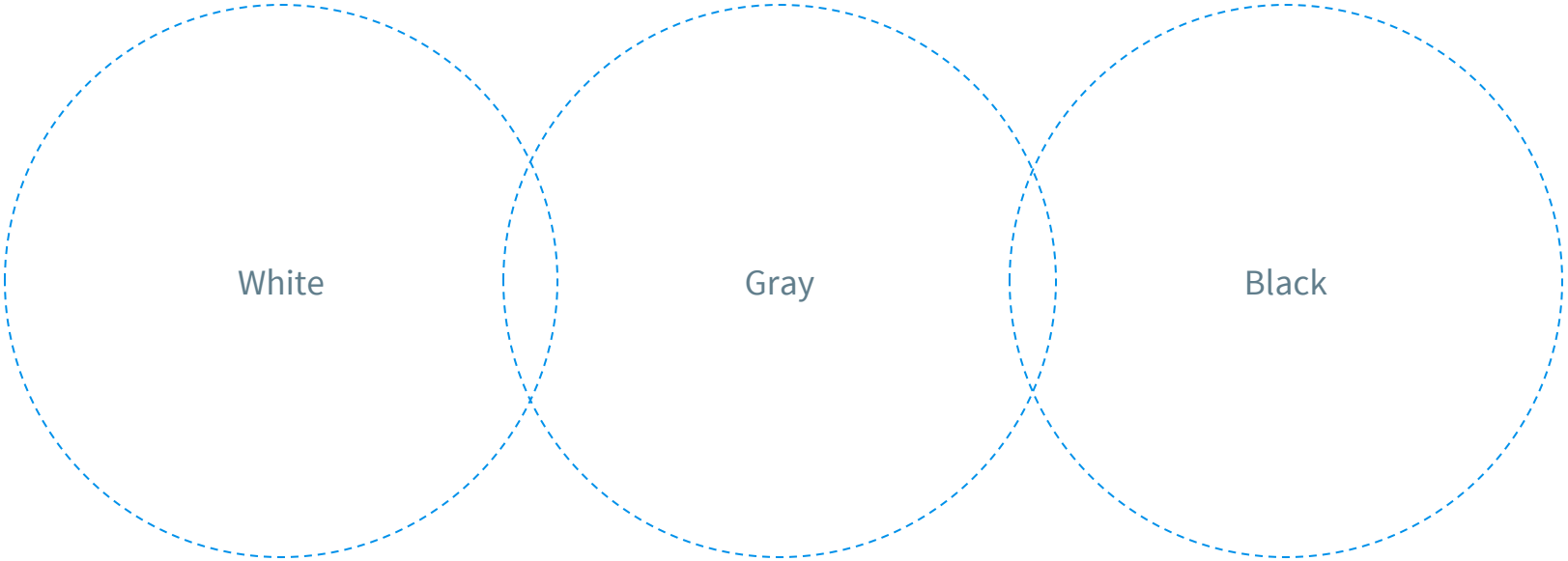
Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

A picture is worth a thousand words

A complex idea can be conveyed with just a single still image, namely making it possible to absorb large amounts of data quickly.


Want big impact?
Use big image.

Use charts to explain your ideas



And tables to compare data

	A	B	C
Yellow	**10**	**20**	**7**
Blue	**30**	**15**	**10**
Orange	**5**	**24**	**16**

Maps

89,526,124

Whoa! That’s a big number, aren’t you proud?

89,526,124$

That’s a lot of money

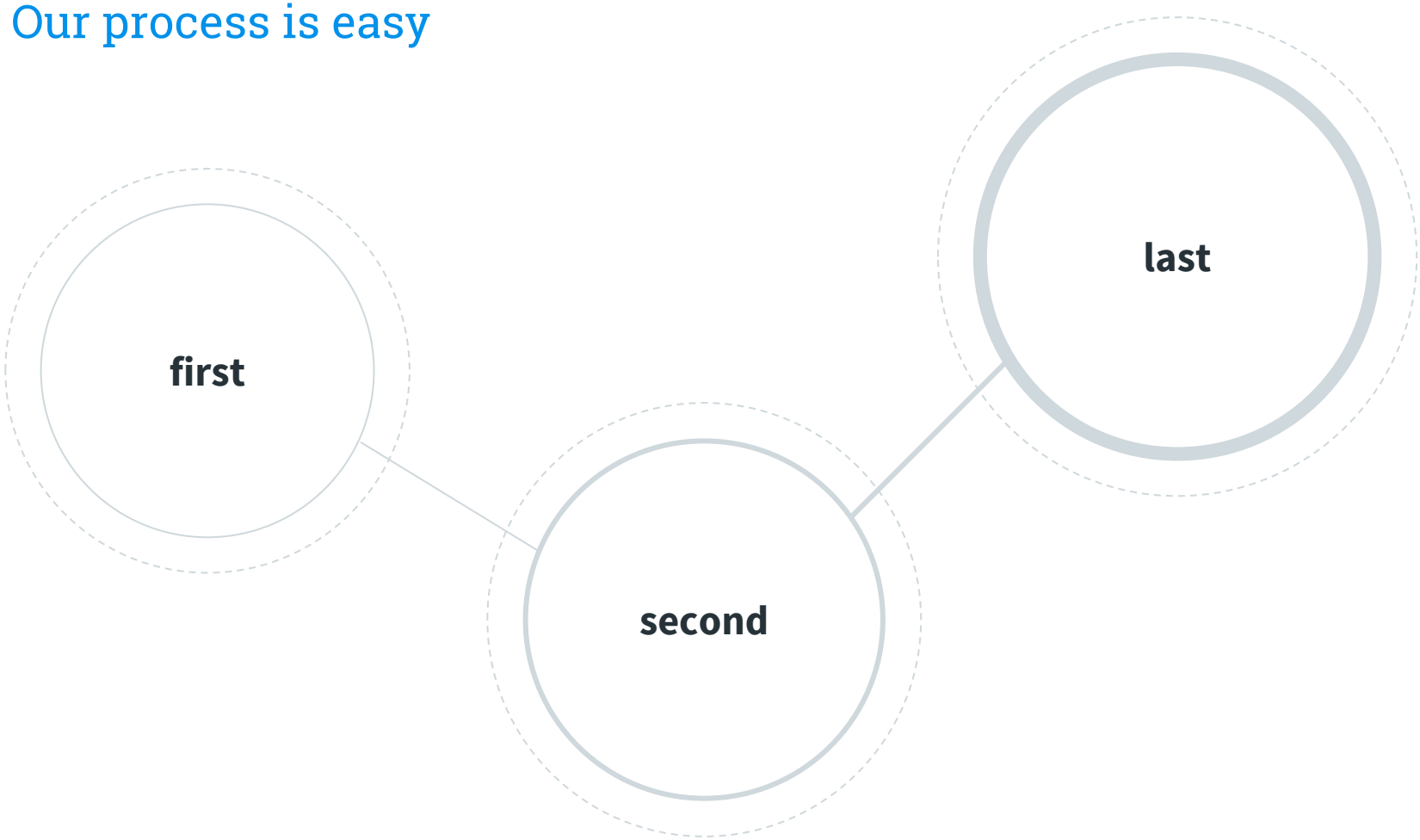
185,244 users

And a lot of users

100%

Total success!

Our process is easy



Let’s review some concepts

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

Yellow

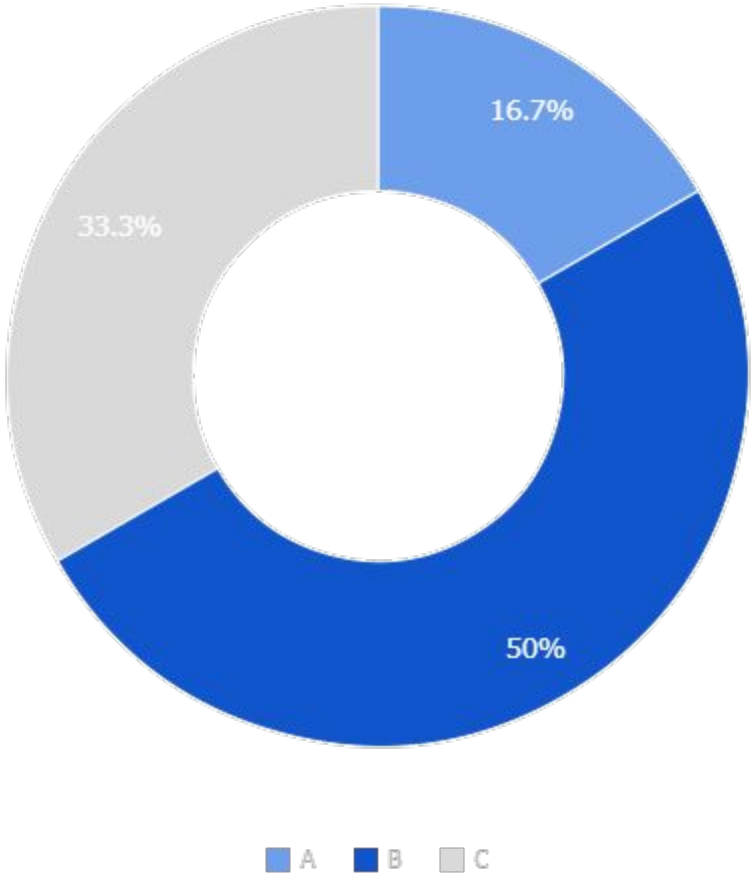
Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

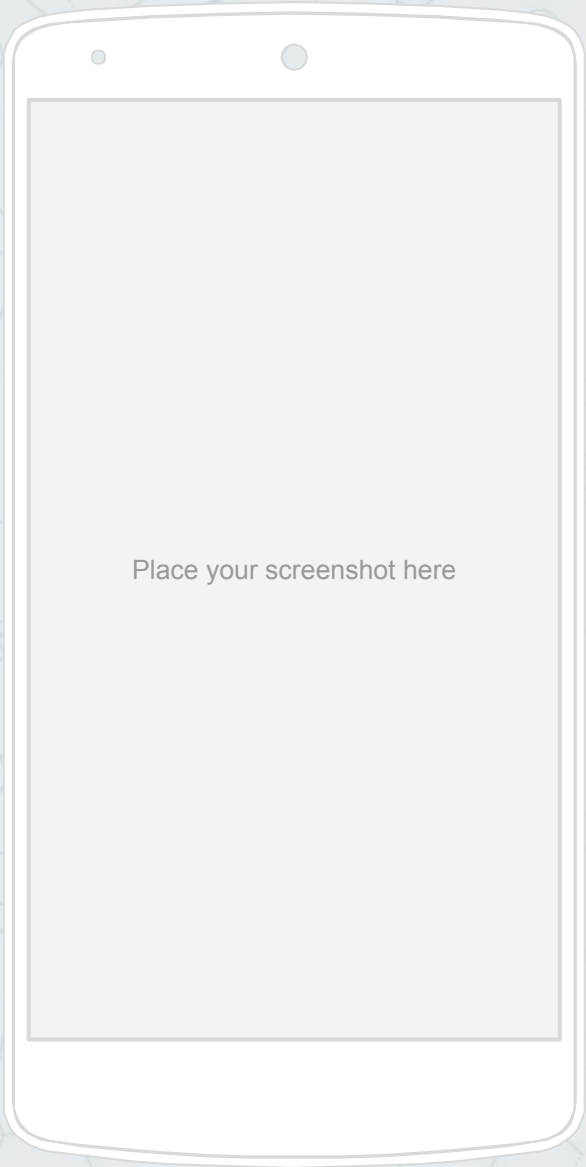
Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.



You can copy&paste graphs from Google Sheets

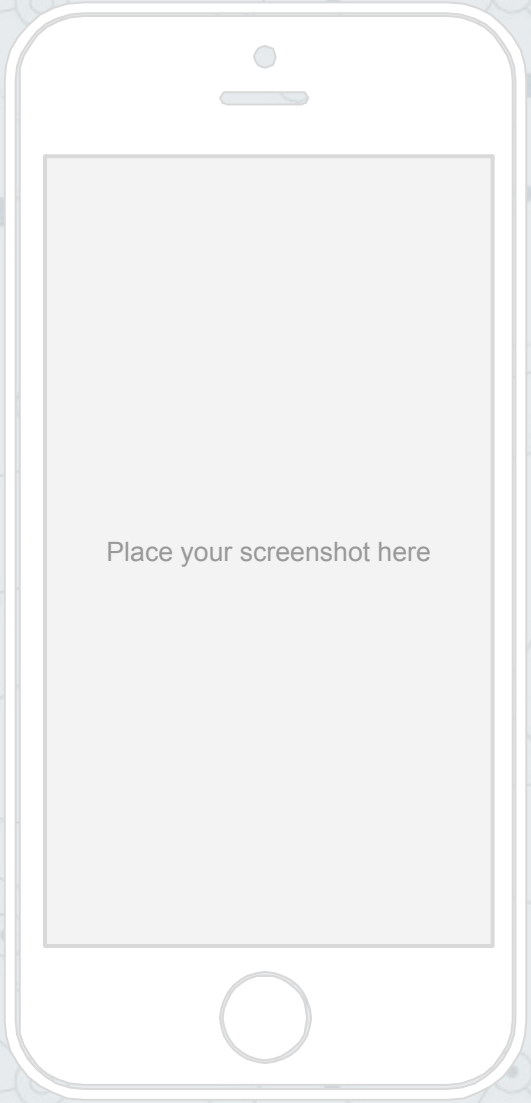
Android project

Show and explain your web, app or software projects using these gadget templates.



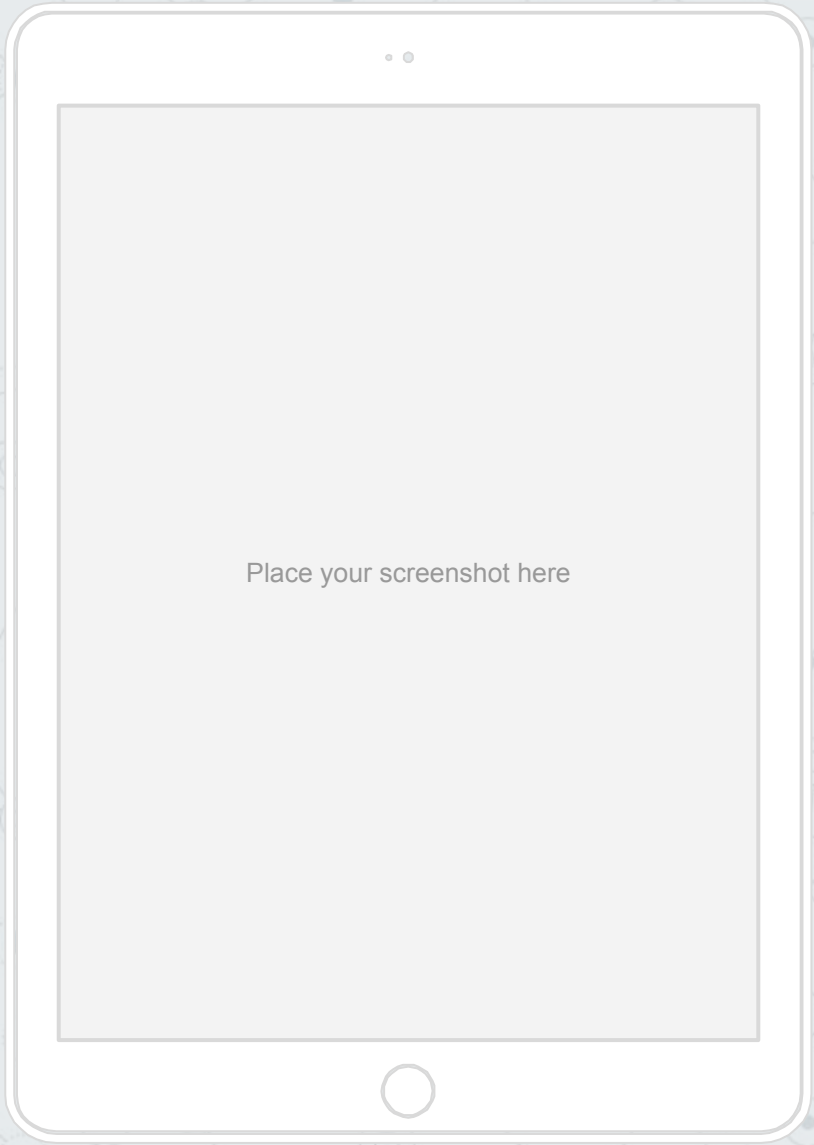
iPhone project

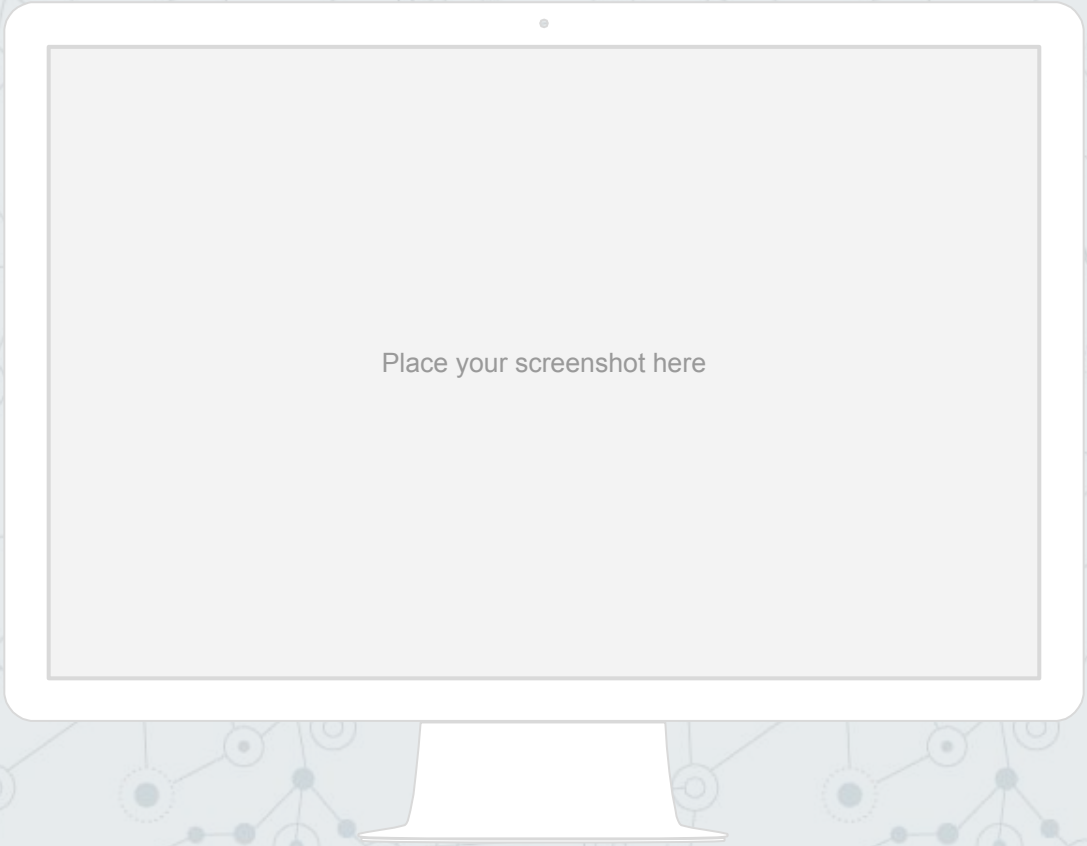
Show and explain your web, app or software projects using these gadget templates.



Tablet project

Show and explain your web, app or software projects using these gadget templates.





Desktop project

Show and explain your web, app or software projects using these gadget templates.

Thanks!

Any questions?

You can find me at:

@username

user@mail.me

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash & Death to the Stock Photo (license)

Presentation design

This presentations uses the following typographies and colors:

- Titles: **Roboto Slab**
- Body copy: **Source Sans Pro**

You can download the fonts on this page:

https://www.google.com/fonts#UsePlace:use/Collection:Source+Sans+Pro:400,700,400italic,700italic|Roboto+Slab:400,700

Click on the “arrow button” that appears on the top right 

- Blue **#0091ea**
- Dark gray **#263238**
- Medium gray **#607d8b**
- Light gray **#cfd8dc**

You don’t need to keep this slide in your presentation. It’s only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change line color, width and style.

Isn't that nice? :)

Examples: