CS 559: Machine Learning Fundamentals & Applications

Lecture 1: Linear Algebra and Probability Theory

In Jang: ijang@stevens.edu

Fall 2021

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Outline

- Course Information
- Python Review I – Starting Data Science with Python (see demonstration file)
- Python Review II – Visualization (see demonstration file)
- Linear Algebra Review
- Probability Theory

Course Information

- Lectures:
  - Lecture Video will be posted on Thursday Night
  - Office Hours – Friday 10 – 11 AM or by appointment
- Tech Requirement:
  - Python (v2.7 or higher)
  - Jupyter Notebook: https://jupyter.org/install

Python Review

- This review focuses on basic knowledge of python you must have to do Machine Learning and Data Science.
- Jupyter Notebook is a user-friendly computational environment for data scientists.
  - Stores work automatically
  - Markdown allows to export work in pdf, html, etc. for presentations, notes, etc.
- Why python?
  - It distinguishes itself from other languages by its flexibility, simplicity, and reliable tools.
  - It is consistent and is anchored on simplicity which makes it most appropriate for machine learning.
  - Fits the best on machine learning due to its independent platform and its popularity in the programming community.
  - We have other languages to do machine learning projects such as R and MATLAB.
    - R and MATLAB are more specialized in specific fields (statistics, mathematics, and bioinformatics)
    - Python is much convenient to do data mining and text analysis.

Linear Algebra Review

- Scalars, Vectors, and Matrices
- Adding Matrices and Vectors
- Multiplying Matrices and Vectors
- Identity and Inverse Matrices
- Linear Dependence and Span
- Norms
- Special Matrices and Vectors
- Matrix Representation
- Linear Operators
- Trace
- Determinant
- Eigenvalues and Eigenvectors
- Subspace

Scalars, Vectors, and Matrices

- **Scalars**: a single number. (x)
  - Let $s \in \mathbb{R}$ be the slope of the line: real-valued scalar
  - Let $n \in \mathbb{N}$ be the number of units: natural number scalar
- **Vectors**: an array of numbers that are arranged in order. ($\boldsymbol{x}$)
  - Elements of vectors: typically written as x_i where x is scalar and the subscript i is the order.
  - If each element is in $\mathbb{R}$ and the vector has n elements, then the vector lies in $\mathbb{R}^n$
  - $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ or $\boldsymbol{x} = [x_1 \quad \cdots \quad x_n]$
- **Matrix**: a 2-D array of numbers. ($\boldsymbol{A}$)
  - $\boldsymbol{A} \in \mathbb{R}^{m \times n}$: A real-valued matrix $\boldsymbol{A}$ has a height of m and a width of n
  - $\boldsymbol{A} = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}$
  - Vectors can be considered as matrices that contain only one column.

Scalars, Vectors, and Matrices

- **Transpose**: The mirror image of the matrix across a diagonal line, called the main diagonal, running down and to the right, staring from its upper left corner.
  - $\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}^T = \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix}$
  - $\boldsymbol{A}_{i,j}^T = \boldsymbol{A}_{j,i}$
  - If $\boldsymbol{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$, then its transpose $\boldsymbol{a}^T$ is $\boldsymbol{a}^T = [a_1 \quad \cdots \quad a_n]$

$$A\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad A^T\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$A\begin{bmatrix} 1 & 4 & 3 \\ 8 & 2 & 6 \\ 7 & 8 & 3 \\ 4 & 9 & 6 \\ 7 & 8 & 1 \end{bmatrix} \qquad A^T\begin{bmatrix} 1 & 8 & 7 & 4 & 7 \\ 4 & 2 & 8 & 9 & 8 \\ 3 & 6 & 3 & 6 & 1 \end{bmatrix}$$

Adding Matrices and Vectors

- Addition between matrices?
  - As long as the shapes are the same,
    - $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$ where $C_{i,j} = A_{i,j} + B_{i,j}$
- A scalar to Matrix?
  - Perform the operations to each element
    - $\boldsymbol{D} = a \cdot \boldsymbol{B} + c$ where $D_{i,j} = aB_{i,j} + c$
- A vector to Matrix?
  - Yields another matrix
    - $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{b}$ where $C_{i,j} = A_{i,j} + b_j$
    - Copying of **b** to many locations is called **broadcasting.**

The vector b is added to each row of the matrix.

Multiplying Matrices and Vectors

- Multiplication (**Matrix product**) between matrices?
  - **A** must have the same number of columns as **B** has rows
    - Let **A** be $m \times n$ and **B** be $n \times p$, then $\boldsymbol{C} = \boldsymbol{AB}$ is a matrix with shape of $m \times p$
    - $\boldsymbol{C} = \boldsymbol{AB}$ where $C_{i,j} = \sum_k A_{i,k} B_{k,j}$
    - The product of individual elements is called **element-wise product**, or **Hadamard product** and is denoted as $\boldsymbol{A} \odot \boldsymbol{B}$.
- The dot product between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
  - $\boldsymbol{x}$ and $\boldsymbol{y}$ must have the same dimension
  - Also can be considered as matrix product with the same dimensionality
  - $\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^T \boldsymbol{y}$

Multiplying Matrices and Vectors

$$\begin{bmatrix} -4 & 4 & 6 \\ 2 & 3 & -1 \end{bmatrix} \begin{bmatrix} 0 & -4 & -2 & 5 \\ 1 & 6 & -1 & 7 \\ 8 & 2 & 4 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} -4\times0 + 4\times1 + 6\times8 & -4\times-4 + 4\times6 + 6\times2 & -4\times-2 + 4\times-1 + 6\times4 & -4\times5 + 4\times7 + 6\times3 \\ 2\times0 + 3\times1 + -1\times8 & 2\times-4 + 3\times6 + -1\times2 & 2\times-2 + 3\times-1 + -1\times4 & 2\times5 + 3\times7 + -1\times3 \end{bmatrix}$$

$$= \begin{bmatrix} 52 & 52 & 28 & 26 \\ -5 & 8 & -11 & 28 \end{bmatrix}$$

Multiplying Matrices and Vectors

- **Distributive property:**
  - $\boldsymbol{A(B+C) = AB + AC}$
- **Associative property:**
  - $\boldsymbol{A(BC) = (AB)C}$
- **But not commutative!**
  - $\boldsymbol{AB \neq BA}$
  - but in dot product between two vectors, we have $\boldsymbol{x^T y = y^T x}$.

Identity and Inverse Matrices

Matrix inversion: Enables to analytically solve $\boldsymbol{Ax} = \boldsymbol{b}$ for many values of $\boldsymbol{A}$.

- An identity matrix: a matrix that does not change any vector when we multiply that vector by that matrix. It is denoted as $\boldsymbol{I_n}$ that is $\in \mathbb{R}^{n \times n}$ and for $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{I_n x} = \boldsymbol{x}$.
  - All the entries along the main diagonal are 1 and other entries are zero.
- The matrix inverse of $\boldsymbol{A}$ is denoted as $\boldsymbol{A^{-1}}$ and it is defined as the matrix such that

$$\boldsymbol{A^{-1}A} = \boldsymbol{I_n}$$

If we have $\boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{x} = \boldsymbol{A^{-1}b}$.

- $\boldsymbol{A^{-1}}$ is primarily useful as a theoretical tool but *should not be used* in practice for most software applications.
- $\boldsymbol{A^{-1}}$ can be represented with only limited precision on a digital computer and algorithms that make use of the value of $\mathbf{b}$ can be usually obtain more accurate estimates of $\boldsymbol{x}$.

The eight axioms

Associativity of addition	$\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
Commutativity of addition	$\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
Identity element of addition	There exists an element $0 \in \mathbf{V}$, called the zero vector, such that $\mathbf{v} + 0 = \mathbf{v}$ for all $\mathbf{v} \in V$.
Inverse elements of addition	For every $v \in V$, there exists an element $-\mathbf{v} \in V$, called the additive ***inverse*** of $\mathbf{v}$, such that $v + (-v) = 0$.
Compatibility of scalar multiplication with field multiplication	$a(b\mathbf{v}) = (ab)\mathbf{v}$
Identity element of scalar multiplication	$1v = v$, where 1 denotes the multiplicative identity in F.
Distributivity of scalar multiplication with respect to vector addition	$a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$
Distributivity of scalar multiplication with respect to field addition	$(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$

Linear Dependence and Span

Recall $\boldsymbol{Ax} = \boldsymbol{b} \rightarrow \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$

- One solution for every value of b but also can have no solutions or infinitely many solutions
- It is not possible to have more than one but less than infinitely many solutions for particular b.
- If both x and y are solutions, then

$$\boldsymbol{z} = \alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y}$$

Linear Dependence and Span

The columns of **A** specify directions from the origin

- Determines *how many ways* there are of reaching $\boldsymbol{b}$.
- **Linear combination**

$$\boldsymbol{Ax} = \sum_i x_i A_{:,i}$$

- **Span** – set of all points obtained by linear combination of the original vectors.
- Determining whether $\boldsymbol{Ax} = \boldsymbol{b}$ has a solution is testing whether $\boldsymbol{b}$ is in the *span of the columns* of $\boldsymbol{A}$.
  - The column space of $\boldsymbol{A}$ be all of $\mathbb{R}^m$ for the system to have solution for all values of $\boldsymbol{b} \in \mathbb{R}^m$!
  - $\boldsymbol{A}$ must have at least m columns!

Linear Dependence and Span

Consider A have columns n and b has m elements where $n \geq m$,

- If A is a 3×2 matrix and x is 2×1, the target b is a 3×1.
- The target $\boldsymbol{b}$ is in 3-D but x is only 2-D.
- In this case, the solution is only when $\boldsymbol{b}$ lies on that plane.
- What happens if columns are identical?
  - Wil fail to encompass all of $\mathbb{R}^m$
  - **Linear independent** – if no vector in the set is a linear combination of the other vectors.
  - For the column space of the matrix to encompass all of $\mathbb{R}^m$, the matrix must contain at least one set of m linearly intendent columns.
  - This requirement is to have exactly m linear independent columns, not at least m.
  - No set of m-D vectors can have more than m mutually independent columns, but a matrix with more than m columns may have more than one such set.

Linear Dependence and Span

- $\boldsymbol{Ax = b \rightarrow x = A^{-1}b}$

For A to have an inverse:

- Linear independency
- Must have at most one solution for each value of b → at most m columns
  - Otherwise, there is more than one way of parameterizing each solution.

- In other words… A must be **square** matrix by being $m \times m$ and all columns be **linearly independent**.
- A square matrix with linearly dependent columns is known as **singular.**

- For non-square or a square but singular **A?**
  - **We can solve using other than the method of matrix inversion.**

Norms

Norm – In ML, we measure the size of vectors by mapping vectors to non-negative values.

$$||x||_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

for $p \in \mathbb{R}, p \geq 1$.
Also the norm of vector x measures the distance from the origin to the point *x*.

Matrix Representation

- Let $\{\mathbf{v}_1,\ldots,\mathbf{v}_k\}$ be a basis
- Every $\mathbf{v}$ can be uniquely represented as:

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \cdots + \alpha_k\mathbf{v}_k$$

- Denote $\mathbf{v}$ by the column-vector: $\mathbf{v} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_k \end{pmatrix}$
- Denote the basis vectors as: $\begin{pmatrix} \mathbf{1} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \\ \vdots \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{1} \end{pmatrix}$

Linear Operators

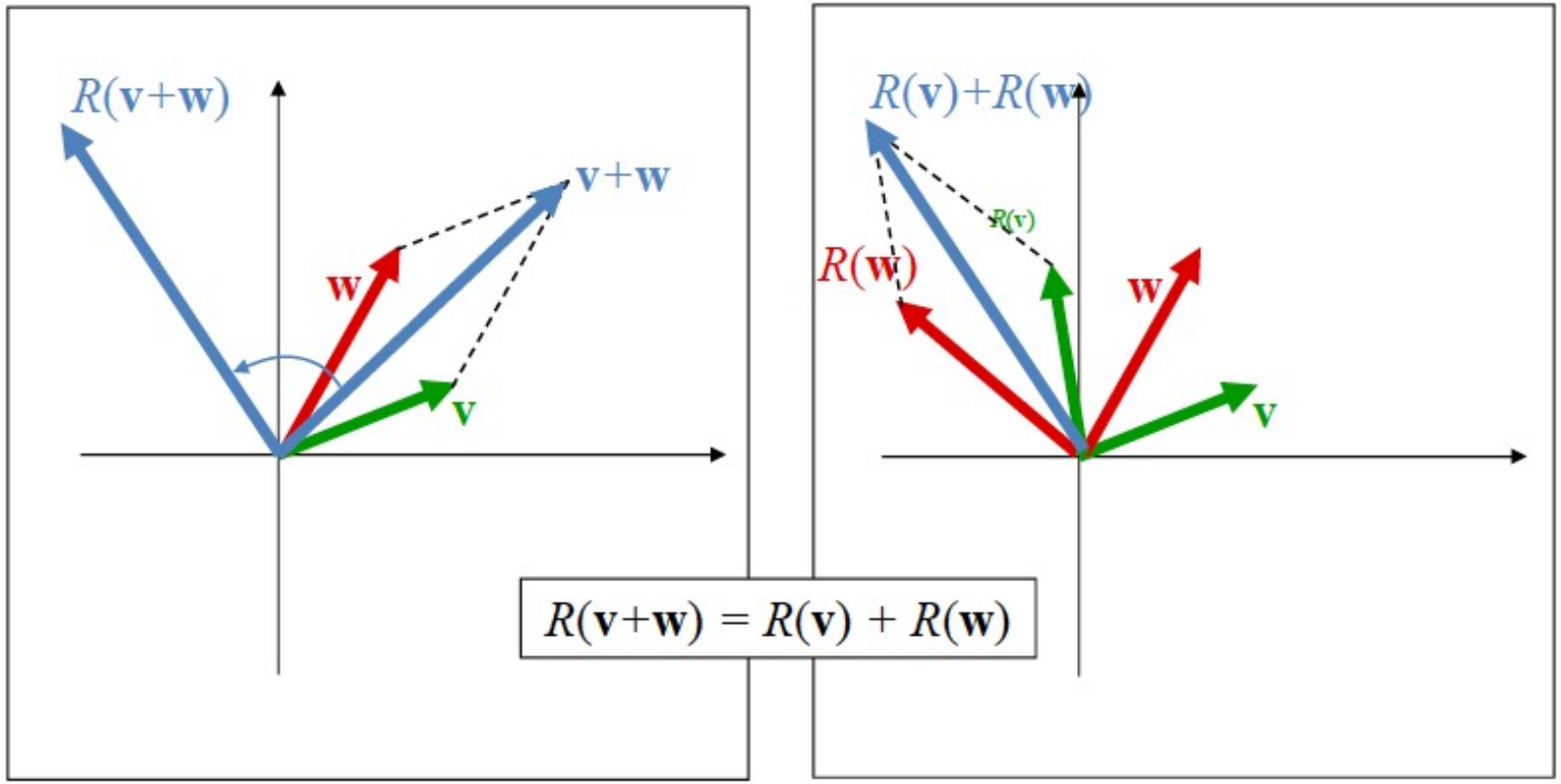
- $A : \mathrm{V} \rightarrow \mathrm{W}$ is called linear operator if:
  - $A(\mathbf{v} + \mathbf{w}) = A(\mathbf{v}) + A(\mathbf{w})$
  - $A(\alpha\mathbf{v}) = \alpha A(\mathbf{v})$

- In particular, $A(\mathbf{0}) = 0$

- Are the following operators linear?
  - Scaling
  - Rotation
  - Translation

Linear operators - illustration

- Rotation is a linear operator:



Matrix properties

- Matrix A ($n \times n$) is **non-singular** if $\exists B,\ AB = BA = I$
- $B = A^{-1}$ is called the **inverse** of A
- A is non-singular $\Leftrightarrow$ $det\ A \neq 0$

- If A is non-singular then the equation
  - $A\mathbf{x}=\mathbf{b}$ has one unique solution for each $\mathbf{b}$
  - the rows of A are linearly independent (and so are the columns)

Orthogonal matrices

- Matrix A $(n \times n)$ is orthogonal if $A^{-1} = A^T$
- Follows: $AA^T = A^TA = I$
- The rows of A are orthonormal vectors!

  Proof:

$$I = A^TA = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \\ \mathbf{v}_n \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & & \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i^T\mathbf{v}_j \end{pmatrix} = \begin{pmatrix} \delta_{ij} \end{pmatrix}$$

$$\Rightarrow \langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1 \quad \Rightarrow \quad \|\mathbf{v}_i\| = 1; \quad \langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$$

Delta function δ_{ij}:

$$\delta_{ij} = \begin{cases} 1 \; if \; i = j \\ 0 \; else \end{cases}$$

Trace

- The trace of a square matrix denoted by tr(A) is sum of the diagonal elements

$$tr(A) = \sum_{i=1}^{N} A_{ii}$$

- $tr\left(\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}\right) = A_{11} + A_{22} = 1 + 1 = 2$

Determinant

- For a square matrix A, the determinant is denoted by $|A|$ or $\det(A)$

$$
\begin{aligned}
\left|\left[a_{11}\right]\right| &= a_{11} \\
\left|\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right| &= a_{11}a_{22} - a_{12}a_{21} \\
\left|\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right| &= \begin{aligned}[t] & a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} \\ & -a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31} \end{aligned}
\end{aligned}
$$

Determinant

- $|A| = |A^{T}|$
- $|AB| = |A|\,|B|$
- $|A| = 0$, if and only if A is singular
  - Else, $|A^{-1}| = 1/|A|$

Eigenvalues and Eigenvectors

- For an $n \times n$ **square** matrix A, $\boldsymbol{\mu}$ is an eigenvector with eigenvalue λ if
$$\boldsymbol{A\mu} = \lambda\boldsymbol{\mu}$$
- that is
$$(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{\mu} = 0$$
- If $(\boldsymbol{A} - \lambda\boldsymbol{I})$ is invertible, then the only solution is $\boldsymbol{\mu} = 0$. (trivial)
- For non-trivial solutions:
$$\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0$$
- Above equation is called the "characteristic polynomial"
- Solutions are not unique:
  - If $\boldsymbol{\mu}$ is an eigenvector $\alpha\boldsymbol{\mu}$ is also an eigenvector where α is any constant.

Eigenvalues and Eigenvectors

- For a 2×2 matrix

$$\det[\mathbf{A}-\lambda\mathbf{I}]=\begin{vmatrix} a_{11}-\lambda & a_{12} \\ a_{21} & a_{22}-\lambda \end{vmatrix}=(a_{11}-\lambda)(a_{22}-\lambda)-a_{12}a_{21}=0$$

$$0=a_{11}a_{22}-a_{12}a_{21}-\lambda(a_{11}+a_{22})+\lambda^2$$

If $\boldsymbol{A}=\begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix}$, we get

$$0=1\cdot 4-2\cdot 2-(1+4)\lambda+\lambda^2$$

$$(1+4)\lambda=\lambda^2$$

- The solutions are λ=0 and λ=5.

Eigenvalues and Eigenvectors

- The eigenvector for the first eigenvalue, λ=0 is:

$$\mathbf{Ax} = \lambda\mathbf{x}, \quad (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$$

$$\left[\begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix} - \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}\right] \cdot \begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix} \cdot \begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}1x+2y\\ 2x+4y\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix}$$

- One solution for both equations is $x = 2, y = -1$.
- The second eigenvalue is λ=5

$$\left[\begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix} - \begin{bmatrix}5 & 0\\ 0 & 5\end{bmatrix}\right] \cdot \begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}-4 & 2\\ 2 & -1\end{bmatrix} \cdot \begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}-4x+2y\\ 2x-1y\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix}$$

- A solution is $x = 1, y = 2$.

Properties

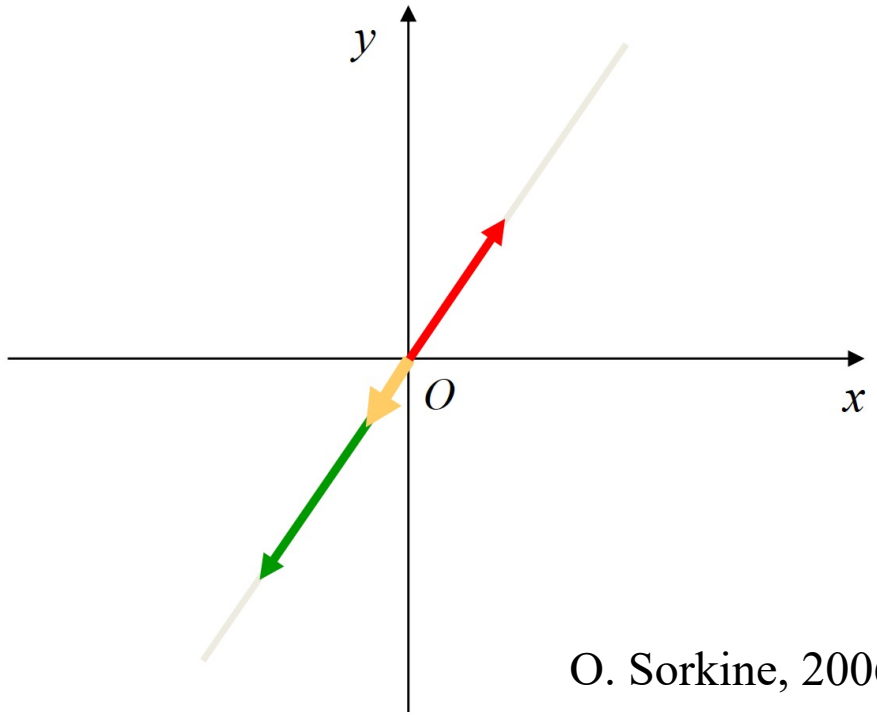
- The product of the eigenvalues = |A|
  - $\lambda_1\lambda_2 = 0 \cdot 5 = 0 = (1 \cdot 4 - 2 \cdot 2) = 0$
- The sum of the eigenvalues = trace(A)
  - $\lambda_1 + \lambda_2 = 0 + 5 = 5 = A_{11} + A_{22} = 1 + 4 = 5$
- The eigenvectors are pairwise orthogonal
  - $\begin{bmatrix} 2 \\ -1 \end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix} 1 \\ 2 \end{bmatrix} = (2 \cdot 1 - 1 \cdot 2) = 0$

Subspace

- Let F be a field, V be a vector space over F, and let W be a subset of V. Then W is a **subspace** if:
  - The zero vector, $\mathbf{0}$, is in W.
  - If $\mathbf{u}$ and $\mathbf{v}$ are elements of W, then the sum $\mathbf{u} + \mathbf{v}$ is an element of W.
  - If $\mathbf{u}$ is an element of W and c is a scalar from K, then the scalar product $c\mathbf{u}$ is an element of W.

Subspace Example

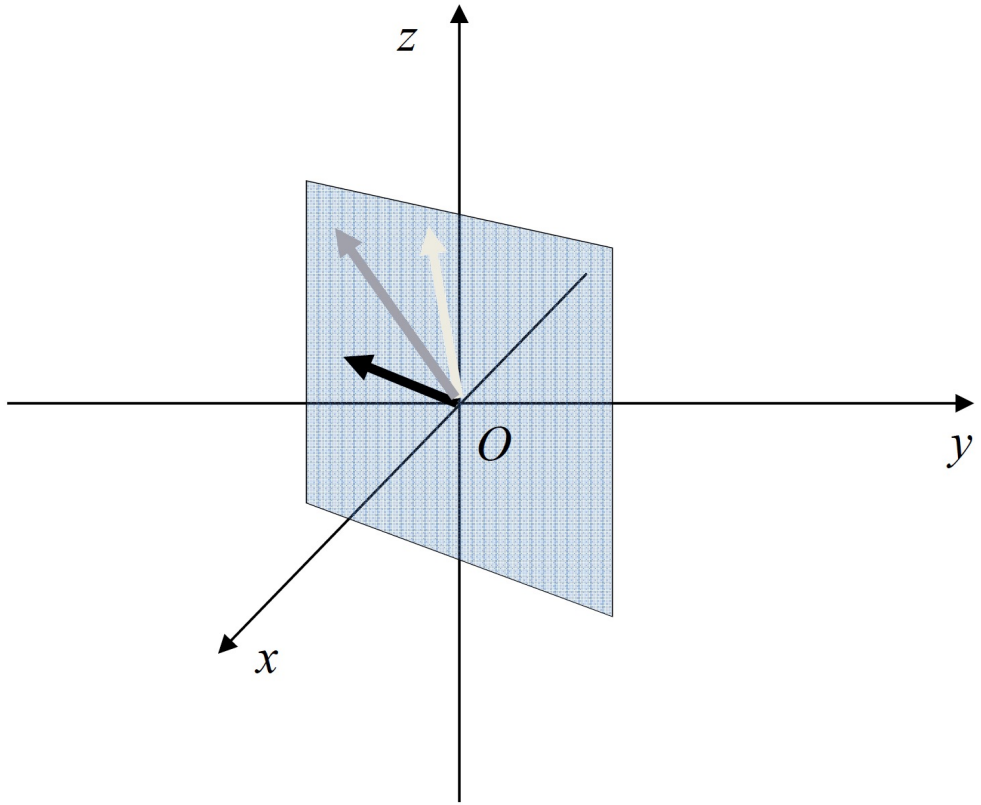
- Let l be a 2D line though the origin
- $L = \{\mathbf{p} - O \mid \mathbf{p} \in l\}$ is a linear subspace of R^2



O. Sorkine, 2006

Subspace Example

- Let π be a plane through the origin in 3D
- $V = \{\mathbf{p} - O \mid \mathbf{p} \in \pi\}$ is a linear subspace of R^3



Linear Independence

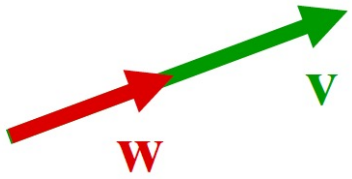
- The vectors $\{\mathbf{v}_1,\ldots,\mathbf{v}_k\}$ are a linearly independent set if:

$$\alpha_1\mathbf{v}_1 + \cdots + \alpha_k\mathbf{v}_k = \mathbf{0} \Leftrightarrow \alpha_i = 0\ \forall i$$

- It means that none of the vectors can be obtained as a linear combination of the others.

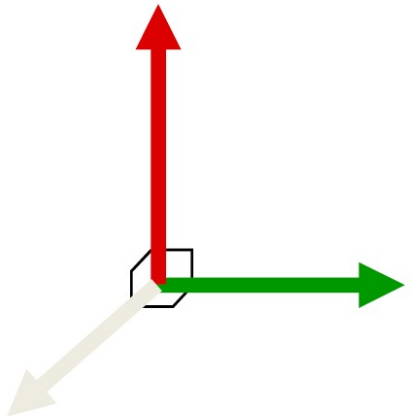
Linear independence - example

- Parallel vectors are always dependent:



$$\mathbf{v} = 2.4\,\mathbf{w} \quad \Rightarrow \quad \mathbf{v} + (-2.4)\mathbf{w} = 0$$

- Orthogonal vectors are always linearly independent:



Basis of Vector Space

- $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ are **linear independent**
- $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ **span** the whole vector space $\boldsymbol{V}$

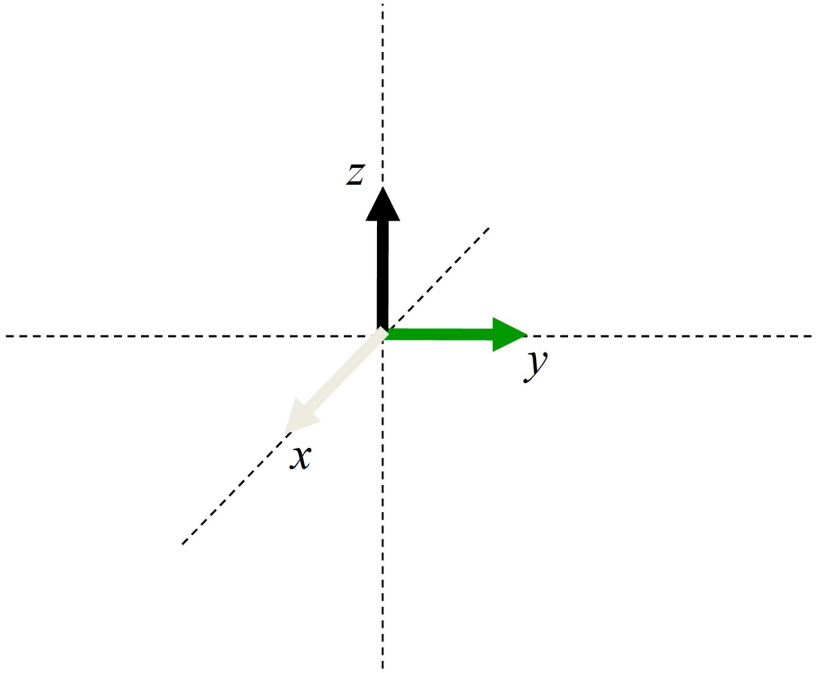
$$\mathbf{V} = \{\alpha_1 \mathbf{v}_1 + \cdots + \alpha_k \mathbf{v}_k | \alpha_1 \in R\}$$

- Any vector in $\mathbf{V}$ is a **unique** linear combination of the basis

- The number of basis vectors is called the **dimension** of $\mathbf{V}$

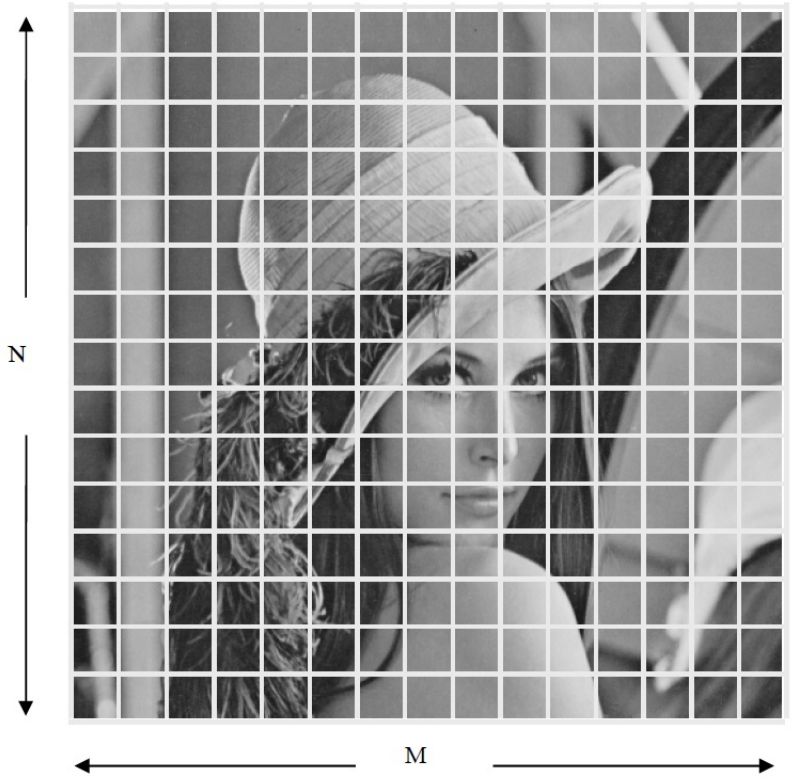
Basis Example

- The standard basis of $\mathbb{R}^3$



Basis – Another Example

- Grayscale $N \times M$ images:
  - Each pixel has value between 0 (black) and 1 (white)
  - The image can be interpreted as a vector $\in \mathbb{R}^2$



Probability Theory

- Axioms of Probability
- Discrete Random Variables
- Continuous Random Variables
- The Univariate Gaussian
- Maximum-likelihood Estimator

The Axioms of Probability

- P(A) = # of event/# of possible event
- 0 <= P(A) <= 1
- P(True) = 1 & P(False) = 0
- P(A or B) = P(A) + P(B) – P(A and B)

Discrete Random Variables

- Sample space Ω: Possible "states" x of the random variable X (Outcomes of the experiment, output of the system, measurement)
- Either have a finite or countable number of states.
- Events: Possible combination of states ('subsets of Ω')

Probability Mass Functions (PMF)

A function which tells us how likely each possible outcome is:

$$P(X = x) = P_x(x) = P(x)$$

$$\sum_{x \in \Omega} P(x) = 1$$

$$P(A) = P(x \in A) = \sum_{x \in A} P(X = x)$$

Expectation and Variance

- Expectation (or mean):

$$E(x) = \sum_x P(X = x)x$$

- Expectation of a function:

$$E\big(f(x)\big) = \sum_x P(X = x)f(x)$$

- Moments = expectation of power of X:

$$M_k = E(X^k)$$

Expectation and Variance

- Variance: Average (squared) fluctuation from the mean

$$Var(X) = E\left(\left(X - E(X)\right)^2\right) = E(X^2) - E(X)^2$$
$$= M_2 - M_1^2$$

- Standard deviation: square root of variance $\sqrt{Var(x)} = \sqrt{M_2 - M_1^2}$

Bivariate Distributions

- Joint Distributions: $P(X = x, Y = y)$, a list of all probabilities of all possible pairs of observations
- Marginal Distribution:

$$P(X = x) = \sum_{y} P(X = x, Y = y)$$

- Conditional Distribution: $P(X = x|Y = y) = P(X = x, Y = y)$
  - $X|Y$ has distribution $P(X|Y)$, where $P(X|Y)$ specifies a "lookup-table" of all possible $P(X = x|Y = y)$

Expectation and Covariance of Bivariate Distributions

- Conditional distributions are just distributions which have a (conditional) mean or variance
- Covariance is the expected value of the product of fluctuations:

$$Cov(X,Y) = E\Big(\big(X - E(X)\big)\big(Y - E(Y)\big)\Big) = E(XY) - E(X)E(Y)$$

$$Var(X) = Cov(X,X)$$

- One common way to construct bivariate random variables is to have a random variable whose parameter is another random variable.

Expectation and Covariance of Bivariate Distributions

- Two events are independent if knowing that the first took places tells us nothing about the probability of the second: $P(A|B) = P(A)$
- $P(A)P(B) = P(A \cap B)$
- Two random variables are independent if the joint PMF is the product of the marginals:
  - $P(X = x, Y = y) = P(X = x)P(Y = y)$
- If X and Y are independent, we write $X \perp Y$. Knowing the value of X does not tell us anything about Y.
- If X and Y are independent, $Cov(X,Y)=0$.
- Mutual information is a measure of how “non-independent" two random variables are.

Multivariate Distributions

- $\boldsymbol{X}$, $\boldsymbol{x}$ are vector.
- Mean:

$$E(\boldsymbol{X}) = \sum_{x} \boldsymbol{x} P(\boldsymbol{x})$$

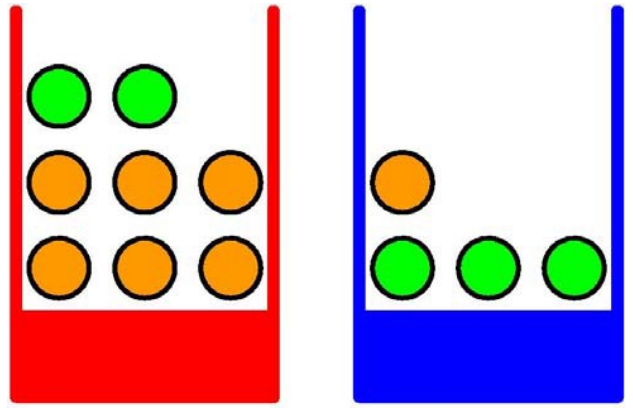
- Covariance Matrix:

$$Cov(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j)$$
$$Cov(\boldsymbol{X}) = E(\boldsymbol{X}\boldsymbol{X}^T) - E(\boldsymbol{X})E(\boldsymbol{X})^{\boldsymbol{T}}$$

- Conditional and marginal distributions: Can define and calculate any (multi- or single dimensional) marginals or conditional distributions we need: $P(X_1), P(X_1, X_2), P(X_1, X_2, X_3|X_4)$, etc…

Example

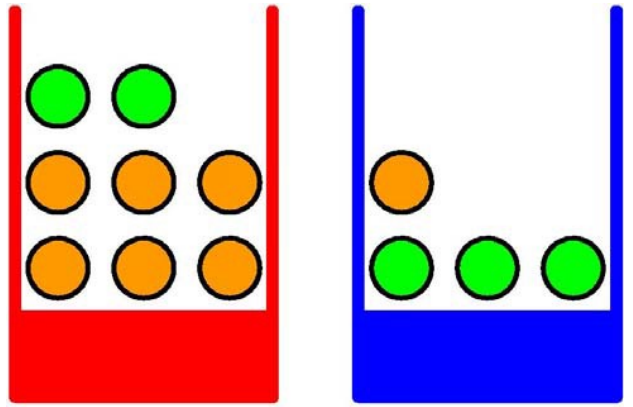
- Let us look at the following example:
  - We have two boxes, one red and one blue
  - Red box: 2 apples and 6 oranges
  - Blue box: 3 apples and 1 orange
  - Pick red box 40% of the time and blue box 60% of the time, then pick one item of fruit



C.M. Bishop, “Pattern Recognition and Machine Learning”, 2006

Example

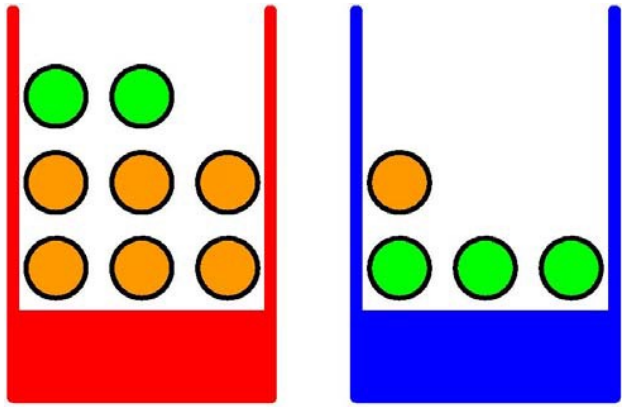
- Define:
  - B random variable for box picked
    - B = {blue(b), red(r)}
  - F identity of fruit
    - F = {apple(a), orange(o)}
  - P(B=r)=0.4 and P(B=b)=0.6
  - Events are mutually exclusive and include all possible outcomes
  - Their probabilities must sum to 1



Example

- P(B=r) = 0.4, P(B=b) = 0.6
- P(B=r) + P(B=b) = 1.0

- Conditional Probabilities
  - P(F=a|B=r) = 2/8 = 0.25
  - P(F=o|B=r) = 6/8 = 0.75
  - P(F=a|B=b) = 3/4 = 0.75
  - P(F=o|B=b) = 1/4 = 0.25



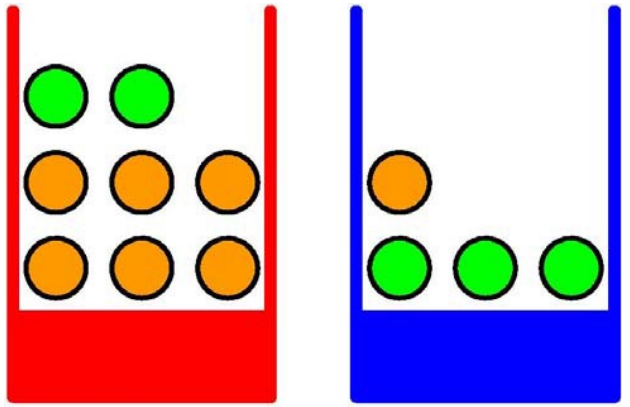
Example

- Note: P(F=a|B=r)+P(F=o|B=r) = 1

- P(F=a) = P(F=a|B=r)P(B=r)+P(F=a|B=b)P(B=b)
  = 1/4 * 4/10 + 3/4 * 6/10 = 11/20

- P(F=o) = 1-11/20=9/20=0.45

P(F=a|B=r) = 2/8 = 0.25
P(F=o|B=r) = 6/8 = 0.75
P(F=a|B=b) = 3/4 = 0.75
P(F=o|B=b) = 1/4 = 0.25



Prior vs. Posterior

- Prior Probability - If we had been asked which box had been chosen before being told the identity of the selected item of fruit, then the most complete information we have available is provided by the probability P(B).
- Posterior Probability - Once we are told that the fruit is an orange, we can then use Bayes' theorem to compute the probability $P(B|F)$, which we shall call the posterior probability because it is the probability obtained after we have observed F.

Conditional Probability

- Conditional probability: Recalculated probability of event A after someone tells you that event B happened.

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

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

- Bayes Rule:

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

Bayesian Probability

- Bayesian view: probabilities provide a quantification of uncertainty. Before observing the data, the assumptions about *w* are captured in the form of a prior probability distribution $P(\boldsymbol{w})$. The effect of the observed data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is expressed by $P(D|\boldsymbol{w})$.
- Bayes' theorem:

$$P(\boldsymbol{w}|D) = \frac{P(D|\boldsymbol{w})P(\boldsymbol{w})}{P(D)}$$

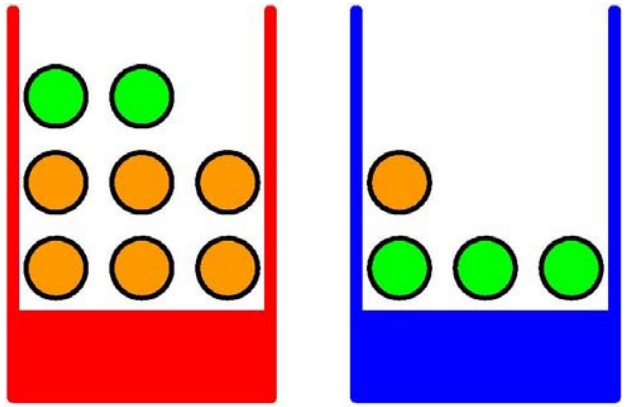
- Bayes' theorem in words: **posterior $\propto$ likelihood $\times$ prior**

Bayes Rule on the Fruit Example

- Assume we have now picked an orange
- Question: which is the probability that it was from the red box?

$$P(B = r|F = o) = \frac{P(F = o|B = r)P(B = r)}{P(F = o)} = \frac{0.75 \times 0.4}{0.45} = \frac{2}{3}$$

P(B=r) = 0.4, P(B=b) = 0.6
P(F=o) = 0.45,
P(F=o|B=r) = 6/8 = 0.75



Continuous Random Variables

- A random variable X is continuous if its sample space X is uncountable.
- In this case, $P(X = x) = 0$ for each x.
- If $p_x(x)$ is a probability density function for X, then

$$P(a < X < b) = \int_a^b p(x)dx$$

$$P(a < X < a + dx) \approx p(a) \cdot dx$$

Continuous Random Variables

- The cumulative distribution function is $F_x(x) = P(X < x)$. We have that $p_x(x) = F'(x)$, and $F(x) = \int_{-\infty}^{x} p(s)ds$.
- More generally, If A is an event, then

$$P(A) = P(X \in A) = \int_{x \in A} p(x)dx$$

$$P(\Omega) = P(X \in \Omega) = \int_{x \in \Omega} p(x)dx = 1$$

Mean, Variance, and Conditionals

- Mean: $E(x) = \int_x x \cdot p(x)dx$
- Variance: $Var(X) = E(X^2) - E(X)^2$
- If X has pdf *p(x)*, then $X|(X \in A)$ has pdf

$$p_{(x|A)}(x) = \frac{p(x)}{P(A)} = \frac{p(x)}{\int_{x \in A} p(x)dx}$$

- Only makes sense if $P(A) > 0$!

Bivariate Continuous Distributions

- $p_{x,y}(x, y)$, joints probability density function of X and Y
- $\int_x \int_y p(x, y) dx dy = 1$
- Marginal distribution: $p_X(x) = \int_{-\infty}^{\infty} p(x, y) dy$
- Conditional distribution: $p(x|y) = \frac{p(x,y)}{p(y)}$
- Note: $P(Y = y) = 0$!
- Independence: X and Y are independent if $p_{(x,y)}(x, y) = p_x(x) p_y(y)$

The Univariate Gaussian

- Probability density function

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

- Easy to validate:

$$\int_{-\infty}^{\infty} p(x|\mu,\sigma^2)dx = 1$$

- Expectation

$$E(x) = \int_{-\infty}^{\infty} p(x|\mu,\sigma^2)xdx = \mu$$

- Variance

$$Var(x) = E(x^2) - E(x)^2 = \sigma^2$$

Products of Gaussian pdfs

- Suppose $p_1(x) = p\left(x, \mu_1, \frac{1}{\beta_1}\right)$ and $p_2(x) = p\left(x, \mu_2, \frac{1}{\beta_2}\right)$, then

$$p_1(x)p_2(x) \propto p\left(x, \mu, \frac{1}{\beta}\right)$$

$$\beta = \beta_1 + \beta_2$$

$$\mu = \frac{1}{\beta}(\beta_1\mu_1 + \beta_2\mu_2)$$

- In general,

$$p_1(x)p_2(x) \dots p_n(x) \propto p\left(x, \mu, \frac{1}{\beta}\right)$$

$$\beta = \sum_n \beta_n$$

$$\mu = \frac{1}{\beta}\sum_n \mu_n\beta_n$$

- This is also true for multivariate Gaussians!

Maximum Likelihood Estimator

- Assuming data points are independent and identically distributed (i.i.d.), the probability of the data set given μ and σ^2 (the likelihood function):

$$P(\boldsymbol{x}|\mu,\sigma^2) = \prod_{n=1}^{N} p(x_n|\mu,\sigma^2)$$

- log-likelihood:

$$\log P(\boldsymbol{x}|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\log\sigma^2 - \frac{N}{2}\log(2\pi)$$

- Maximizing log-likelihood with respect to μ and σ^2:

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n \,, \sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{ML})^2$$

Maximum Likelihood Estimator

- Assuming data points are independent and identically distributed (i.i.d.), the probability of the data set given μ and σ^2 (the likelihood function):

$$P(\boldsymbol{x}|\mu,\sigma^2) = \prod_{n=1}^{N} p(x_n|\mu,\sigma^2)$$

- log-likelihood:

$$\log P(\boldsymbol{x}|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\log\sigma^2 - \frac{N}{2}\log(2\pi)$$

- Maximizing log-likelihood with respect to μ and σ^2:

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n \,, \sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{ML})^2$$

Summary

- Linear Algebra & Probability Theory
  - Linear Algebra: we reviewed definitions.
  - Linear Algebra: Orthogonality, Operations, and Eigen vectors and values will be the major concepts need to know for machine learning algorithms' understanding
  - Probability theory will be mainly used to understand the models made from machine learnings.
  - Note that not all machine learning algorithms use probability theories to produce outputs. There are also non-probabilistic algorithms that sometimes are relatively useful and robust.
- Python Review
  - From the demonstration files, we looked some of data types in python and observed how to handle them. (There will be more as we go through the semester.)
  - We observed how to computations.

CS 559: Classification

Lecture 5

In Jang

ijang@stevens.edu

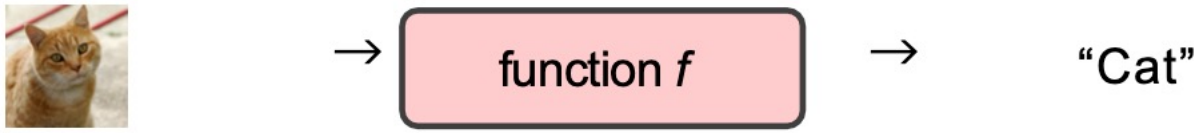
Lecture Outline

- Classification I
  - KNN
  - Linear Discriminant Analysis (LDA)
  - Perceptron
  - Naïve Bayes
- Classification II
  - Probabilistic Generative Models
  - Probabilistic Discriminative Models (Logistic Regression)
  - Generalization

Classification

Classification task: finding a function f that classifies examples into given set of categories $\{C_1, C_2, .., C_k\}$

A classification example:



KNN - K nearest neighbors

- Idea: average the values of the k closest observations

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  where $N_k(x)$ is the set of observations with the k smallest distances to the query point x.

- Assumption: similar inputs have similar outputs
- Rule: for a test input x, assign the most common label amongst its k most similar training inputs.
- Steps:
  1. Choose the number of k and a distance metric.
  2. Find the k-nearest neighbors of the sample that we want to classify.
  3. Assign the class label by majority vote.

KNN - K nearest neighbors

- What distance function to use?
  - KNN relies on a distance metric.
  - Better classification from the better metric reflecting similar similarity
  - Common choice: the Makowski distance

$$\text{dist}(\boldsymbol{x}, \boldsymbol{z}) = \left(\sum_{r=1}^{d} |x_r - z_r|^p \right)^{1/p}$$

    - If $p = 1$: Manhattan distance
    - If $p = 2$: Euclidean distance
    - If $p \to \infty$: Max

- Advantage: the classifier adopts immediately as we collect new training data.
- Disadvantage:
  - the computational cost grows linearly with the number of samples
  - Very susceptible to overfitting due to the **curse of dimensionality**

KNN – an example

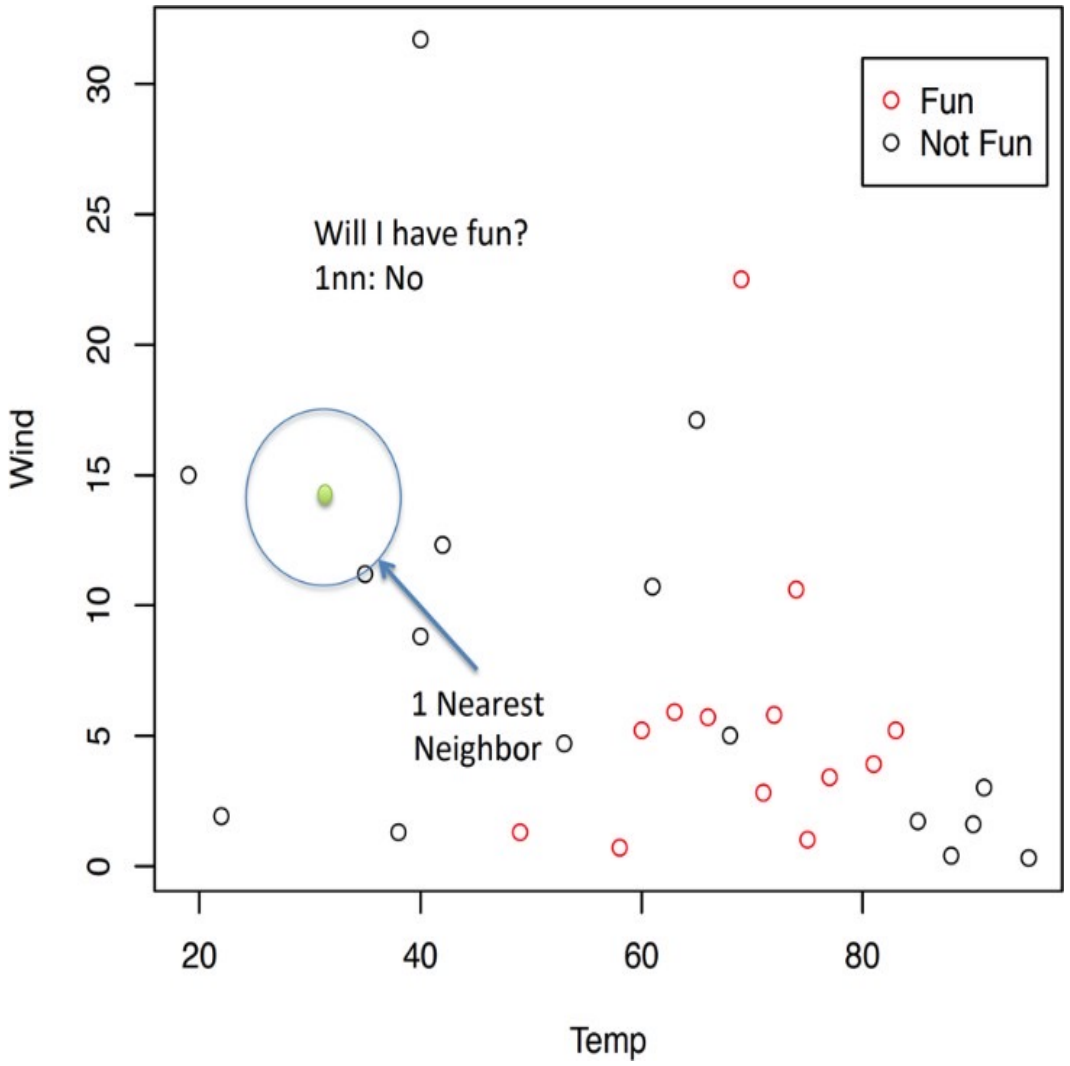
- Is it a good day to go for a run?

- A runner´s data on past running. With recorded temperature, wind and whether the run was fun:
  - Temperature (degrees F)
  - Wind Speed (mph)
  - Fun (yes, no)

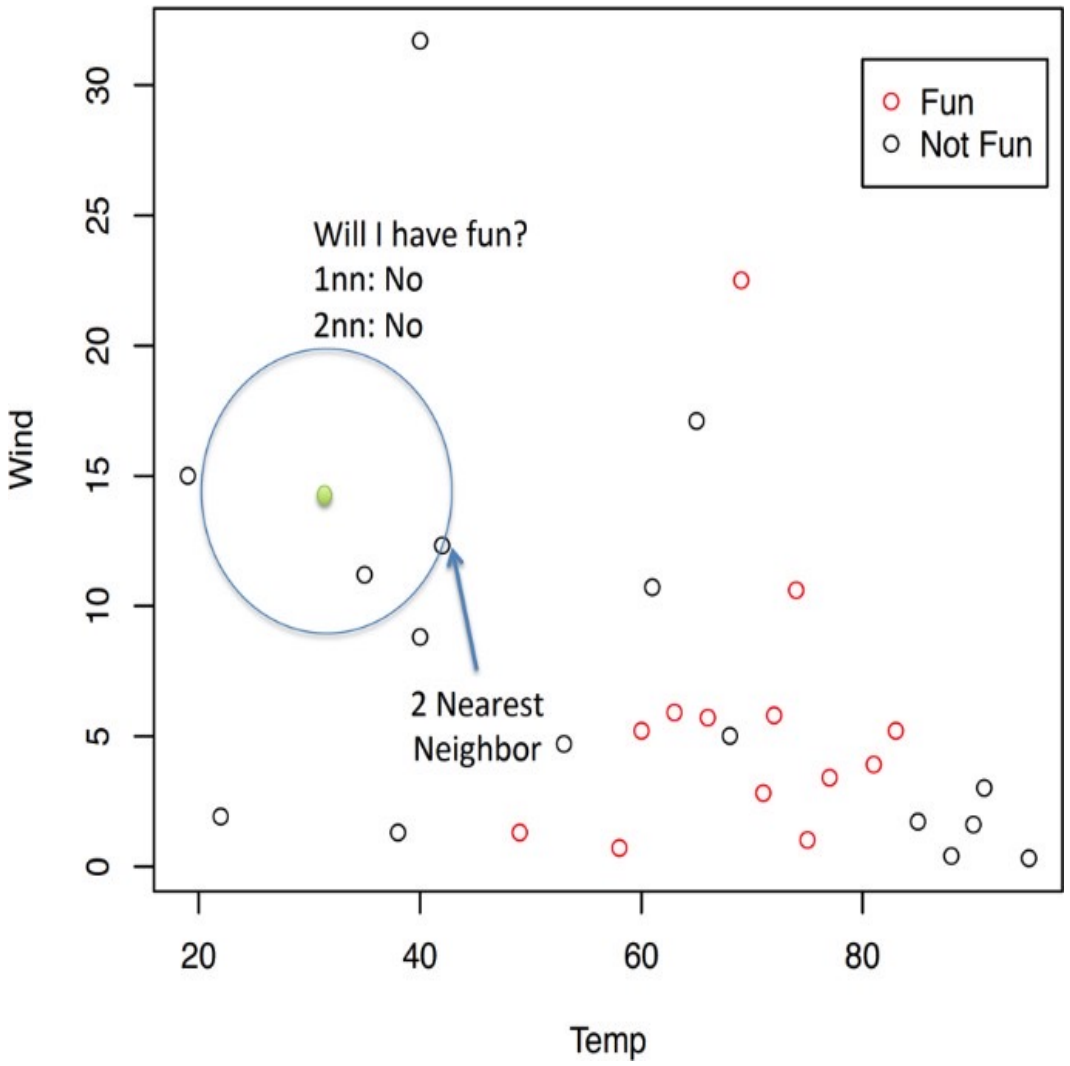
- It is now 65 degrees and the wind is 9 mph. Will a run be fun?

KNN classification

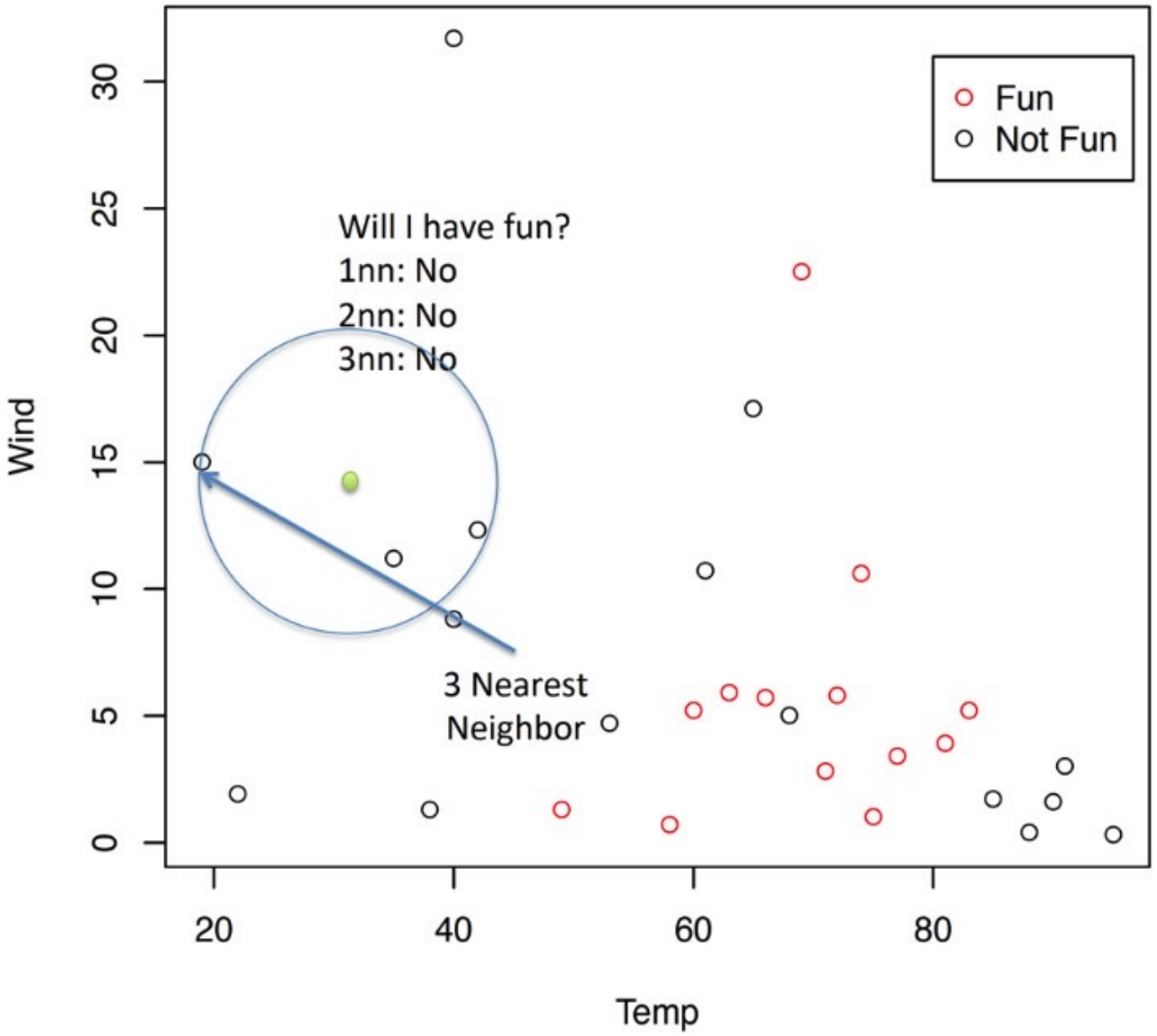
KNN classification: k=1



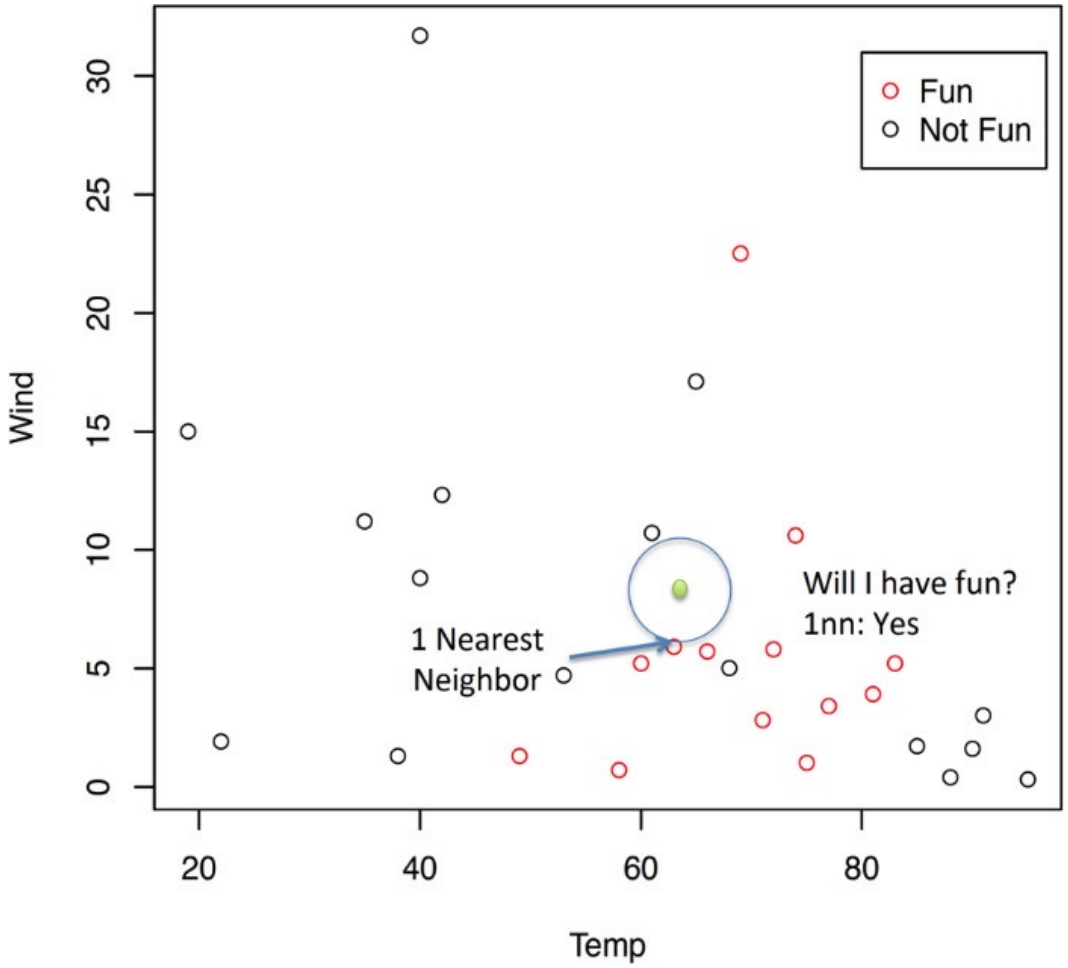
KNN classification: k=2

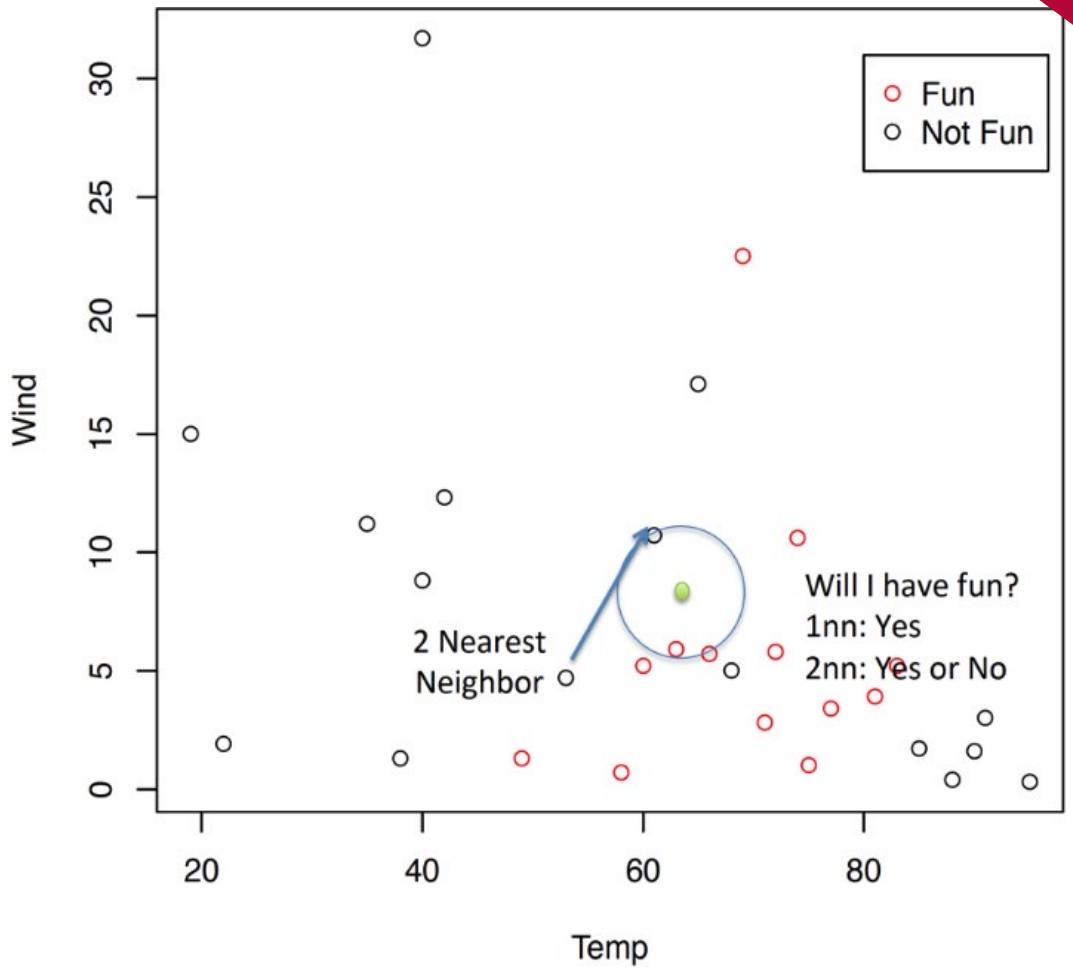


KNN classification: k=3

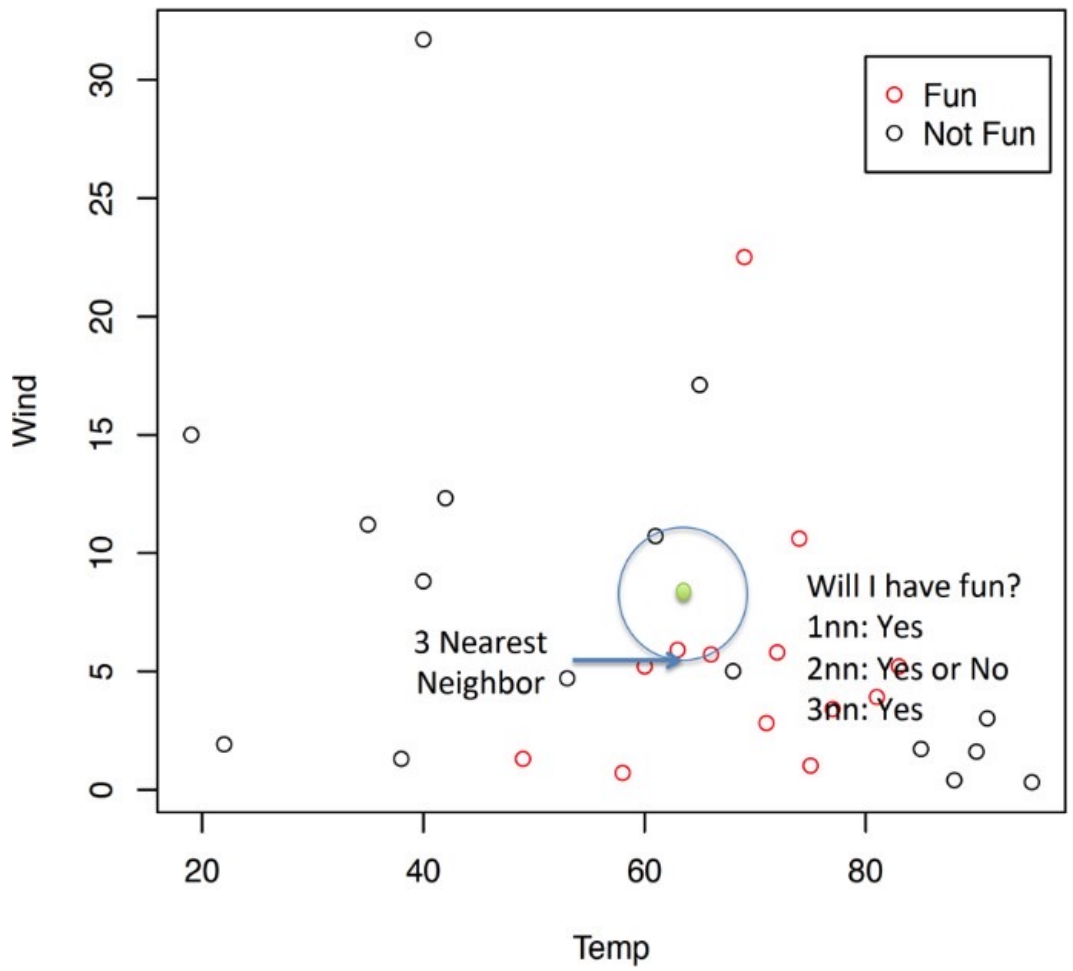


KNN classification: k=1





KNN classification: k=3



Decision Theory for Classification

Decision theory, when combined with probability theory, allows us to make optimal decisions in situations involving uncertainty.

- Training data: input values X and target values y
- Inference stage: use the training data to learn a model for $p(C_k|x)$
- Decision stage: use the given posterior probabilities to make optimal class assignments.

Generative Methods

- Solve the inference problem of estimating the class-conditional densities $p(x|C_k)$ for each class C_k
- Infer the prior class probabilities $p(C_k)$
- Use Bayes' theorem to find the class posterior probabilities:

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})}$$

where

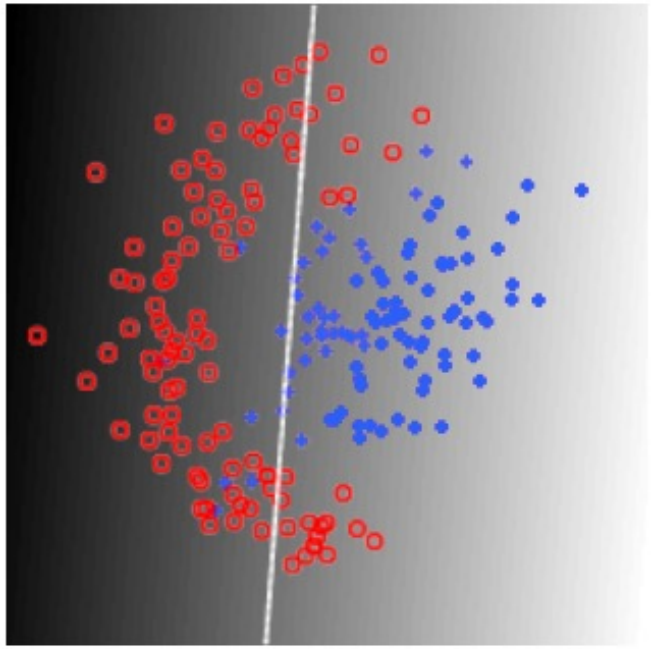
$$p(x) = \sum_k p(\boldsymbol{x}|C_k)p(C_k)$$

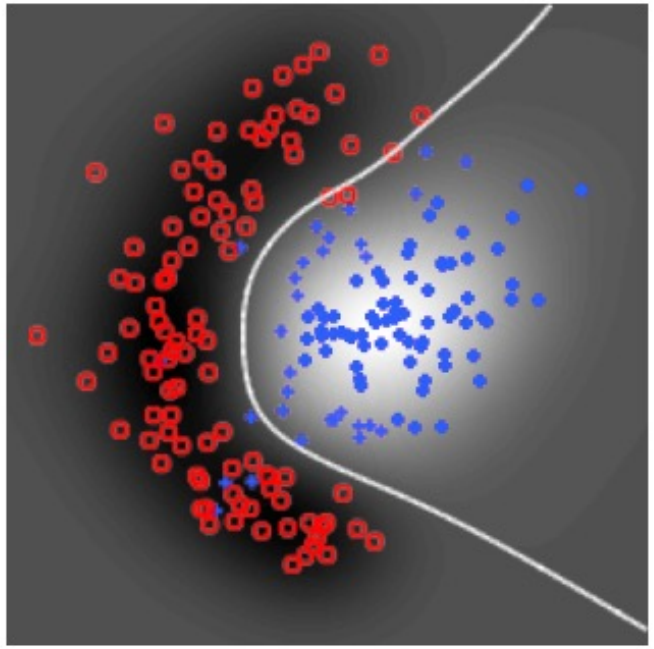
- Use decision theory to determine class membership for each new input *x*.

Discriminative Methods

- Solve directly the inference problem of estimating the class posterior probabilities $p(C_k|x)$.
- Discriminative Functions: Find a function $f(x)$ which maps each input directly onto a class label. Probabilities play no role here.
- Use decision theory to determine class membership for each new input x.

Linear Discriminant Functions





Of course, linear algorithms can be used together with nonlinear feature spaces or nonlinear basis functions in order to solve nonlinear classification problems!

Linear Discriminant Functions

- Linear discriminants separate the space by a hyperplane, and the parameters define its normal vector.
- Decision function: $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_o$
- Classification:

$$\text{if } f(x) > 0 \text{ say } x \text{ belongs to class 1 if } f(x) < 0 \text{ say } x \text{ belongs to class -1}$$

- The decision-surface has equation $f(x) = 0$, and is a hyperplane of dimensionality $D - 1$.
- $\boldsymbol{w}$ is the normal vector to the hyperplane, and points into the positive class or negative class.
- w_0 determines the location of the decision-surface
- $|f(\boldsymbol{x})|$ is proproptional to the perpendicular distance to the decision-surface (with factor 1 if $\|\boldsymbol{w}\| = 1$).

Linear Discriminant Functions-Geometrical Properties

- Decision boundary:

$$f(x) = w^T x + \omega_o = 0$$

- Let x_1, x_2 be two points which lie on the decision boundary

$$f(x_1) = w^T x_1 + \omega_o = 0, f(x_2) = w^T x_2 + \omega_o = 0$$
$$\Rightarrow w^T (x_1 - x_2) = 0$$

- ***w*** represents the orthogonal direction to the decision boundary.

Linear Discriminant Functions-Geometrical Properties Cont.

$$x_1 = x_2 + r\frac{w}{\|w\|}$$

$$f(x_2) = f\left(x_1 - r\frac{w}{||w||}\right) = (w^T x_1 + w_0) - r||w||$$

$$\rightarrow r = \frac{f(x_1)}{\|w\|}$$

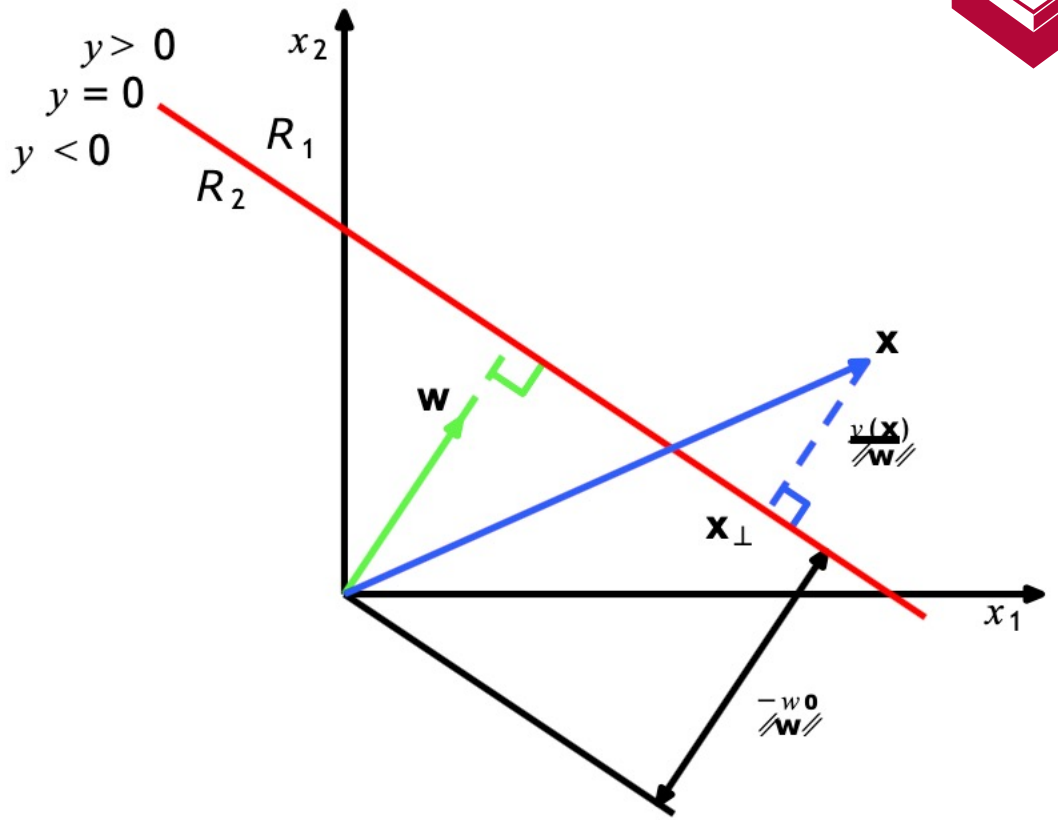
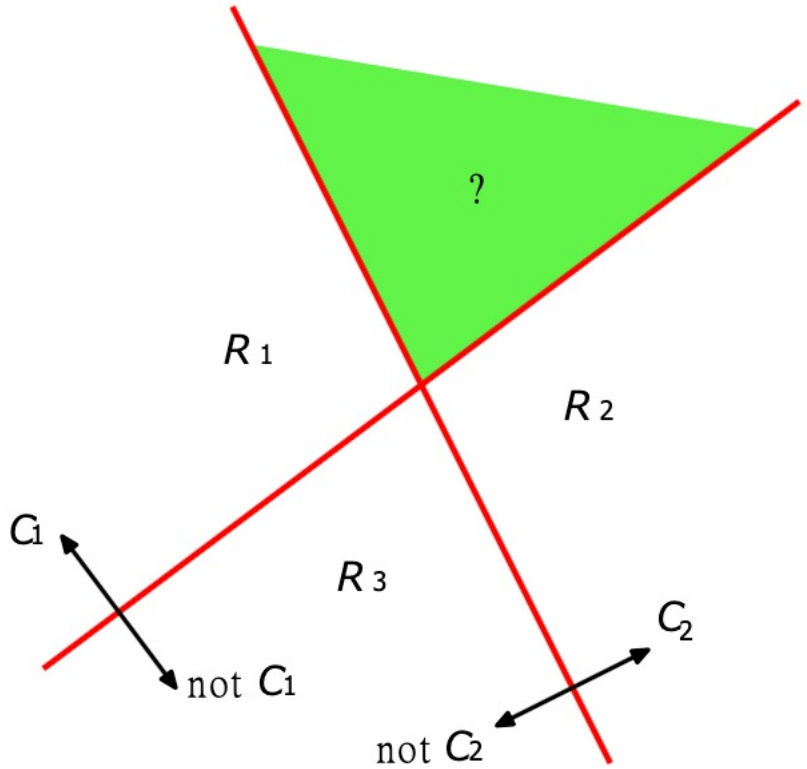


Figure: Signed orthogonal distance of the origin from the decision

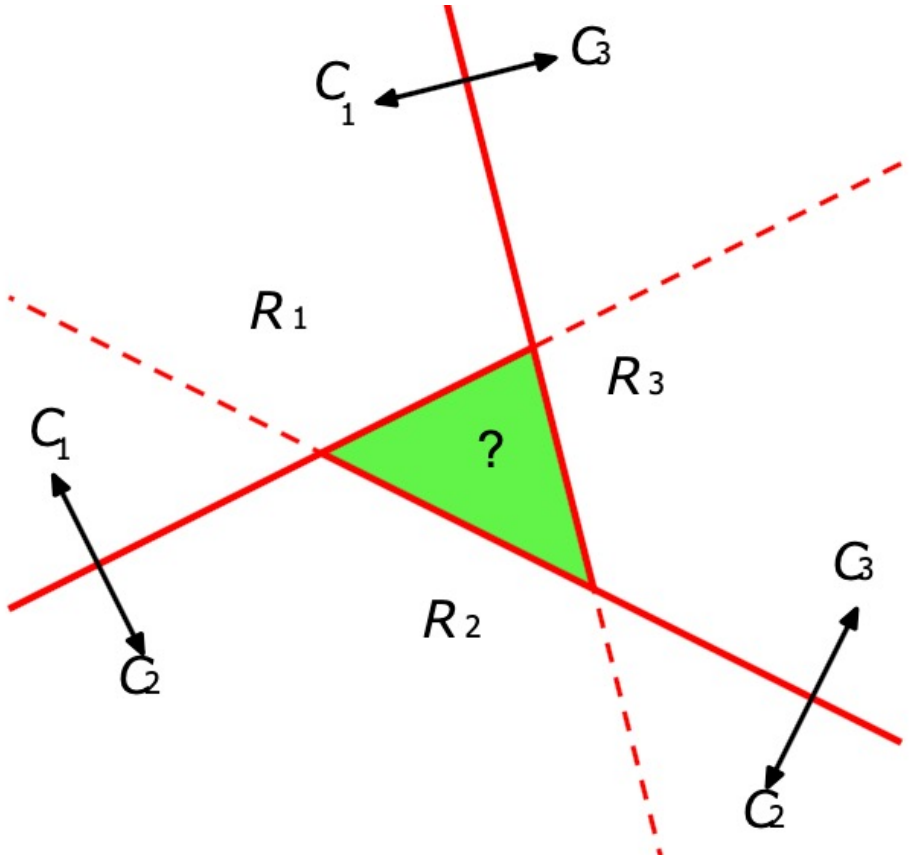
Linear Discriminant Functions: Multiple classes

one-versus-the-rest: $K - 1$ classifiers each of which solves a two-class problem of separating points of C_k from points not in that class.



Linear Discriminant Functions: Multiple classes

one-versus-one: K(K-1)/2 binary discriminant functions, one for every possible pair of class. Each point is then classified according to a majority vote amongst the discriminant functions.



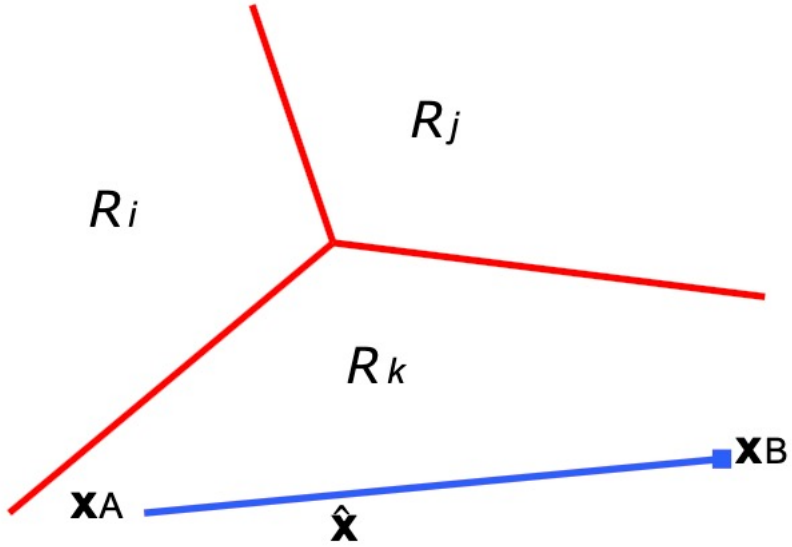
Linear Discriminant Functions: Multiple classes

- Solution: consider a single K-class discriminant comprising K linear functions of the form

$$f_k(x) = w^T x + w_{k0}$$

- Assign a point x to class C_k if $f_k(x) > f_j(x)\ \forall j, x = k$
- The decision boundary between class C_k and class C_j is given by:

$$f_k(x) = f_j(x) \Rightarrow (wk - w_j)^T x + (w_{k0} - w_{j0}) = 0$$

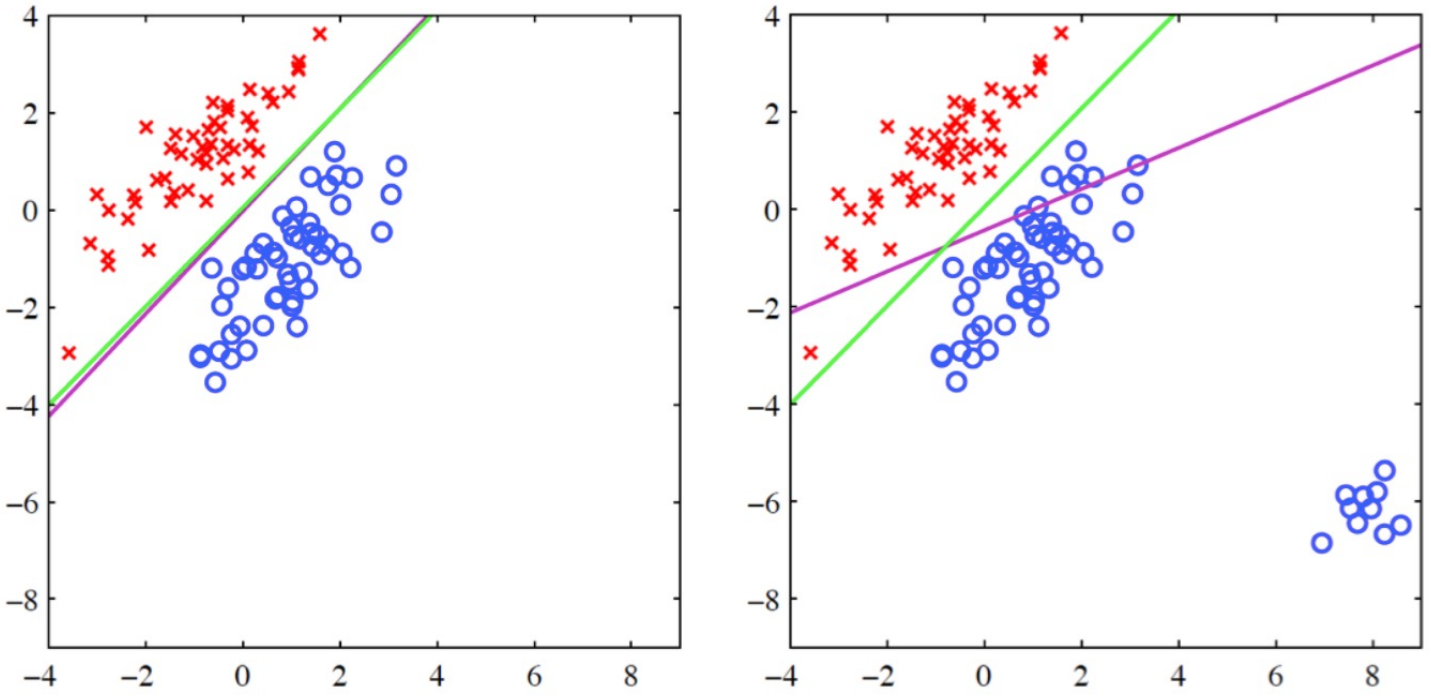


Least square classification

- We have to fit the function $f\ (x)\ =\ w^T x\ +\ \omega_o$ to data.
- Simply do a linear regression from x to y by minimizing the sum-of-squared errors $\sum_n (f(\boldsymbol{x_n}) - y_n)^2$.

$$w_{reg} = \left(\sum_n x_n x_n^T\right)^{-1} \sum_n x_n y_n$$

- Q: In what situations might this be a bad idea?



Least square classification

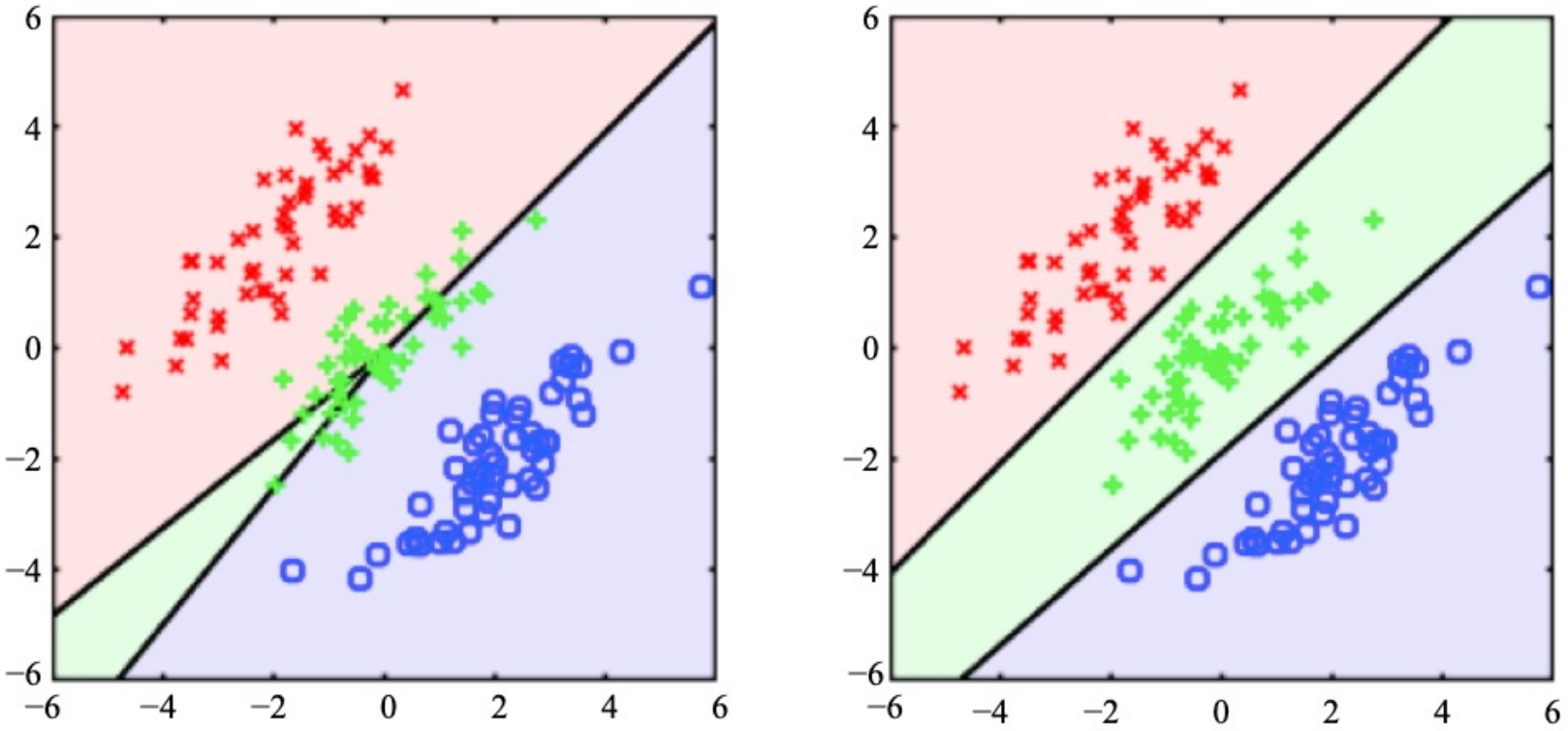
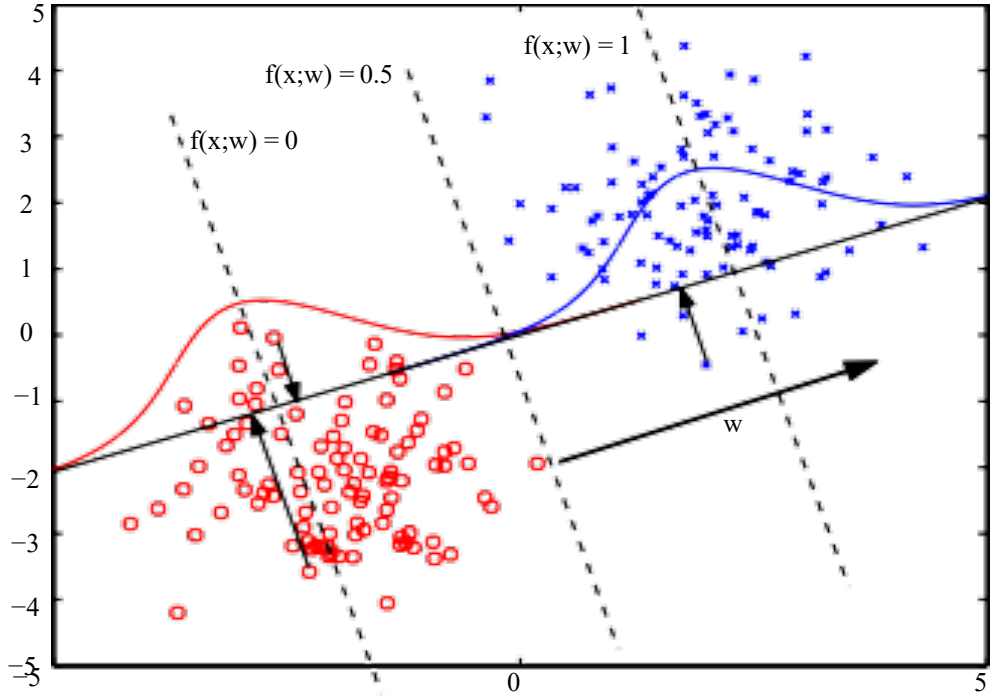


Figure: Left: using a least-squares discriminant; Right: using logistic regression

Classification via projection

- A linear function: $f(\boldsymbol{x}) = \boldsymbol{w^T x} + w_o$ assuming in 2D, projects each point $\boldsymbol{x} = [x_1, x_2]^T$ to a line parallel to $\boldsymbol{w}$:

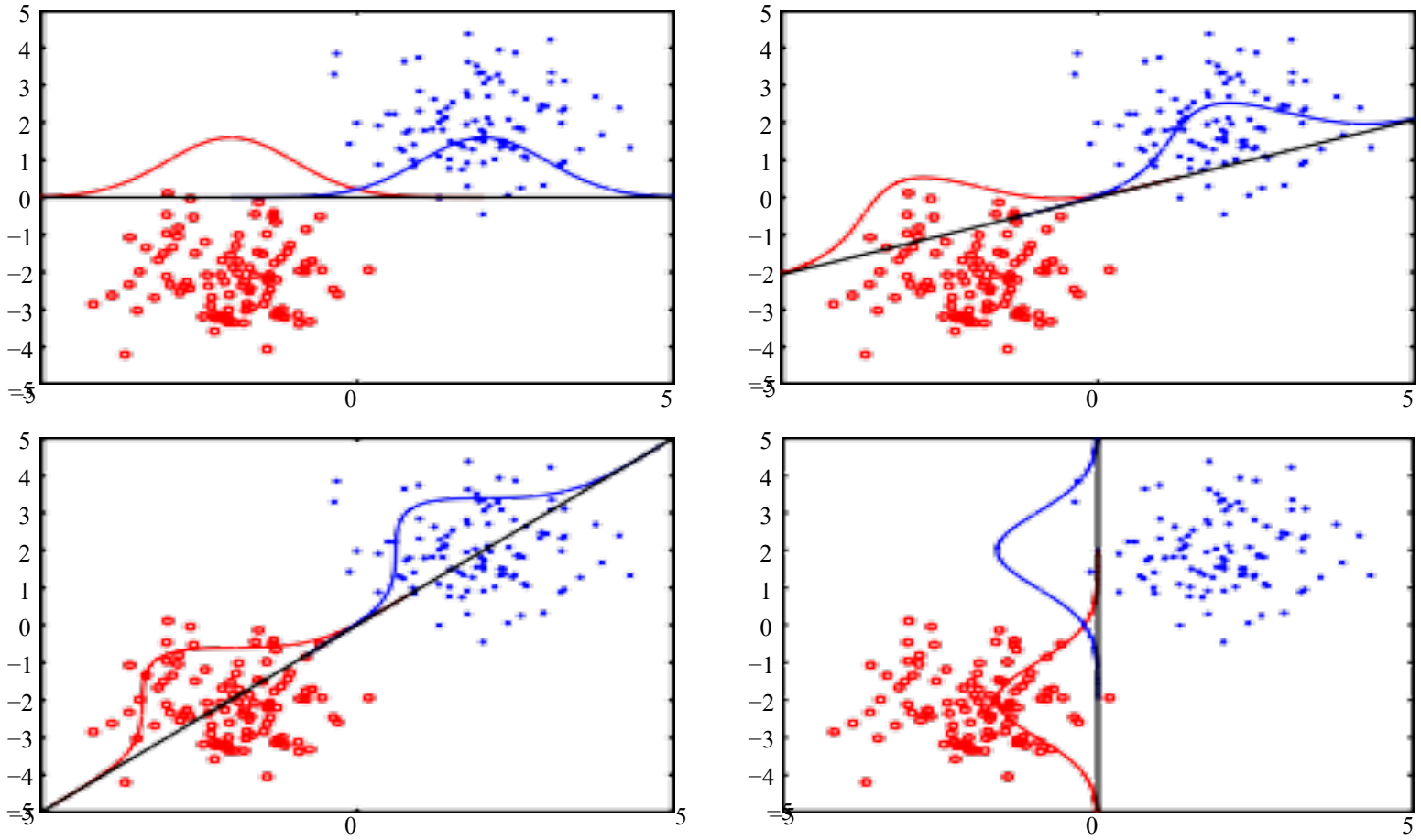
Point in $\boldsymbol{R^d}$	Projected point in R
x_1	$z_1 = w^T x_1$
x_2	$z_2 = w^T x_2$
$\vdots$	$\vdots$
x_n	$z_n = w^T x_n$



- We can study how well the projected points $z_1, \ldots, z_n$ viewed as functions of w are separated across the classes.

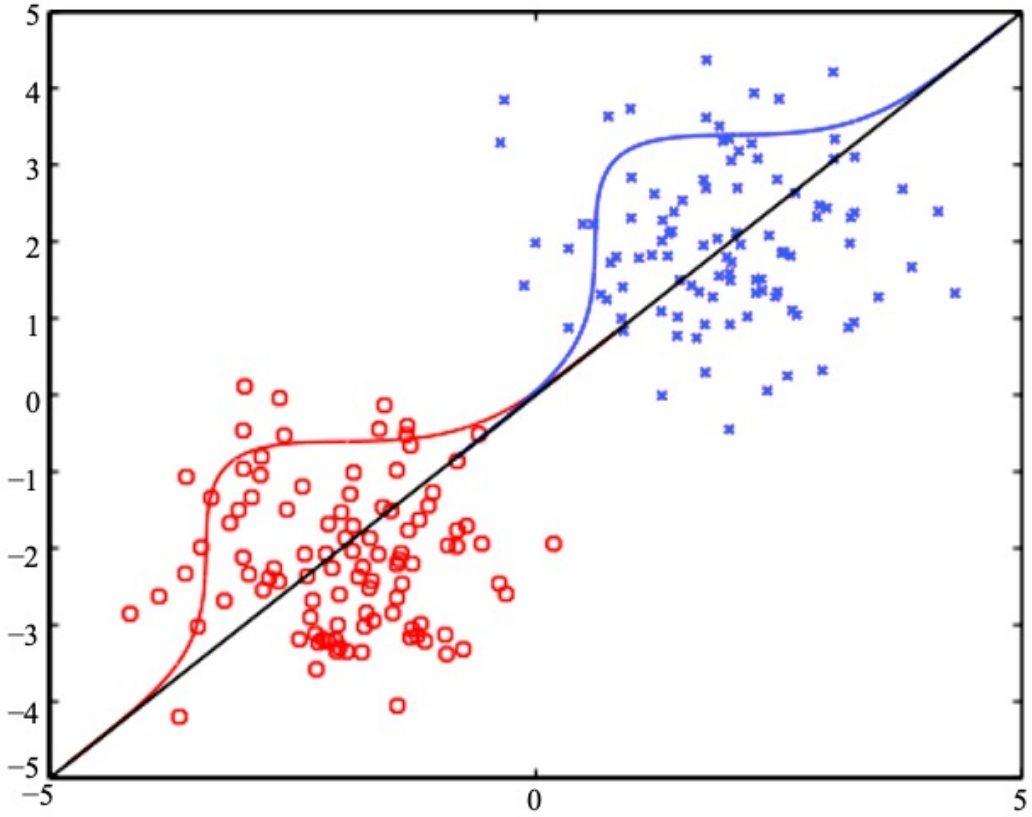
Classification via projection

By varying $\boldsymbol{w}$ we get different levels of separation between the projected points



Optimizing the projection

- We would like to find $\boldsymbol{w}$ that somehow maximizes the separation of the projected points across classes.



- We can quantify the separation (overlap) in terms of means and variances of the resulting 1-dimensional class distributions

Fisher's linear discriminant

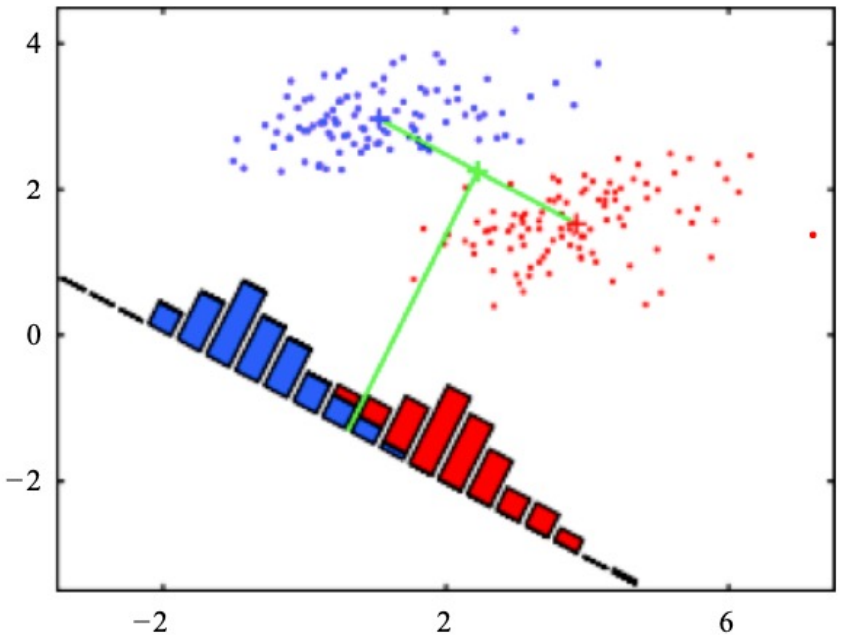
- One way to view a linear classification model is in terms of dimensionality reduction.
- Two class case: suppose we project x onto one dimension:

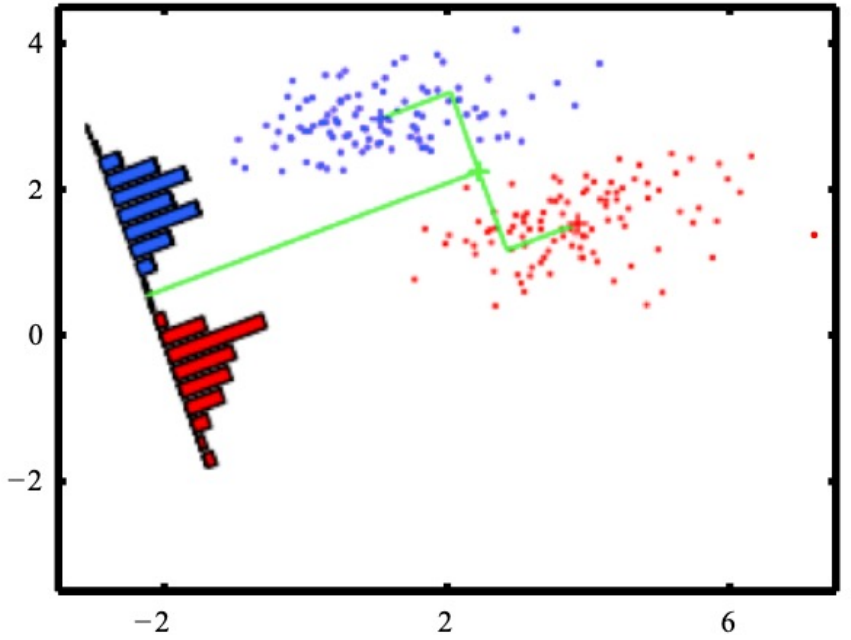
$$f = \boldsymbol{w}^T \boldsymbol{x}$$

- Set a threshold t:

$$\begin{array}{ll} \text{if } f \leq t & \text{assign } C_1 \text{ to } \boldsymbol{x} \\ \text{otherwise} & \text{assign } C_2 \text{ to } \boldsymbol{x} \end{array}$$

Fisher's linear discriminant





- Find an orientation along which the projected samples are well separated;
- This is exactly the goal of linear discriminant analysis (LDA);
- In other words: we are after the linear projection that best separates the data, i.e. best discriminates data of different classes.

Fisher's linear discriminant

- Two classes: $\{C_1, C_2\}$
- N_1 samples of class C_1
- N_2 samples of class C_2
- Consider $\boldsymbol{w} \in R^d$ with $||\boldsymbol{w}|| = 1$
- Then: $\boldsymbol{w}^T \boldsymbol{x}$ is the projection of x along the direction of $\boldsymbol{w}$.
- We want the projections $\boldsymbol{w}^T \boldsymbol{x}$ where $\boldsymbol{x} \in C_1$ separated from the projections $\boldsymbol{w}^T \boldsymbol{x}$ where $\boldsymbol{x} \in C_2$

Fisher's linear discriminant

- A measure of the separation between the projected points is the difference of the sample means:
- Sample mean of class C_1:

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{\mathrm{n}\in C_1} x_n$$

- Sample mean for the projected points:

$$m_1 = \frac{1}{N_1}\sum_{n\in C_1} \boldsymbol{w}^T\boldsymbol{x}_n = \boldsymbol{w}^T\boldsymbol{m}_1$$

$$\rightarrow |m_1 - m_2| = \boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

- We wish to make the above difference as large as we can. In addition, ...

Fisher's linear discriminant

To obtain good separation of the projected data, we really want the difference between the means to be large relative to some measure of the standard deviation of each class:

- Scatter of the projected samples of class C_1:

$$s_1^2 = \sum_{n \in C_1} (w^T x_n - m_1)^2$$

- Total within-class scatter of the projected samples:

$$s_1^2 + s_2^2$$

- Fisher linear discriminant analysis:

$$\arg\max_w \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

Fisher's linear discriminant

Fisher's criterion J(w): $J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$

To obtain J(w) as an explicit function of w, we define the following matrices:

$$\boldsymbol{S}_1 = \sum_{n \in C_1} (\boldsymbol{x_n} - \boldsymbol{m_1})(\boldsymbol{x_n} - \boldsymbol{m_1})^T$$

Within-class scatter matrix:

$$S_w = S_1 + S_2$$

Then:

$$s_1^2 = \sum_{x \in C_1} \left(\boldsymbol{w^T x} - m_1\right)^2 = \sum_{x \in C_1} \left(\boldsymbol{w^T x} - \boldsymbol{w^T m_1}\right)^2$$

$$= \sum_{x \in C_1} \boldsymbol{w^T}(\boldsymbol{x} - \boldsymbol{m_1})(\boldsymbol{x} - \boldsymbol{m_1})^{\boldsymbol{T}}\boldsymbol{w} = \boldsymbol{w^T S_1 w}$$

Fisher's linear discriminant

So, $s_1^2 = w^T S_1 w$ and $s_2^2 = w^T S_2 w$

Thus,

$$\begin{aligned} s_w = s_1^2 + s_2^2 &= \boldsymbol{w^T S_1 w + w^T S_2 w} \\ &= \boldsymbol{w^T (S_1 + S_2) w} \\ &= \boldsymbol{w^T S_w w} \end{aligned}$$

where $\boldsymbol{S_w} = \sum_{n \in C_1} (\boldsymbol{x_n - m_1})(\boldsymbol{x_n - m_1})^T + \sum_{n \in C_2} (\boldsymbol{x_n - m_2})(\boldsymbol{x_n - m_2})^T$.

Similarly:

$$\begin{aligned} (m_1 - m_2)^2 &= \left(\boldsymbol{w^T m_1 - w^T m_2}\right)^2 \\ &= \boldsymbol{w^T (m_1 - m_2)(m_1 - m_2)^T w} \\ &= \boldsymbol{w^T S_B w}. \end{aligned}$$

where $S_B = (\boldsymbol{m_1 - m_2})(\boldsymbol{m_1 - m_2})^T$ is the between-class scatter matrix.

Fisher's linear discriminant

We have obtained:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w^T S_B w}}{\boldsymbol{w^T S_w w}}$$

J(w) is maximized when

$$\boldsymbol{(w^T S_B w) S_w w = (w^T S_w w) S_B w}$$

We observe that

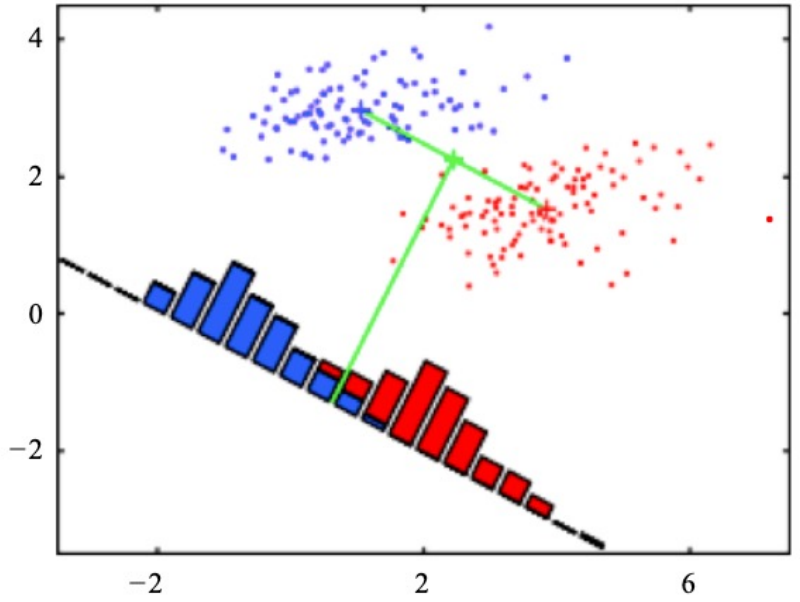
$$\boldsymbol{S_B w = (m_1 - m_2)(m_1 - m_2)^T w}$$

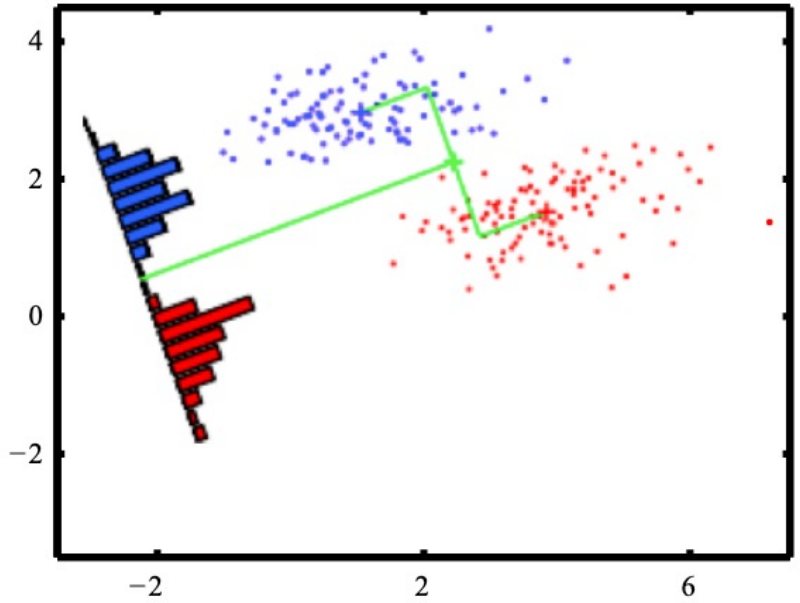
where $\boldsymbol{(m_1 - m_2)^T w}$ is a scalar and always in the direction of $\boldsymbol{(m_1 - m_2)}$.

Solution:

$$\boldsymbol{w \propto S_w^{-1}(m_1 - m_2)}$$

Fisher's linear discriminant Summary





- $\boldsymbol{m}_1 = \frac{1}{N}\sum_{n\in C_1} \boldsymbol{x_n}$ & $\boldsymbol{m}_2 = \frac{1}{N_1}\sum_{n\in C_2} \boldsymbol{x_n}$
- Maximize Projection-distance of class means $\boldsymbol{w_{simple}} \propto \boldsymbol{m_1 - m_2}$
- Maximizing distance between means ignores that the projected variances might also be big.
- Fix: Maximize the ratio of between-class variance to within-class variance ('signal to noise'). Fisher criterion:

$$J_w = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$w_{Ida} = S_w^{-1}(m_1 - m_2)$$

Fisher's linear discriminant Summary: Multi-Class

- The analysis can be extended to multiple classes.
- $S_w = \sum_{k=1}^{K} \sum_{\boldsymbol{x_i} \in C_k} (\boldsymbol{x_i} - \boldsymbol{m_k})(\boldsymbol{x_i} - \boldsymbol{m_k})^T$
- $S_B = \sum_{k=1}^{K} m_k (\boldsymbol{m_k} - \boldsymbol{m})(\boldsymbol{m_k} - \boldsymbol{m})^T$ where m is the global mean and m_k is the number of samples in class k.

- Solve: $S_B \boldsymbol{v} = \lambda S_W \boldsymbol{v}$ the generalized eigenvalue problem
- At most K-1 distinct solution eigenvalues
- The optimal projection matrix $\boldsymbol{V}$ to a subspace of dimension $\boldsymbol{k}$ is given by the eigenvectors corresponding to the largest k eigenvalues

Fisher's linear discriminant Summary: Multi-Class

- LDA is a linear technique for dimensionality reduction: it projects the data along directions that can be expressed as linear combination of the input features.
- The "appropriate" transformation depends on the data and on the task we want to perform on the data. Note that LDA uses class labels.
- Non-linear extensions of LDA exist (e.g., generalized LDA).

The Perceptron Algorithm (Frank Rosenblatt, 1957)

- First learning algorithm for neural networks.
- Originally introduced for character classification, where each character is represented as an image.
- Total input to output node:

$$\sum_j w_j x_j$$

- Output unit performs the function (activation function):

$$H(x) = \begin{cases} 1 \; if \; x \geq 0 \\ -1 \; if \; x < 0 \end{cases}$$

Perceptron: Learning Algorithm

- Goal: compute a mapping from inputs to the outputs.
- Example: two class character recognition problem.
  - Training set: set of images representing either the character ‘a’ or the character ‘b’ (supervised learning);
  - Learning task: learn the weights so that when a new unlabelled image comes in, the network can predict its label.
  - Setting: d input units (intensity level of a pixel), 1 output unit.

- The algorithm proceeds as follows:
  - Initial random setting of weights;
  - The input is a random sequence $\{x_k\}$
  - For each element of class C_1, if output = 1 (correct), do nothing; otherwise, update weights;
  - For each element of class C_2, if output = 0 (correct), do nothing; otherwise, update weights;

Perceptron: Learning Algorithm

- More formally: $x = (x_1, x_2, .., x_d)^T, w = (w_1, w_2, .., w_d)^T$
- θ: Threshold of the output unit
- Unit output: $w^T x = w_1 x_1 + w_2 x_2 + ... + x_d x_d$
- Output class 1 if $w^T x - \theta \geq 0$
- To eliminate the explicit dependence on θ: Output class 1 if: $w^T x \geq 0$

Perceptron: Learning Algorithm

- We want to learn values of the weights so that the perceptron correctly discriminate elements of C_1 from elements of C_2.

- Given **x** in input, if **x** is classified correctly, weights are unchanged, otherwise:

$$w = \begin{cases} w + x & \text{if an element of class } C_1 \text{ was classified as in } C_2 \\ w - x & \text{if an element of class } C_2 \text{ was classified as in } C_1 \end{cases}$$

Perceptron: Learning Algorithm

- 1st case: $\mathbf{x} \in C_1$ and was classified in C_2. The correct answer is 1, which corresponds to: $\mathbf{w}^T\mathbf{x} \geq 0$, we have $\mathbf{w}^T\mathbf{x} < 0$. We want to get closer to the correct answer: $\mathbf{w}^T\mathbf{x} < \mathbf{w}'^T\mathbf{x}$.

$$\mathbf{w}^T\mathbf{x} < \mathbf{w}'^T\mathbf{x}, \text{ iff } \mathbf{w}^T\mathbf{x} < (\mathbf{w} + \mathbf{x})^T\mathbf{x}$$

$$(\mathbf{w} + \mathbf{x})^T\mathbf{x} = \mathbf{w}^T\mathbf{x} + \mathbf{x}^T\mathbf{x} = \mathbf{w}^T\mathbf{x} + ||\mathbf{x}||^2$$

because $||\mathbf{x}||^2 > 0$, the condition is verified.

- 2nd case: $\mathbf{x} \in C_2$ and was classified in C_1. The correct answer is 0, which corresponds to: $\mathbf{w}^T\mathbf{x} < 0$, we have $\mathbf{w}^T\mathbf{x} \geq 0$. We want to get closer to the correct answer: $\mathbf{w}^T\mathbf{x} > \mathbf{w}'^T\mathbf{x}$.

$$\mathbf{w}^T\mathbf{x} > \mathbf{w}'^T\mathbf{x}, \text{ iff } \mathbf{w}^T\mathbf{x} > (\mathbf{w} - \mathbf{x})^T\mathbf{x}$$

$$(\mathbf{w} - \mathbf{x})^T\mathbf{x} = \mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{x} = \mathbf{w}^T\mathbf{x} - ||\mathbf{x}||^2$$

because $||\mathbf{x}||^2 > 0$, the condition is verified.

Perceptron: Learning Algorithm

In summary:

- A random sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ is generated such that $x_i \in C_1 \cup C_2$
- If $\mathbf{x}_k$ is correctly classified, then $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k$ otherwise:

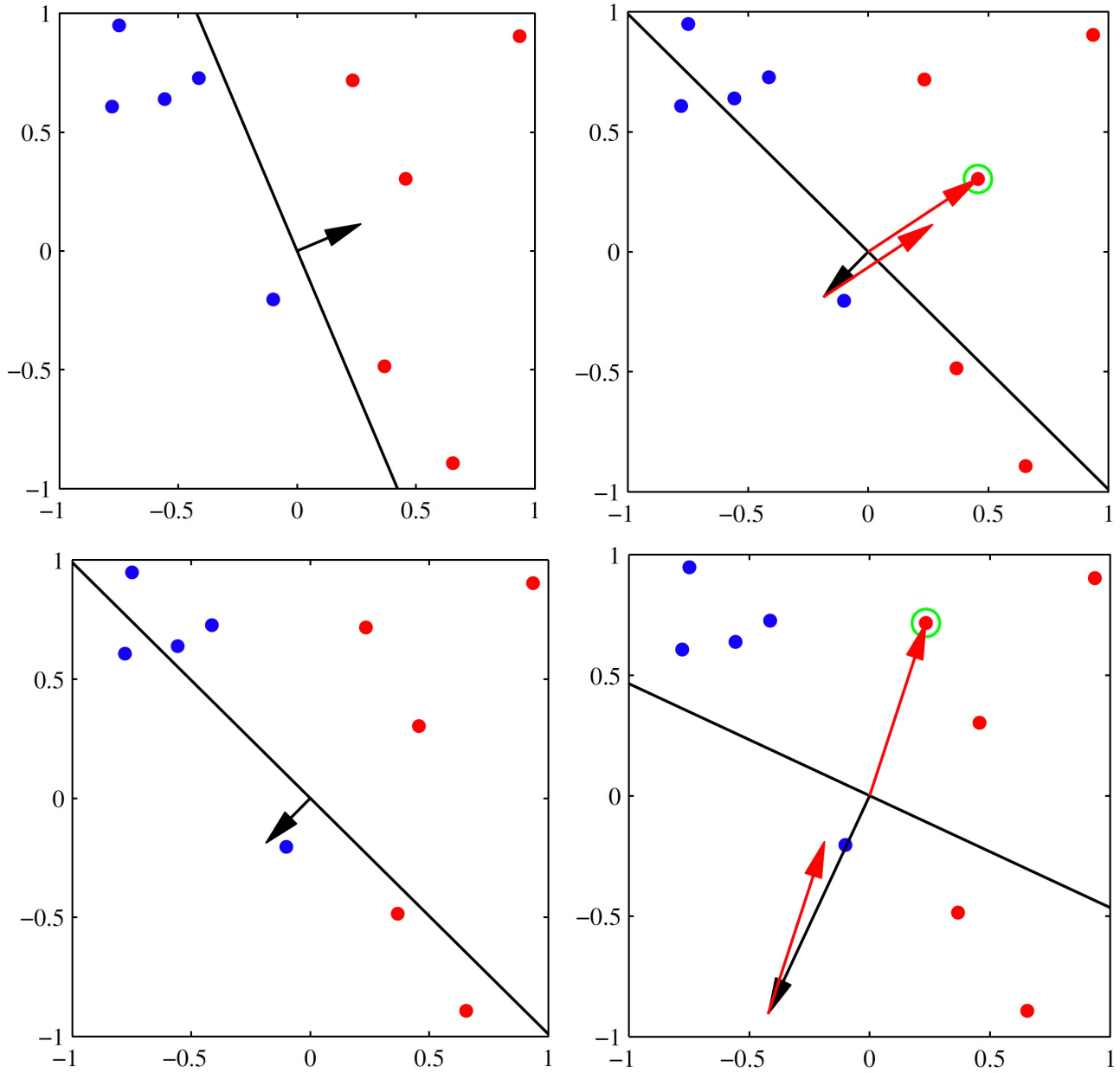
$$\mathbf{w_{k+1}} = \begin{cases} \boldsymbol{w_k} + \boldsymbol{x_k}\ if\ \boldsymbol{x_k} \in C_1 \\ \boldsymbol{w_k} - \boldsymbol{x_k}\ if\ \boldsymbol{x_k} \in C_2 \end{cases}$$

- Convergence theorem: regardless of the initial choice of weights, if the two classes are linearly separable, there exists $\mathbf{w}$ such that:

$$= \begin{cases} \boldsymbol{w^T x} \geq 0 \\ \boldsymbol{w^T x} < 0 \end{cases}$$

then the learning rule will find such solution after a finite number of steps.

Perceptron: Example



Naive Bayes: not (necessarily) a Bayesian method

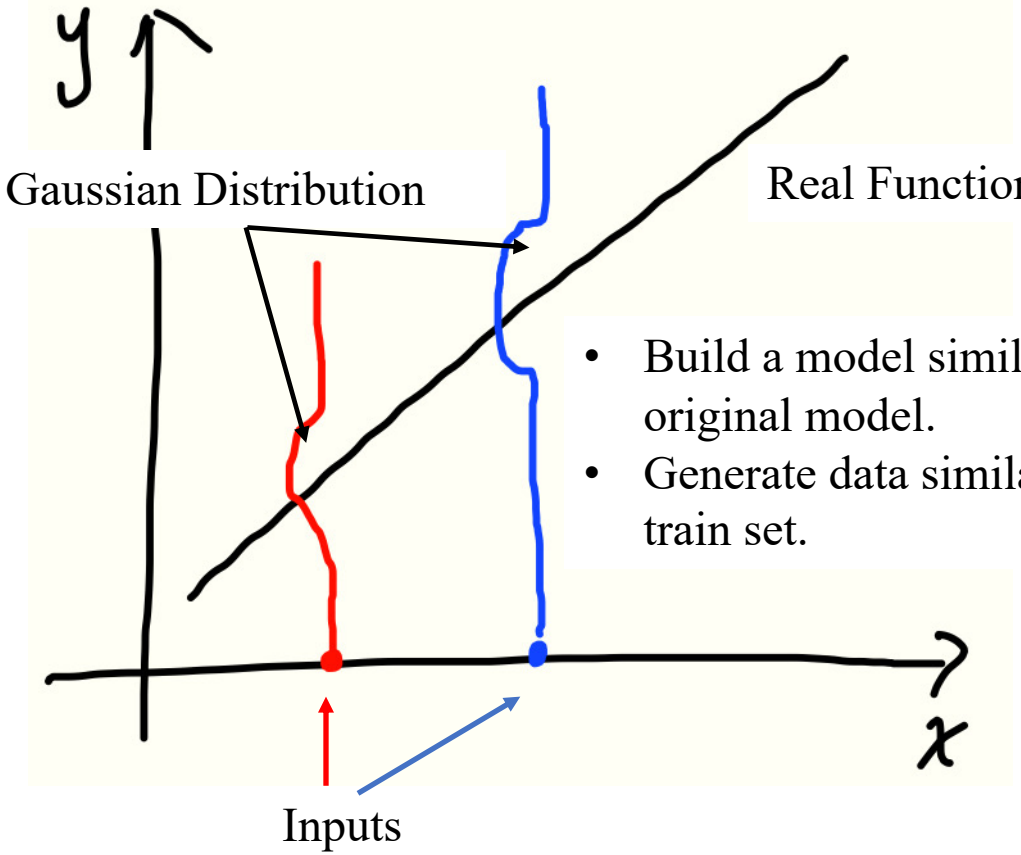
- A and B are independent iff $p(A, B) = p(A)p(B)$
- A and B are conditionally independent given C iff

$$p(A, B|C) = p(A|C)p(B|C)$$

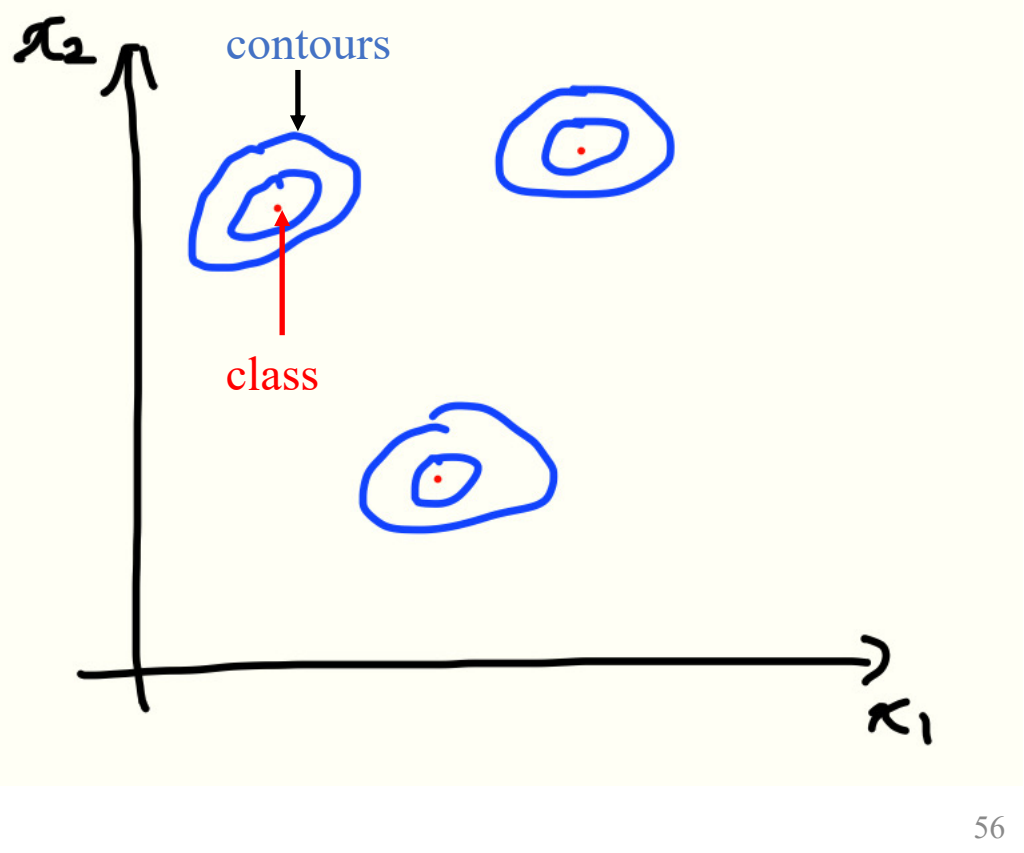
Linear Classification II

Probabilistic Generative Models

- We now turn to a probabilistic approach to classification.
- How models with linear decision boundaries arise from simple assumptions about the distribution of the data.



- Build a model similar to the original model.
- Generate data similar to the train set.



Generative Approach

- Infer the prior class probabilities $p(C_k)$.
- Solve the inference problem of estimating the class-conditional densities $p(\mathbf{x}|C_k)$ for each class C_k.
- Use Bayes' theorem to find the class posterior probabilities:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} \quad (1)$$

where

$$p(x) = \sum_k p(\mathbf{x}|C_k)p(C_k) \quad (2)$$

- Use decision theory to determine class membership for each new input x.

Assumptions

- In classification, the number of classes is finite, so natural prior $p(C)$ is the multinomial
  (e.g., coin or dice)

$$p(C = c_k) = \pi_k$$

- When $x \in \mathbb{R}^D$, then it is okay to assume that $p(x|C)$ is Gaussian.

- The class-conditional densities are Gaussian with the same covariance matrix:

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}} \left(\frac{1}{|\mathbf{\Sigma}|^{1/2}} \right) \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu_k})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu_k}) \right\} \qquad (3)$$

General Posterior Distribution

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_k p(\mathbf{x}|C_k)p(C_k)} \tag{1}$$

$$= \frac{\pi_k \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_k})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_k})\right\}}{\sum_k \pi_k \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_k})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_k})\right\}} \tag{4-1}$$

$\frac{1}{(2\pi)^{D/2}}\left(\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\right)$ vanishes

$$= \frac{\pi_k \exp\left\{-\frac{1}{2}\left(\mathbf{x}^\mathbf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k}\right)\right\}}{\sum_k \pi_k \exp\left\{-\frac{1}{2}\left(\mathbf{x}^\mathbf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k}\right)\right\}} \tag{4-2}$$

$\mathbf{x}^\mathbf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x}$ is independent from C_k !

$$= \frac{\pi_k \exp\left\{\left(\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k}\right)\right\}}{\sum_k \pi_k \exp\left\{\left(\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k}\right)\right\}} \tag{4-3}$$

Two Classes - C_k and C_j

$$p(C_k|\mathbf{x}) = \frac{\pi_k \exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right\}}{\pi_j \exp\left\{\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j\right\} + \pi_k \exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right\}} \tag{5}$$

Using $\frac{a}{a+b} = \frac{1}{1+b/a}$ and $x = e^{\ln x}$

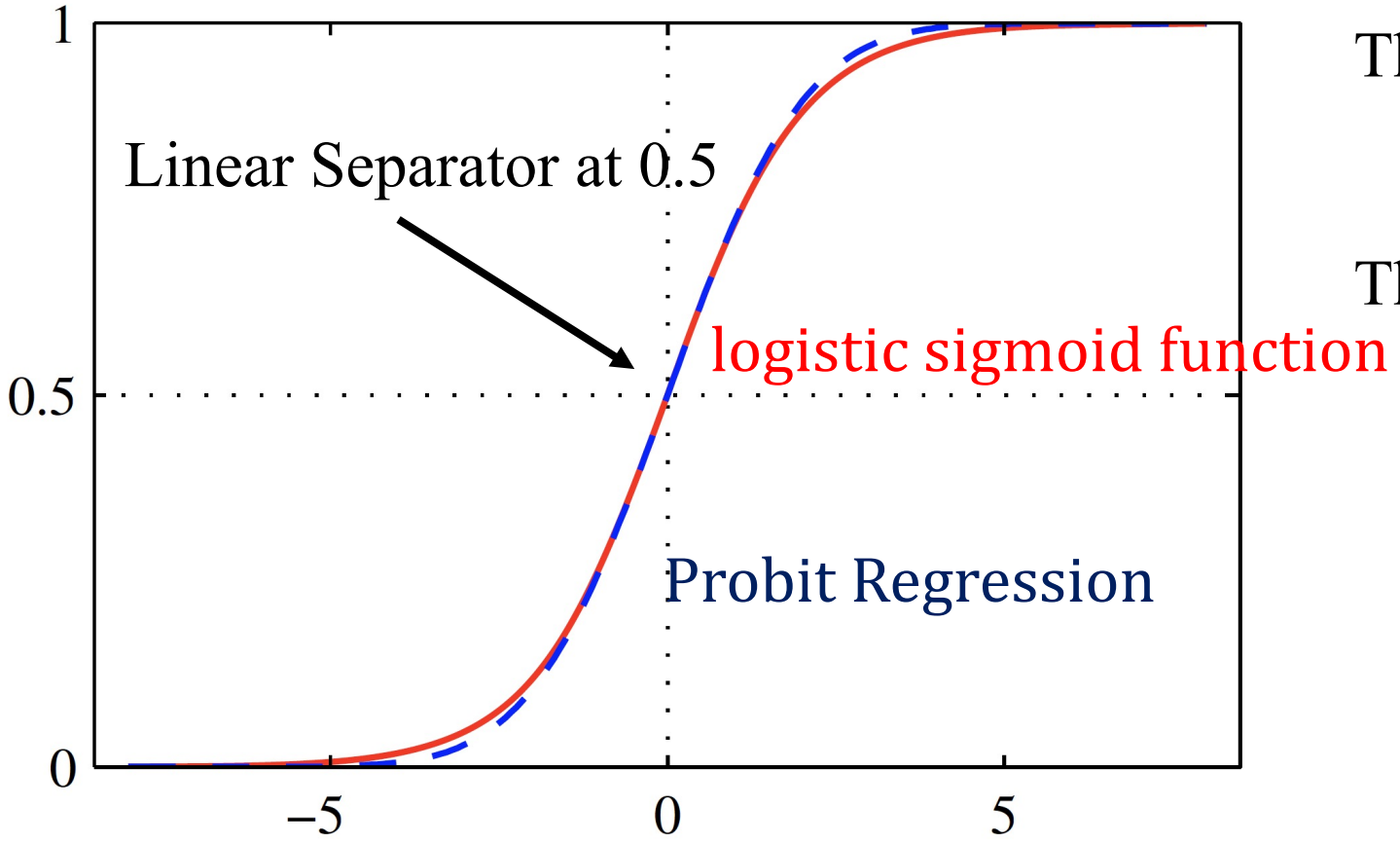
$$= \frac{1}{1 + \pi_j \exp\left\{\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j\right\} / \pi_k \exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right\}}$$

$$= \frac{1}{1 + \exp\left\{-\left[\underbrace{(\boldsymbol{\mu}_k^T - \boldsymbol{\mu}_j^T)\boldsymbol{\Sigma}^{-1}}_{=\boldsymbol{w}^T}\mathbf{x} \underbrace{- \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \ln\left(\frac{\pi_k}{\pi_j}\right)}_{=w_0}\right]\right\}} \tag{6}$$

Equation (6) simply becomes

$$= \frac{1}{1 + \exp\{-(\mathbf{w}^T\mathbf{x} + w_0)\}} \quad \textbf{the logistic sigmoid function} \tag{7}$$

Logistic sigmoid function



The logistic sigmoid function $\sigma(a)$ is

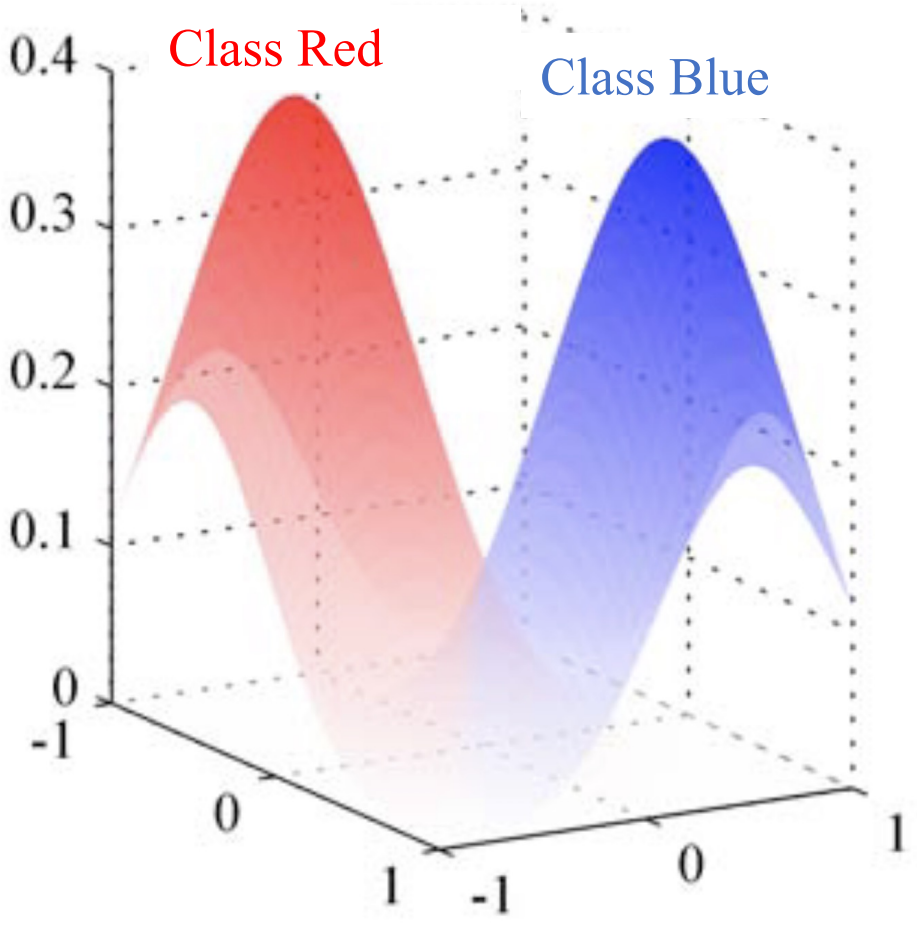
$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Then $p(C_k|\mathbf{x}) = \sigma(-\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0)$

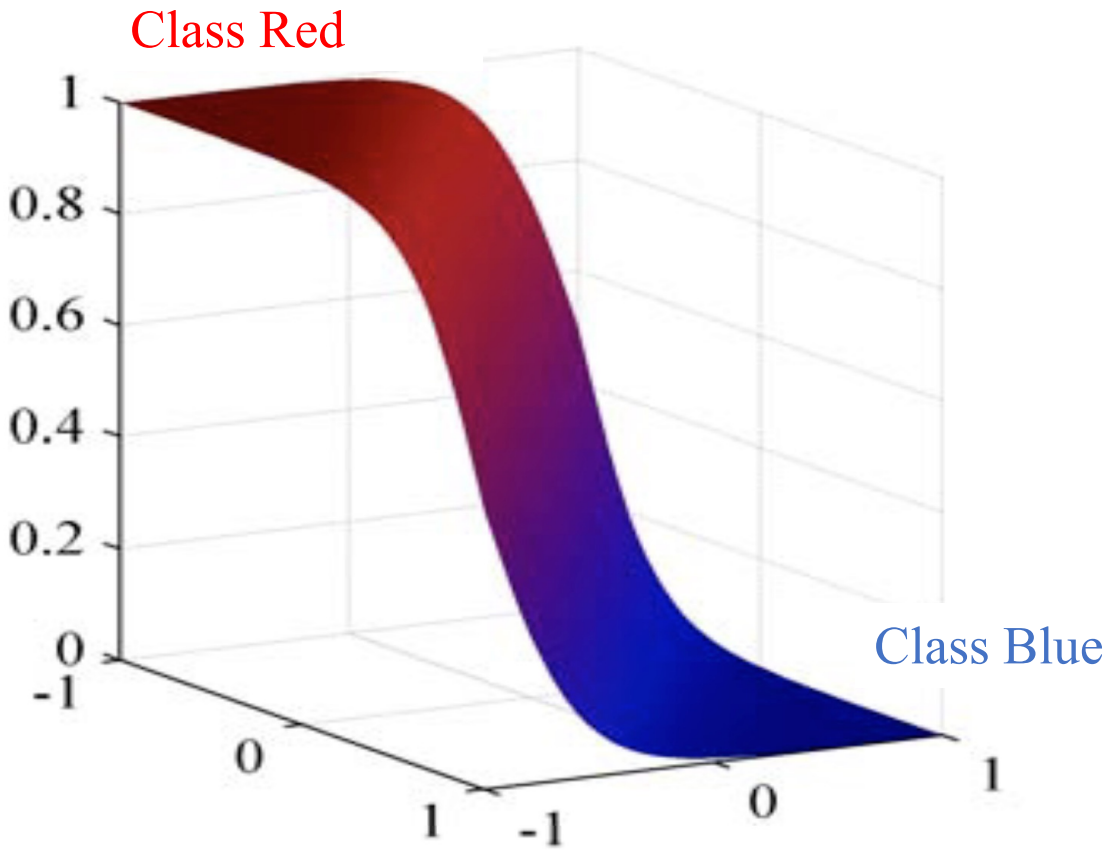
Logistic sigmoid function

Objective: find the probability of inputs likely being classified as Red.

Class Conditional



Posterior



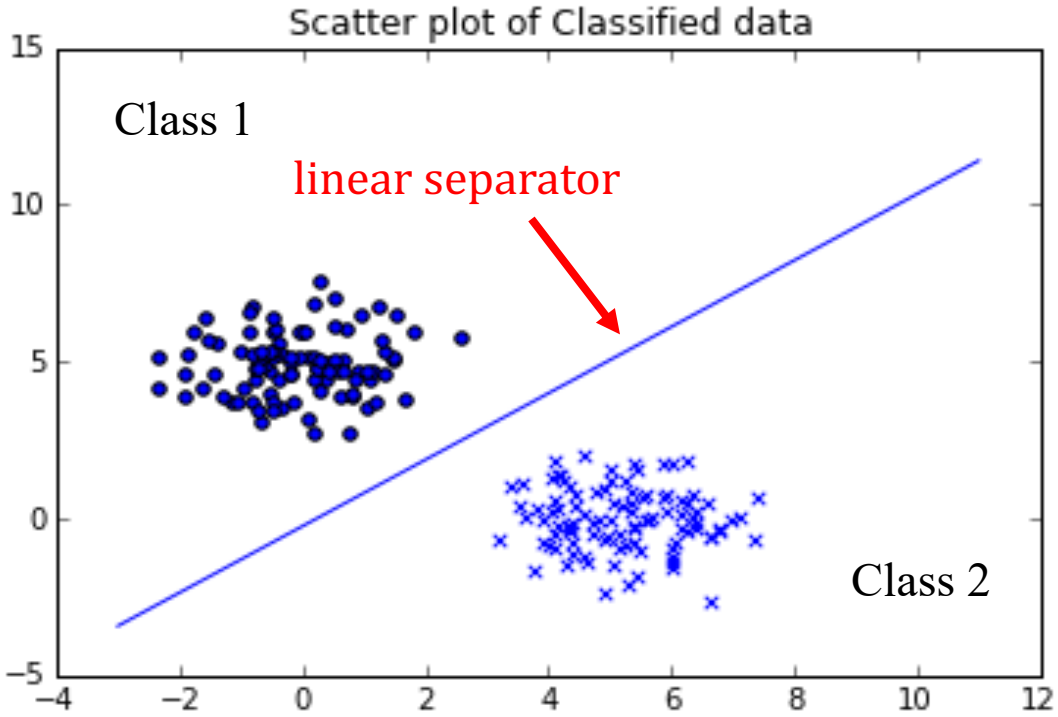
Two Class Prediction

- $best\ class = argmax_k P(C_k|\mathbf{x})$
- $= \begin{cases} c_1 & \sigma(a) \geq 0.5 \\ c_2 & \text{otherwise} \end{cases}$
- Class Boundary: $\sigma\left(w_k^T x + w_0\right) = 0.5$

$$\Longrightarrow e^{-(\mathbf{w^T x} + w_0)} = 1$$

$$\Longrightarrow \mathbf{w^T x} + w_0 = 0$$

$\therefore$ linear separator



Posterior Distribution for Multi-Class: $k > 2$

- The normalized exponential:

$$p(C_k|\mathbf{x}) == \frac{\pi_k \exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right\}}{\sum_k \pi_k \exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right\}} \tag{4-3}$$

$$= \frac{\exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln\pi_k\right\}}{\sum_j \exp\left\{\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \ln\pi_j\right\}}$$

$$\text{softmax} \Longrightarrow = \frac{e^{\mathbf{w_k^T}\bar{\mathbf{x}}}}{\sum_j e^{\mathbf{w_j^T}\bar{\mathbf{x}}}} \tag{8}$$

where $\mathbf{w_k} = \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}$ and $w_{0k} = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln\pi_k$

Softmax

- The posterior is a softmax (generalization of the sigmoid)
- softmax distribution:

$$p(C_k \mid \boldsymbol{x}) = \frac{\exp\left(f_k(x)\right)}{\sum_j \exp(f_j(x))}$$

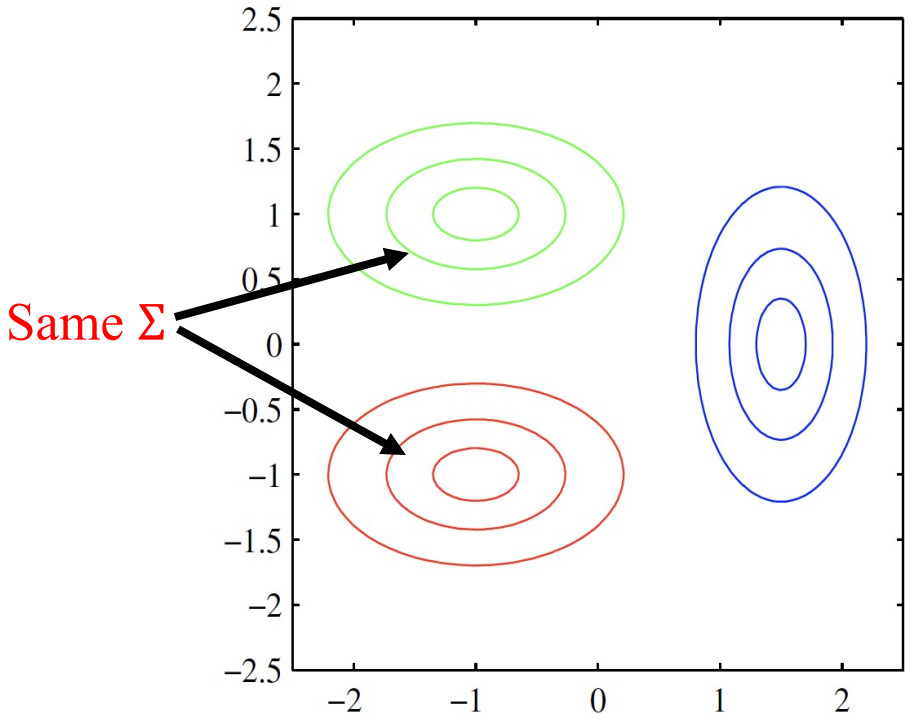
- argmax distribution

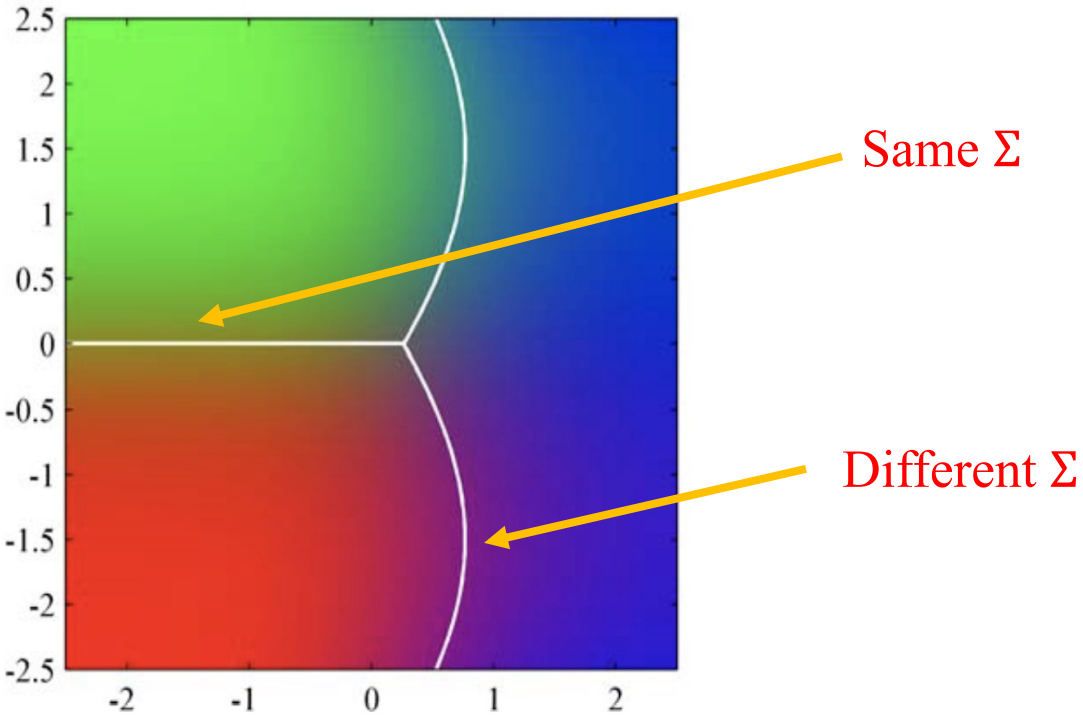
$$p(C_k \mid \boldsymbol{x}) = \begin{cases} 1 & \text{if } k = argmax_j\ f_j(x) \\ 0 & \text{otherwise} \end{cases}$$

$$= \lim_{base \to \infty} \frac{base^{f_k(x)}}{\sum_j base^{f_j(x)}} \tag{9}$$

$$\approx \frac{\exp\left(f_k(x)\right)}{\sum_j \exp(f_j(x))}$$

Softmax





$$p(c_k \mid \mathbf{x}) = \frac{\pi_k \exp\left\{-\frac{1}{2}\left(\mathbf{x}^{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{k}}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_{\mathbf{k}}^{-1}\mathbf{x} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_{\mathbf{k}}^{-1}\boldsymbol{\mu}_k\right)\right\}}{\sum_j \pi_j \exp\left\{-\frac{1}{2}\left(\mathbf{x}^{\mathbf{T}}\boldsymbol{\Sigma}_{\boldsymbol{j}}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_{\mathbf{j}}^{-1}\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_{\mathbf{j}}^{-1}\boldsymbol{\mu}_j\right)\right\}} \quad (4\text{-}3)$$

Depends on class number j

Parameter Estimation

- We have a parametric function form for the class-conditional densities:

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}}\left(\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\right)\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_k})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_k})\right\} \quad (3)$$

- We can estimate the parameters and the prior class probabilities using maximum likelihood.
  - Two class case with shared covariance matrix.
  - Training data:
  - $\{x_n, y_n\}, n = 1, \ldots, N$
  - $y_n = 1$ denotes class C_1; $y_n = 0$ denotes class C_2;
  - Priors: $p(C_1) = \pi, p(C_2) = 1 - \pi$
- For a data point x_n from class C_1, we have $y_n = 1$ and therefore:

$$p(\mathbf{x_n}, C_1) = p(\mathbf{x}|C_1)p(C_1) = \pi\mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_1}, \Sigma) \quad (9)$$

- For a data point x_n from class C_2, we have $y_n = 0$ and therefore:

$$p(\mathbf{x_n}, C_2) = p(\mathbf{x}|C_2)p(C_2) = (1-\pi)\mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_2}, \Sigma) \quad (10)$$

Maximum Likelihood Solution

- Assuming observations are drawn independently, the likelihood function is as below:

$$L(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}, \mathbf{y}|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} [p(\mathbf{x_n}, C_1)]^{y_n} [p(\mathbf{x_n}, C_2)]^{1-y_n} \quad (11\text{-}1)$$

Very common step

$$= \prod_{n=1}^{N} [\pi \mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_1}, \Sigma)]^{y_n} [(1-\pi)\mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_2}, \Sigma)]^{1-y_n} \quad (11\text{-}2)$$

$$\ln L(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{N} [y_n \ln \pi + y_n \ln \mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_1}, \Sigma) + (1-y_n) \ln(1-\pi) - (1-y_n) \ln \mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_2}, \Sigma)] \quad (12)$$

Maximum Likelihood Solution - parameter π

- We first maximize the log-likelihood with respect to π (set derivative to 0)

$$\frac{\partial}{\partial \pi}\left(\sum_{n=1}^{N}[y_n \ln \pi + (1-y_n)\ln(1-\pi)]\right) = 0 \tag{13}$$

- The maximum likelihood estimate of π is the fraction of points in class C_1.
- For multi-class: maximum likelihood estimate for $p(C_k)$ is given by the fraction of points in the training set in C_k.

$$\Rightarrow \pi = \frac{1}{N}\sum_{n=1}^{N} y_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2} \tag{14}$$

Maximum Likelihood Solution - parameter μ

- We them maximize the log-likelihood with respect to μ_1 (set derivate to 0)

$$\frac{\partial}{\partial \mu_1}\left(\sum_{n=1}^{N} y_n \ln \mathcal{N}(\mathbf{x}_n|\mu_1, \Sigma)\right) = 0$$

$$\frac{\partial}{\partial \mu_1}\left(\frac{1}{2}\sum_{n=1}^{N} y_n(\mathbf{x}_n - \mu_1)^T\Sigma^{-1}(\mathbf{x}_n - \mu_1) + \cdots\right) = 0 \tag{15}$$

$$\mu_1 = \frac{1}{N_1}\sum_{n=1}^{N} y_n \mathbf{x}_n \tag{16}$$

- The maximum likelihood estimate of μ_1 is the sample mean of all input xn in class C1.
- The maximum likelihood estimate of μ_2 is:

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - y_n)\mathbf{x}_n \tag{17}$$

Maximum Likelihood Solution - parameter Σ

Maximize the log-likelihood w.r.t. Σ, we obtain

$$\boldsymbol{\Sigma} = \frac{N_1}{N}\boldsymbol{S_1} + \frac{N_2}{N}\boldsymbol{S_2} \tag{18}$$

where

$$\boldsymbol{S_1} = \frac{1}{N_1}\sum_{n\in C_1}(\boldsymbol{x_n} - \boldsymbol{\mu_1})(\boldsymbol{x_n} - \boldsymbol{\mu_1})^T \tag{19}$$

$$\boldsymbol{S_2} = \frac{1}{N_2}\sum_{n\in C_2}(\boldsymbol{x_n} - \boldsymbol{\mu_2})(\boldsymbol{x_n} - \boldsymbol{\mu_2})^T \tag{20}$$

Exponential Family

- Exponential family members - e.g., Gaussian, Bernoulli, Poisson, Beta, Dirichlet, Gamma, etc…
- For all, the posterior is in the format of:

$$P(x|\theta_k) = \exp\left(\theta_k^T T(x) - A(\theta_k) + B(x)\right) \quad (21)$$

- The posterior is a sigmoid logistic linear function x

$$p(c_k|\mathbf{x}) = \sigma\left(\mathbf{w^T x} + w_0\right)$$

Probabilistic Discriminative Models

- Probabilistic Generative Model - Learn prior and posterior by maximum likelihood and find posterior using Bayesian Theorem.
- The posterior is known - either logistic sigmoid or softmax
- Probabilistic Discriminative Model - Learn posterior directly by maximum likelihood.

Logistic Regression

$$L(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}, \mathbf{y}|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} [\pi \mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_1}, \Sigma)]^{y_n} [(1-\pi)\mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_2}, \Sigma)]^{1-y_n} \quad (11\text{-}2)$$

- The log-likelihood function is

$$\ln L(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{N} [y_n \ln \pi + y_n \ln \mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_1}, \Sigma) + (1-y_n) \ln(1-\pi) - (1-y_n) \ln \mathcal{N}(\mathbf{x_n}|\boldsymbol{\mu_2}, \Sigma)] \quad (22)$$

- Use the negative log-likelihood function (cross-entropy error function) to find the parameters

$$E(\mathbf{w}) = -\ln L(\mathbf{w}) = -\sum_{n=1}^{N} \left[y_n \ln \sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right) + (1-y_n) \ln\left(1-\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)\right)\right] \quad (23)$$

- The goal is to estimate the continuous posterior function.
- Logistic regression is a form of classification.

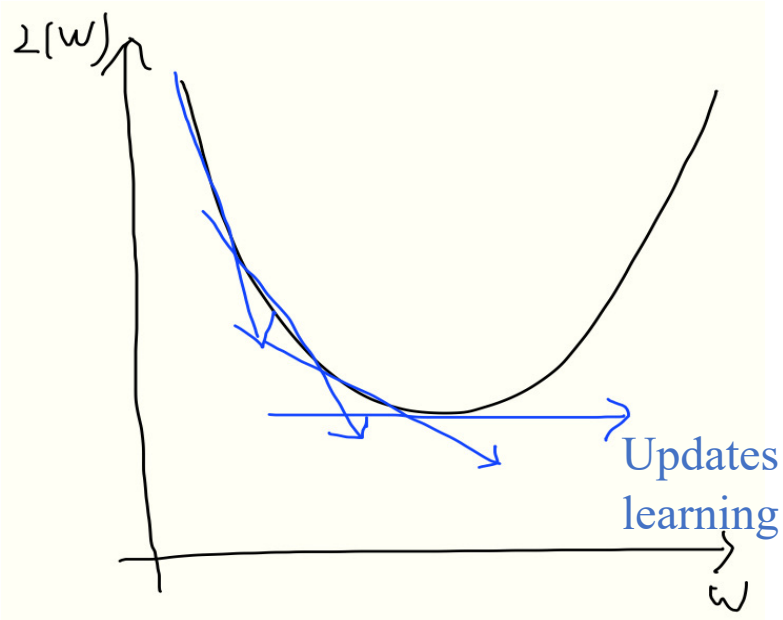
Maximum Likelihood

$$-\frac{\partial E(\mathbf{w})}{\partial w} = -\sum_n y_n\left(\frac{\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)\left(1-\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)\right)\bar{\boldsymbol{x}}_n}{\sigma(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}})} - \sum_n \frac{(1-y_n)\left(1-\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)\right)\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)(-\bar{\boldsymbol{x}}_n)}{1-\sigma(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}})} \right. \quad (24\text{-}1)$$

$$0 = -\sum_n y_n\bar{x}_n - \sum_n y_n\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)(\bar{\boldsymbol{x}}_n) + \sum_n \sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)\bar{x}_n + \sum_n y_n\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right)(\bar{\boldsymbol{x}}_n)$$

$$= \sum_n \left[\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right) - y_n\right]\bar{\boldsymbol{x}}_n \quad (24\text{-}2)$$

- Can we estimate **w**? No
- Then how? Use an iterative method



Updates by the linear gradient and the learning rate

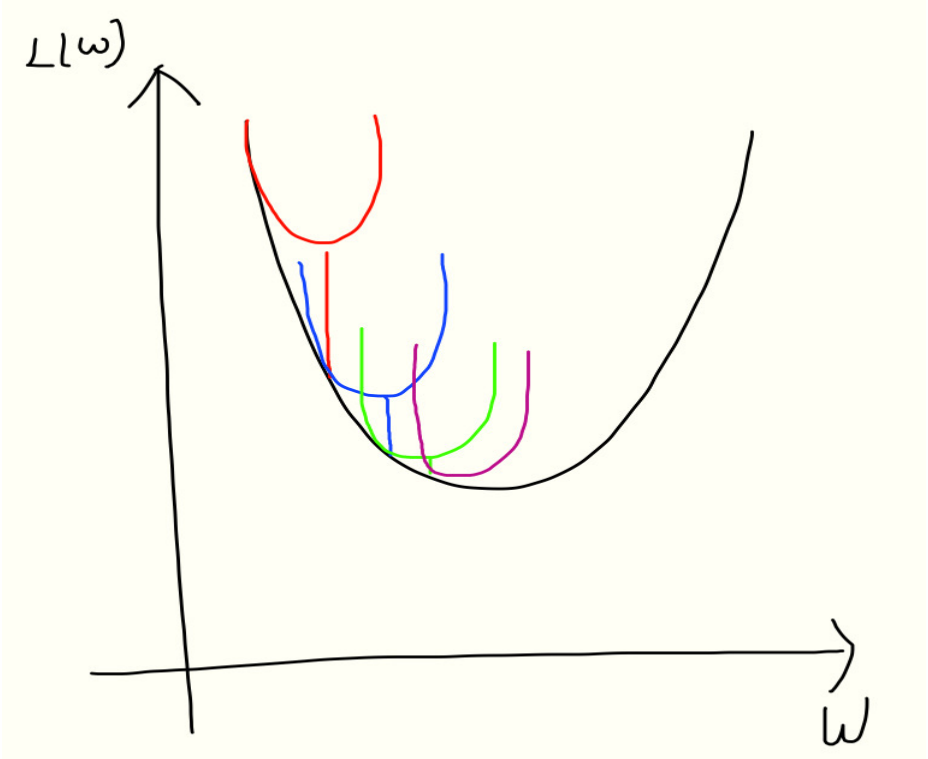
Newton's Method

Iterative reweighted least square:

$$\mathbf{w} \leftarrow \mathbf{w} - \mathbf{H}^{-1}\nabla E(\mathbf{w}) \qquad (25)$$

where ∇E is the gradient (column vector) and H is the Hessian (matrix)

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial^2 w_0} & \cdots & \frac{\partial^2 E}{\partial w_0 \partial w_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_m \partial w_0} & \cdots & \frac{\partial^2 E}{\partial^2 w_m} \end{bmatrix} \qquad (25\text{-}1)$$



Hessian

$$\mathbf{H} = \nabla(\nabla \ln L(w)) \tag{26}$$

$$= \nabla\left(\sum_n \left[\sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}\right) - y_n\right]\bar{\boldsymbol{x}}_{\boldsymbol{n}}\right)$$

$$= \sum_{n=1}^{N} \sigma\left(\mathbf{w}^{\mathrm{T}}\bar{\mathbf{x}}_n\right)\left(1 - \sigma\left(\mathbf{w}^{\mathbf{T}}\bar{\mathbf{x}}_n\right)\right)\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^{\mathrm{T}}$$

$$= \bar{\boldsymbol{X}}\boldsymbol{R}\bar{\boldsymbol{X}}^{T} \tag{27}$$

where $\mathbf{R} = \begin{bmatrix} \sigma_1(1-\sigma_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_N(1-\sigma_N) \end{bmatrix}$

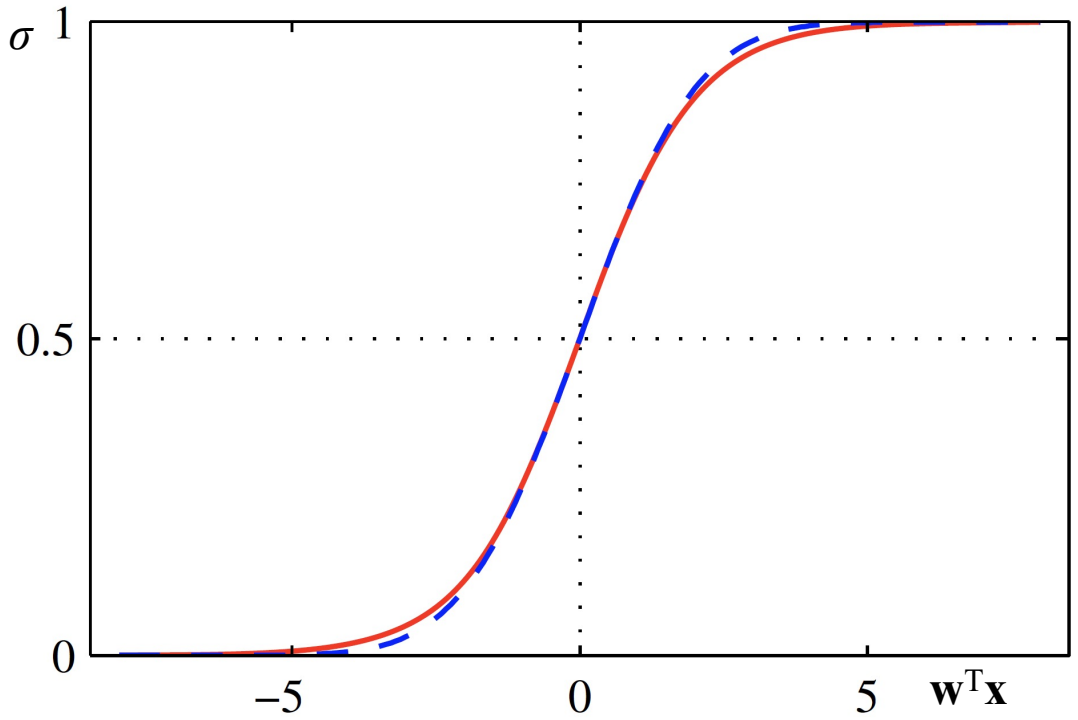
Issues

Issue: Maximum likelihood can exhibit overfitting: logistic regression can classify each data point arbitrarily well when data is small.

Reasons:

As the posterior gets toward 1, $\boldsymbol{w}^T\boldsymbol{x}\rightarrow\infty$ and therefore the magnitude of w must be infinite.

If $\sigma=1$, then $(1-\sigma)=0$ so as $R=0$ and therefore H becomes singular.



Regularization

- We can penalize large weights
- How? Add the penalty term λ.

Penalty Term

$$\min_{w} E(w) + \frac{1}{2}\lambda\left|\left|w\right|\right|^2 \tag{28}$$

$$= \min_{w}\left\{-\sum_{n=1}^{N}\left[y_n \ln \sigma\left(\mathbf{w^T}\bar{\mathbf{x}}\right) + (1-y_n)\ln\left(1-\sigma\left(\mathbf{w^T}\bar{\mathbf{x}}\right)\right)\right]\right\} + \frac{1}{2}\lambda\mathbf{w^T}\mathbf{w} \tag{28-1}$$

- Hessian: $$H = \bar{\mathbf{X}}\mathbf{R}\bar{\mathbf{X}}^T + \lambda\mathbf{I} \tag{29}$$
- The term $\lambda\mathbf{I}$ ensures that H is not singular (eigenvalues $\geq \lambda$)

Estimation with Maximum A Posteriori(MAP) from Bayes Rule

Recall,

- Bayesian view: probabilities provide a quantification of uncertainty. Before observing the data, the assumptions about *w* are captured in the form of a prior probability distribution $P(\boldsymbol{w})$. The effect of the observed data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is expressed by $P(D|\boldsymbol{w})$.
- Bayes' theorem:

$$P(\boldsymbol{w}|D) = \frac{P(D|\boldsymbol{w})P(\boldsymbol{w})}{P(D)}$$

- Bayes' theorem in words: posterior $\propto$ likelihood $\times$ prior

Estimation with Maximum A Posteriori(MAP) from Bayes Rule

$$\boldsymbol{w} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, P(w|(t_1, x_1), \dots (t_n, x_n)) = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{P\left(w\middle|(t_1, x_1), \dots (t_n, x_n)\right)P(w)}{P((t_1,x_1),\dots(t_n,x_n))}$$

- Independence & chain rule of probability.
- x_i is independent of w & $P(\cdot)$ constant and can be dropped.
- $P(t_i, x_i|w) = P(t_i|x_i, w)P(x_i|w) = P(t_i|x_i, w)P(x_i) = P(t_i|x_i, w)$

$$= \underset{\boldsymbol{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(t_i|x_i, \boldsymbol{w})\right] P(\boldsymbol{w}) = \underset{\boldsymbol{w}}{\operatorname{argmax}} \sum_{i=1}^{n} \log[P(t_i|x_i, w)] + \log P(w)$$

- $w \sim N(0, \tau^2)$ assumption

$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(x_i^T w - t_i\right) + \frac{1}{\tau^2} w^T w = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(x_i^T w - y_i\right)^2 + \lambda ||w||^2$$

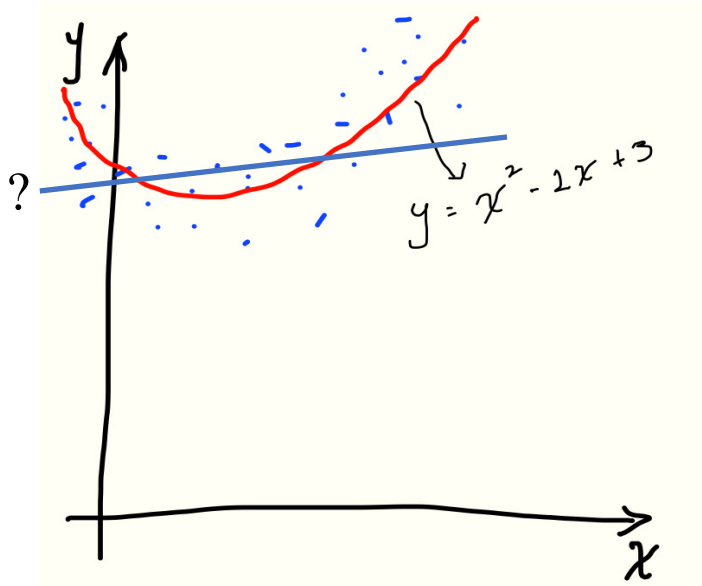
- $\lambda = \sigma^2/n\tau^2$

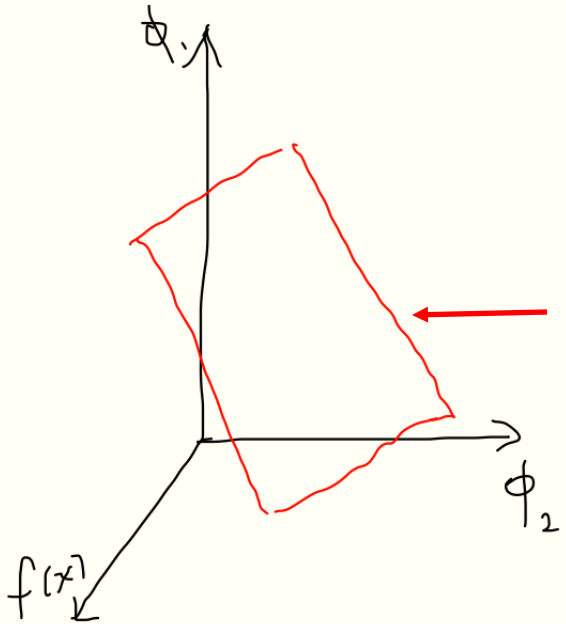
* This objective is known as Ridge Regression.

Generalized Linear Models

- Can we do non-linear classification?
- How can we do so? We can map inputs to a different space and so linear classification in that space.

Basis Functions





Use non-linear basis functions. Examples of basis functions are

polynomial: $\phi_j(x) = x^j$

Gaussian: $\phi_j(x) = e^{-\frac{(x-\mu_j)^2}{2s_j^2}}$

Sigmoid: $\phi(x) = \sigma\left(\frac{x-\mu_j}{s_j}\right)$

Other many as long as they seem appropriate.

The new space we have is $H = \{x \to \sum_i w_i \phi_i(x) \,|w_i \in \mathbb{R}\}$

Linear hyperplane in the new space

CS 559: Kernel Methods, Supportive Vector Machine, & Empirical Risk

Lecture 6

Outline

- Kernel Methods
- SVM
- Empirical Risk

Kernel Methods

Kernel Function

- Given: There are a large set of fixed linear basis functions.
- Problem: It is all depends on how complex the basis functions are.
- Solution: Use "dual trick" that is depends on the amount data than the function.

- We have a set of basis functions $\phi(\boldsymbol{x})$ that map inputs x to a feature space.
- This feature space appears in the dot product $\phi(x)^T\phi(x)$ of input pairs, x, x'.
- The kernel function $k(x, x') = \phi(x)^T\phi(x')$ in feature space.
  - We are interested in kernel function not the set of basis functions.

Dual Representations

The linear regression objective is the error.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}[w^T\phi(x_n) - y_n]^2 + \frac{1}{2}\lambda\mathbf{w}^\mathbf{T}\mathbf{w} \quad (1)$$

We set the gradient to 0.

$$\nabla E(\mathbf{w}) = \sum_{n}(\mathbf{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n) + \lambda\mathbf{w} = 0 \quad (2)$$

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n}(\mathbf{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n) \quad (3)$$

w is linear combination of inputs in feature space

$$\{\phi(x_n)|1 \le n \le N\}$$

Dual Representations

Substitute $\mathbf{w} = \mathbf{\Phi a}$

Where $\Phi = [\phi(x_1), \phi(x_2), \ldots, \phi(x_N)]$

$$a = \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} \text{ and } a_n = -\frac{1}{\lambda}(\mathbf{w}^T \phi(\boldsymbol{x}_n) - y_n)$$

Dual objective: minimize E with respect to a

$$E(a) = \frac{1}{2} a^T \Phi^T \Phi \Phi^T \Phi a - a^T \Phi^T \Phi y + \frac{y^T y}{2} + \frac{\lambda}{2} a^T \Phi^T \Phi a \quad (4)$$

Gram Matrix

The Gram Matrix: $K = \Phi^T \Phi$

Substitution K into (6-1): $\frac{1}{2}\sum_{n=1}^{N}[w^T\phi(x_n) - y_n]^2 + \frac{1}{2}\lambda \mathbf{w^T w}$ where $\boldsymbol{w} = \Phi a$

$$E(a) = \frac{1}{2}a^T KKa - a^T Ky + \frac{y^T y}{2} + \frac{\lambda}{2}a^T Ka \quad (5)$$

$$\nabla E(a) = KKa - Ky + \lambda Ka = 0$$

$$Ky = K(K + \lambda I)a$$

$$a = (K - \lambda I)^{-1} y$$

Prediction: $y_* = \phi(x_*)^T w = \phi(x_*)^T \Phi a = K(x_*, X)(K + \lambda I)^{-1} y$ (6)

where (X,y) is the training set and (x_*, y_*) is a test instance

- Primal Solution: depends on # of basis functions
- Dual solution: depends on amount of data
  - Advantage: can use very large # of basis functions
    Just need to know K

Constructing Kernels

Possibilities:

- Find mapping Φ to feature space and let $K = \phi^T \phi$
- Directly specify K

A valid kernel must be positive semi-definite:

- k must factor into the product of a transpose matrix by itself
- or all eigenvalues ≥ 0
- problem? It is not always easy to verify.

Rules to construct Kernels

Let $k_1(x, x')$ and $k_2(x, x')$ be valid kernels.

The following kernels are also valid:

$= ck_1(x, x')$ c is the constant > 0

$= f(x)k_1(x, x')f(x')$ f is the function

$= q\big(k_1(x, x')\big)$ q is the polynomial

$= \exp\big(k_1(x, x')\big)$

$k(x, x') = k_1(x, x') + k_2(x, x')$

$= k_3\big(\phi(x), \phi(x')\big)$

$x^T A x'$ A is symmetric positive semi-definite

$= k_a(x_a, x'_a) + k_b(x_b, x'_b)$ k_a & k_b are valid kernel

$= k_a(x_a, x'_a)k_b(x_b, x'_b)$

Other Application? sets, strings, graphs, etc…

Example -

Lodhi, Sunders, Shawe-Tylor, Christianini, Watkins,

Text Classification Using Sting Kernels, JMLR, p. 419-444, 2002

Kernel Construction Example

Consider a 2-D input space $\boldsymbol{x}^{\boldsymbol{T}} = (x_1, x_2)^T$ and $\boldsymbol{z} = (z_1, z_2)$ and suppose we are going to map into a feature space in the degree of 2 using

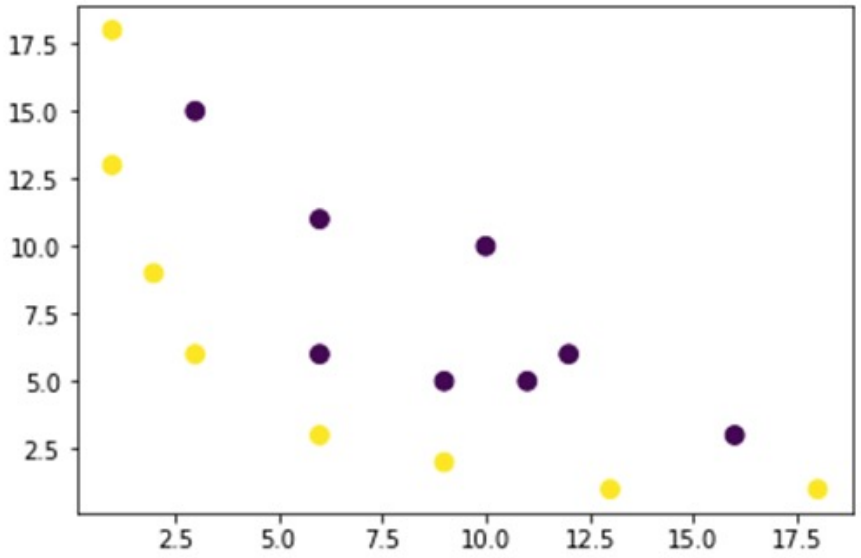
$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z})^2$$

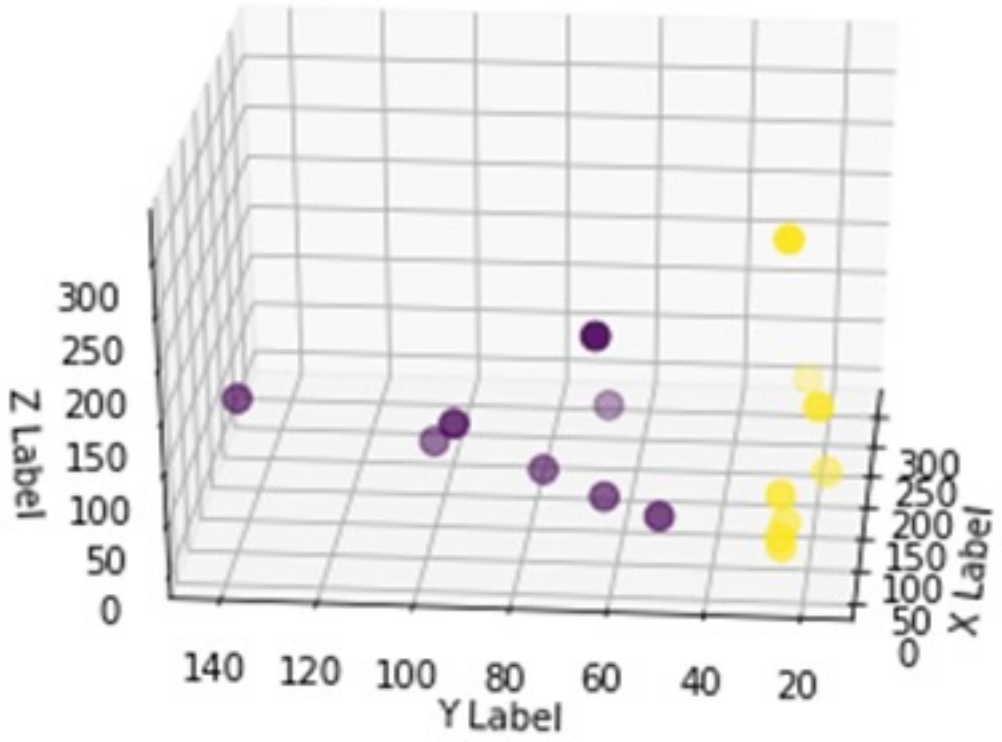
$$\begin{aligned} k(x, z) = (x^T z)^2 &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right)\left(z_1^2, \sqrt{2} z_1 z_2, z_2^2\right) \\ &= \phi(x)^T \phi(z) \end{aligned}$$

The feature mapping comprises all possible second order terms.

Which kernel functions to use? We can always use the functions whose validities are already confirmed.

Kernel Construction Example





Kernel Construction Example

Verifying the validity of kernels: $k(\boldsymbol{x}, \boldsymbol{x}') = ck_1(\boldsymbol{x}, \boldsymbol{x}')$.

- Start with the basic function - $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$
- It follows that $ck_1(\boldsymbol{x}, \boldsymbol{x}') = c\boldsymbol{u}(\boldsymbol{x})^T \boldsymbol{u}(\boldsymbol{x}')$ where $\boldsymbol{u}(\boldsymbol{x}) = \sqrt{c}\phi(\boldsymbol{x})$ and $\boldsymbol{u}(\boldsymbol{x}') = \sqrt{c}\phi(\boldsymbol{x}')$
- So $ck_1(\boldsymbol{x}, \boldsymbol{x}')$ can be expressed as the scalar product of feature vectors and is a valid kernel.

Supportive Vector Machine (SVM)

Supportive Vector Machine

- Maximum Margin Classifiers
- Overlapping class distributions
- Classification
- Regression
- Kernel SVM - Regression

SVM

$$y(x) = w^T \phi(x) + b$$

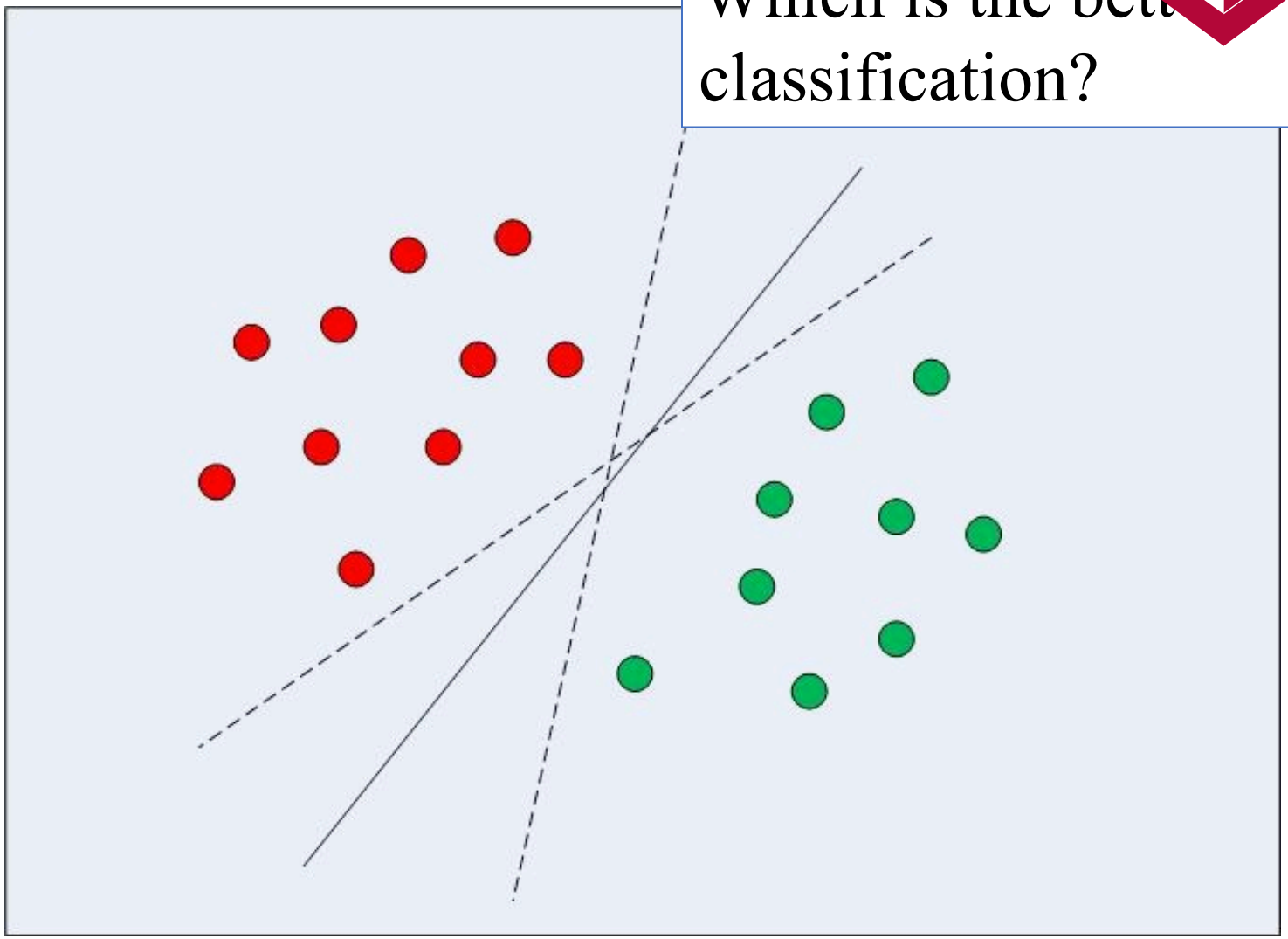
- $\phi(x)$ a fixed feature-space transformation
- b the bias parameter
- training set - linearly separable in feature space
- Data: $\{(x_i, t_i) | n = 1, \ldots, n\}$
- $t_i \in \{-1,1\}$

SVM vs. Perceptron Algorithm

Perceptron Algorithm

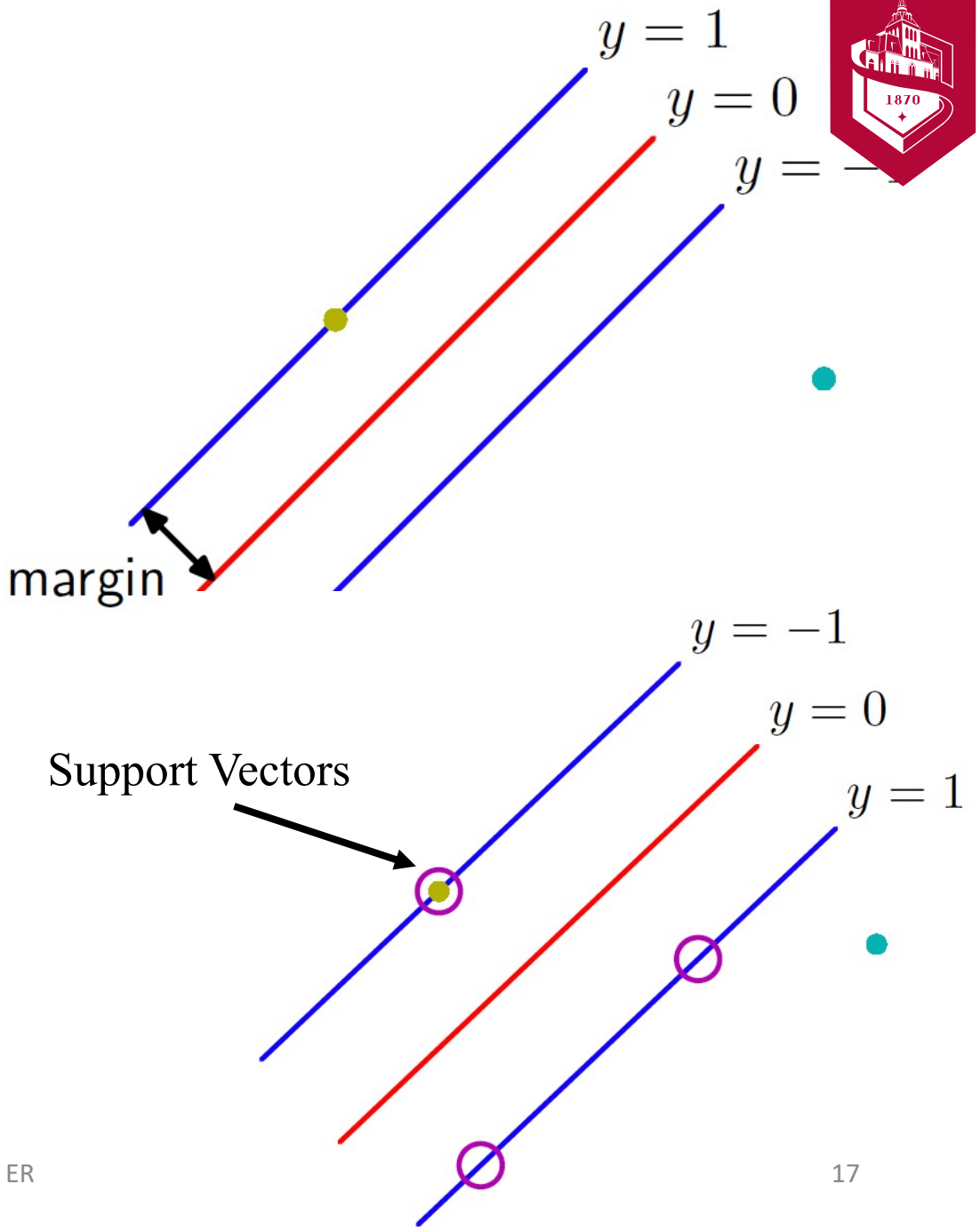
- guarantees to find a solution in a finite number of steps
- initial value for w and b starts from arbitrary chosen value..

If there are multiple solutions of all of which classify the training data set exactly, then we should try to find the one that give **the smallest generalization error!**



SVM

- An extension of the Perceptron developed by Rosenblatt in 1958.
- Concept of the margin - the smallest distance between the decision boundary and any of the samples.
- Choose the max margin
- Using the optimal boundary, determine the best ***hyperplane*** by minimizing the probability of error relative to the learned density mode.
- hyperplane becomes independent of data points that are not ***support vectors (SV)***.



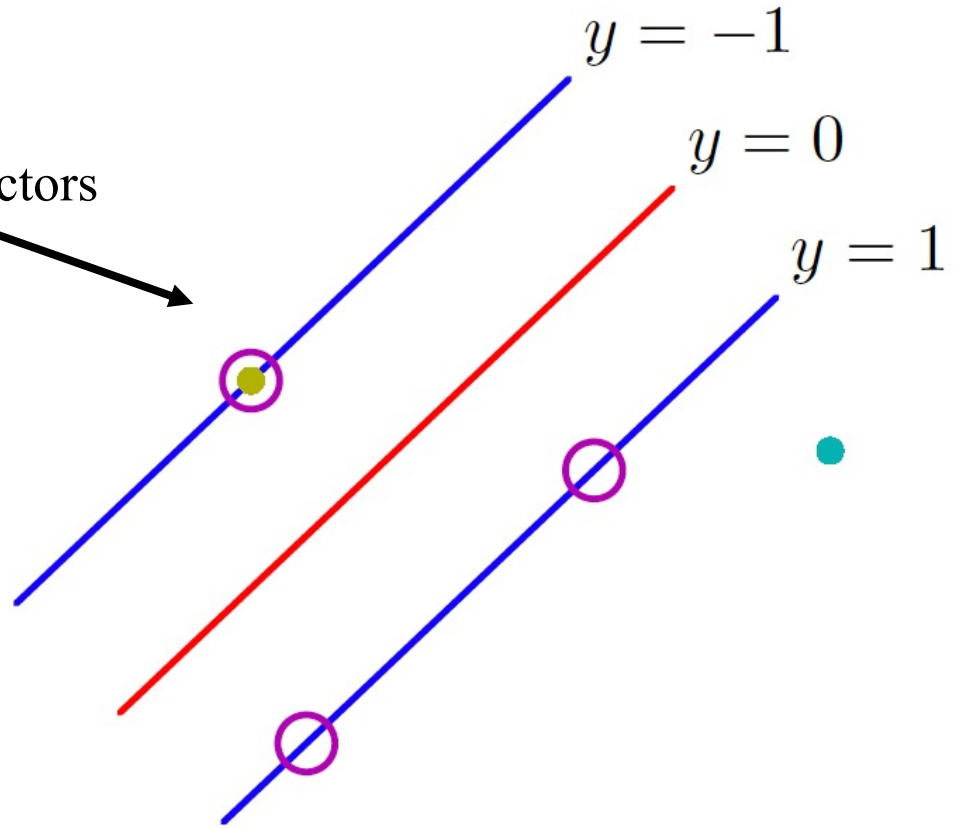
SVM

Hyperplane is $y(x) = 0$

Right classification $t_n y(x_n) > 0$

$$\frac{t_n y(\boldsymbol{x_n})}{||\boldsymbol{w}||} = \frac{t_n[\boldsymbol{w^T}\phi(\boldsymbol{x_n}) + b]}{||\boldsymbol{w}||}$$

Support Vectors



Optimize the parameters w and b to maximize the distance.

Maximum margin solution is

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n \left[t_n \left(\boldsymbol{w^T}\phi(\boldsymbol{x_n}) + b \right) \right] \right\}$$

w does not depend on n. canonical representation of the decision hyperplane.

SVM

Direction solution is quite complex, we convert it into an equivalent problem (rescale)

$$t_n(w^T\phi(x)+b)=1$$

for the point that is closest to the surface. All data points will satisfy the constrains

$$t_n(w^T\phi(x)+b)\geq 1$$

once the margin has been maximized there will be at least two active constraints.

So we maximize $||w||^{-1}$ and this is same as minimizing $||w||^2$

$$\underset{w,b}{\operatorname{argmin}}\frac{1}{2}||w||^2$$

Lagrange multipliers

We introduce Lagrange Multipliers $a_n \geq 0$

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\left|\left|w\right|\right|^2 - \sum_{n=1}^{N} a_n \left\{t_n\left(\boldsymbol{w}^T \phi(\boldsymbol{x_n}) + b\right) - 1\right\} \quad (7)$$

- a strategy for finding the local maxima and minima of a function subject to equality constraints

Derivative:

$$w = \sum_{n=1}^{N} a_n \, t_n \phi(x_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

Dual Representation

Dual representation of the maximum margin problem

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N} a_n\, a_m t_n t_m k(x_n, x_m) \qquad (8)$$

Use the kernel function.

This takes the form of a quadratic programming problem

$$a_n \geq 0$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

Prediction

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x_n}) + b \tag{9}$$

Satisfies the following conditions:

$$a_n \geq 0$$
$$t_n y(x_n) - 1 \geq 0$$
$$a_n\{t_n y(x_n) - 1\} = 0$$

for point with $a_n = 0$ will not appear in the sum and plays no role making predictions for the new point.

The remaining data points are called support vectors because they satisfy $t_n y(x_n) = 1$ and correspond to points that lie on the maximum margin hyperplanes in feature space.

Prediction

Using Eq (9)

$$t_n\left(\sum_{m\in S} a_m t_m k(x_n, x_m) + b\right) = 1 \tag{10}$$

S denotes the set of indices of the support vectors (SV).

Making a use of $t_n^2 = 1$ averaging over all SV and solve for b

$$b = \frac{1}{N_s}\left(\sum_{n\in S}\left(t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)\right)\right) \tag{11}$$

N_s is the total number of SV.

The maximum margin classifier in terms of the minimization of an error function

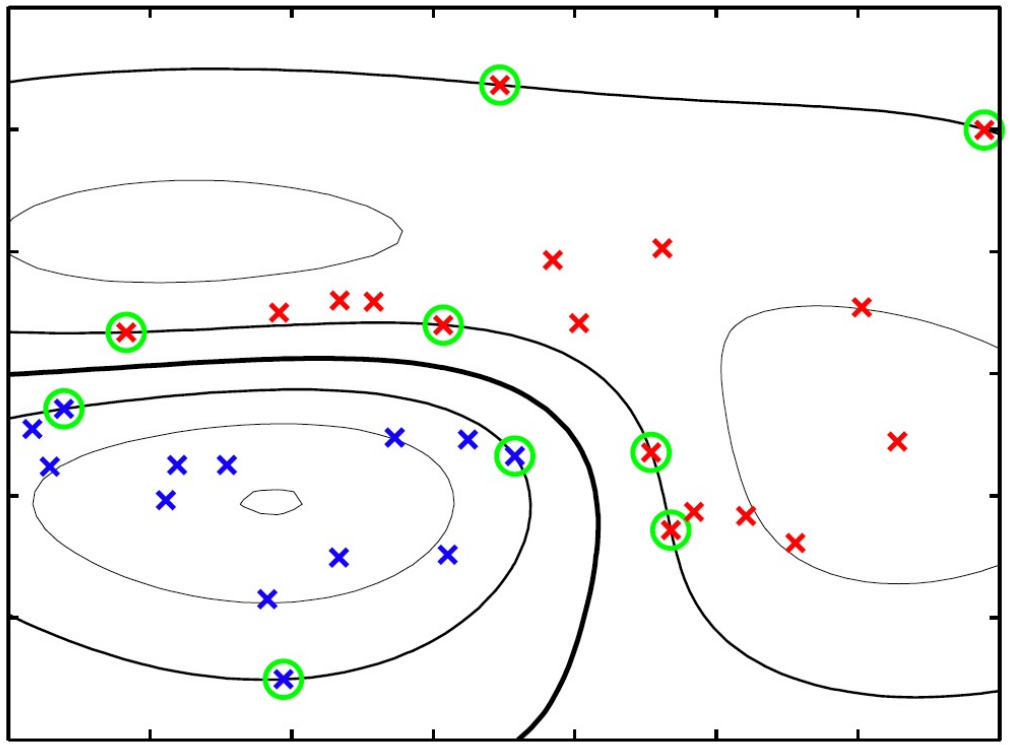
$$\sum_{n=1}^{N} E_\infty(y(x_n)t_n - 1) + \lambda\left|\left|w\right|\right|^2 \tag{12}$$

Prediction

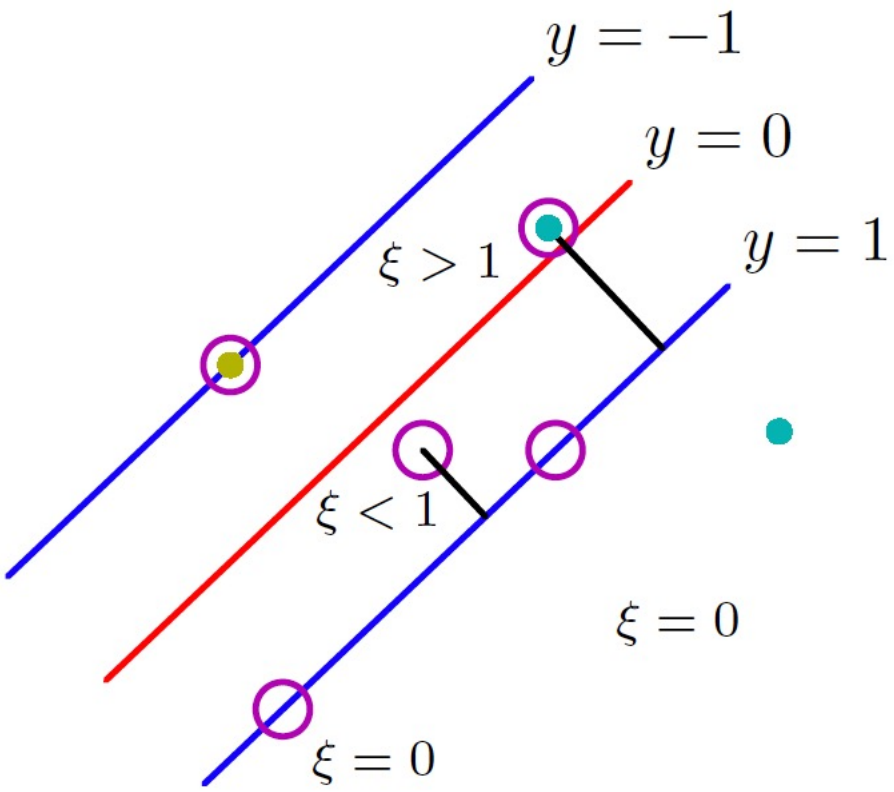
The maximum margin classifier in terms of the minimization of an error function

$$\sum_{n=1}^{N} E_{\infty}(y(x_n)t_n - 1) + \lambda\left|\left|w\right|\right|^2 \qquad (13)$$

where E is a function that is zero if $z \geq 0$ and ∞ otherwise to ensure that the constraints satisfies.



Overlapping Class Distributions



The class-conditional distributions may overlap, in which case exact separation of the training data can lead to poor generalization.

Need to modify SVM

Modify so that data points are allowed to be on the wrong side of the margin boundary, but with penalty that increases with the distance from that boundary.

Overlapping Class Distributions – soft SVM

Make the penalty a linear function of the distance.

slack variable $\xi_n \geq 0$

$\xi = 0$ for data points that are on or inside the correct margin boundary and $\xi = |t_n - y(x_n)|$ for other points.
On the decision boundary, $y(x_n) = 0$ and have $\xi = 1$ and with $\xi > 1$ will be misclassified

$$t_n y(x_n) \geq 1 - \xi_n$$

Goal is to maximize the margin while **softly penalizing points** that lie on the wrong side of the margin boundary

Overlapping Class Distributions – soft SVM

To minimize the margin while softly penalizing points that lie on the wrong side of the margin boundary, we need to minimize

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}||w||^2 \quad (14)$$

where parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.

- Misclassified points have $\xi_n > 1$ that $\sum_n \xi_n$ is an upper bound on the number of misclassified points!
- $C = 1/\lambda$ and if $C \to \infty$, we can recover SVM for separable data.

$$L(w, b, \xi, a, \mu) = \frac{1}{2}||w||^2 + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N} a_n\{t_n y_n(x_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n \quad (15)$$

Lagrange Multipliers - $\{a_n \geq 0\}$ and $\{\mu_n \geq 0\}$:

1) $a_n = 0$
2) $t_n y_n(x_n) - 1 + \xi_n \geq 0$
3) $a_n\{t_n y_n(x_n) - 1 + \xi_n\} = 0$
4) $\mu_n \geq 0$
5) $\xi_n \geq 0$
6) $\mu_n\xi_n = 0$

Overlapping Class Distributions – soft SVM

Optimize out w, b, and ξ

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n \phi(x_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_N} = 0 \Rightarrow a_n = C - \mu_n$$

$0 \leq a_n \leq C$ Box Constraints

Using these results to eliminate from the Lagrangian, the dual Lagrangian forms as below:

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} a_n \, a_m t_n t_m k(x_n, x_m) \Big) \qquad (20)$$

where $a_n, \mu_n \geq 0$ implies that $a_n \leq C$. Then we need to maximize respect to $\{a_n\}$ subject to

$$0 \leq a_n \leq C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

Overlapping Class Distributions – soft SVM

Solution

$$t_n\left(\sum_{m\in S} a_m t_m k(x_n, x_m) + b\right) = 1 \tag{21}$$

$$b = \frac{1}{N_{\mathcal{M}}}\left(\sum_{n\in\mathcal{M}}\left(t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)\right)\right) \tag{22}$$

where M is the set of indices of data points having $0 < a_n < C$.
What condition needs on ξ_n?

Overlapping Class Distributions – soft SVM

IF we wish to use the SVM as a module in a larger probabilistic system

Platt (2000) has proposed fitting a logistic sigmoid to the outputs of a trained SVM.

$$p(t = 1|x) = \sigma(Ay(x) + B)$$

A & B are found by minimizing the cross-entropy error function defined by a training set consisting of pairs of values y and t.

- The data used to fit the sigmoid needs to be independent to avoid sever over-fitting.
- Give a poor approximation to the posterior prob.

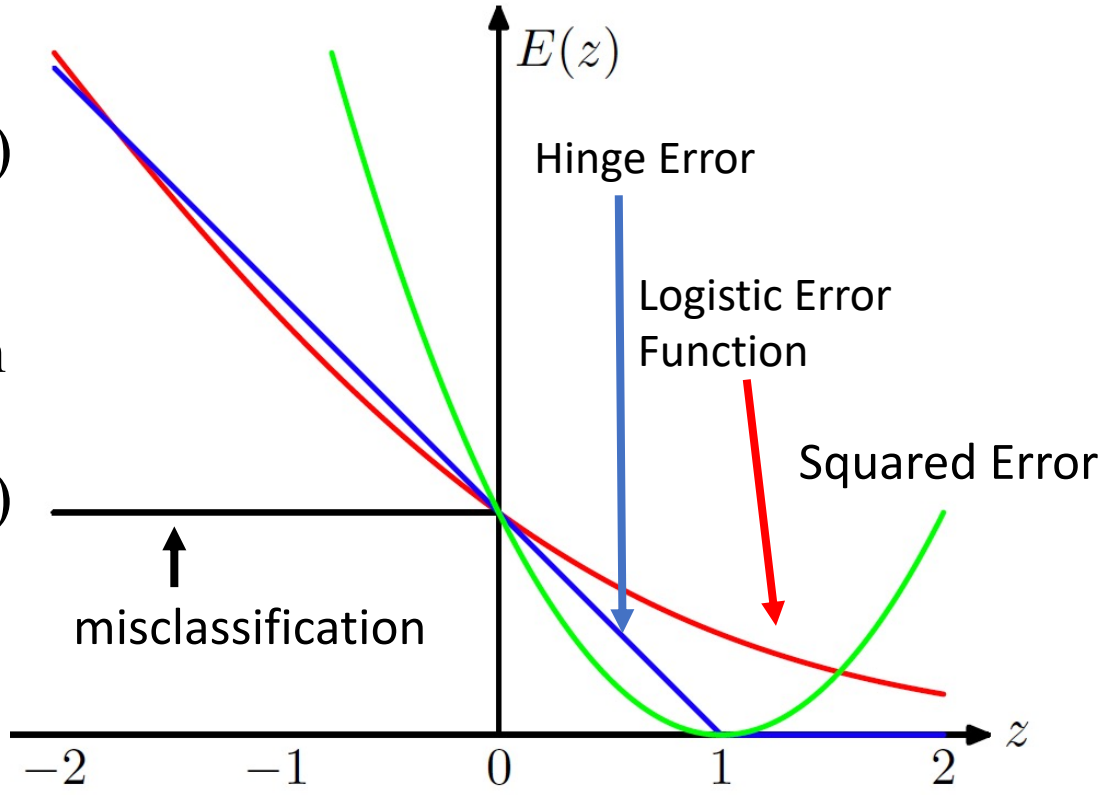
Relation to logistic regression – Hinge Error

Update Eq (7)

$$\sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda ||w||^2 \qquad (6\text{-}42)$$

where $\lambda = (2C)^{-1}$ and E_{SV} is the **hinge error** function

$$E_{SV}(y_n t_n) = [1 - y_n t_n]_+ \qquad (6\text{-}43)$$



Relation to logistic regression

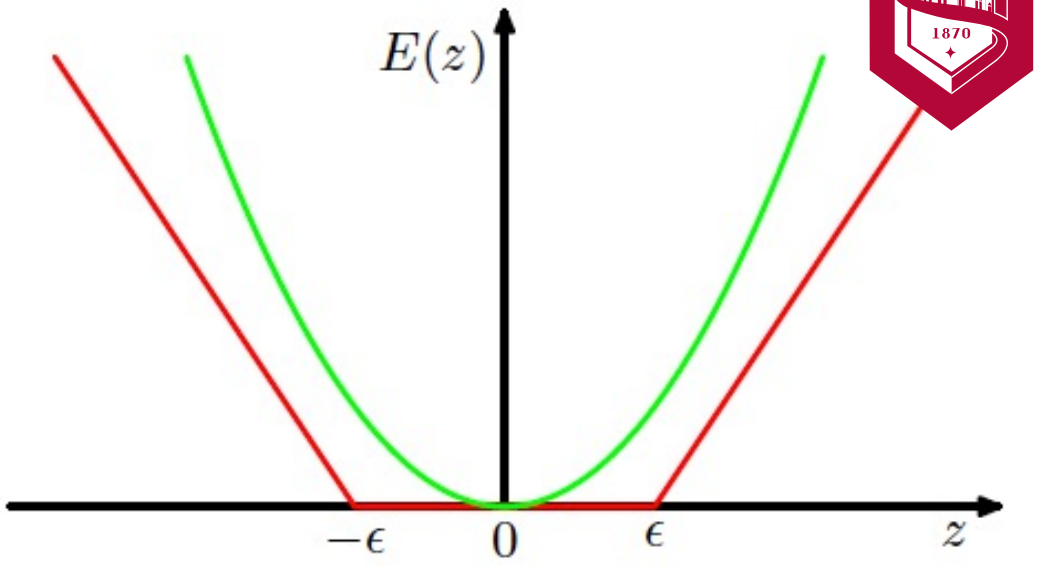
	SVMs	Logistic Regression
Loss Function	Hinge Loss	Log-loss
High dimensional features	Yes	No
Solution Sparse	Yes	Almost no
Output	Margin	Real Probabilities!

SVM for regression

$$L = \frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}||w||^2 \qquad (23)$$

ϵ-insensitive error function (Vapnik, 1995)

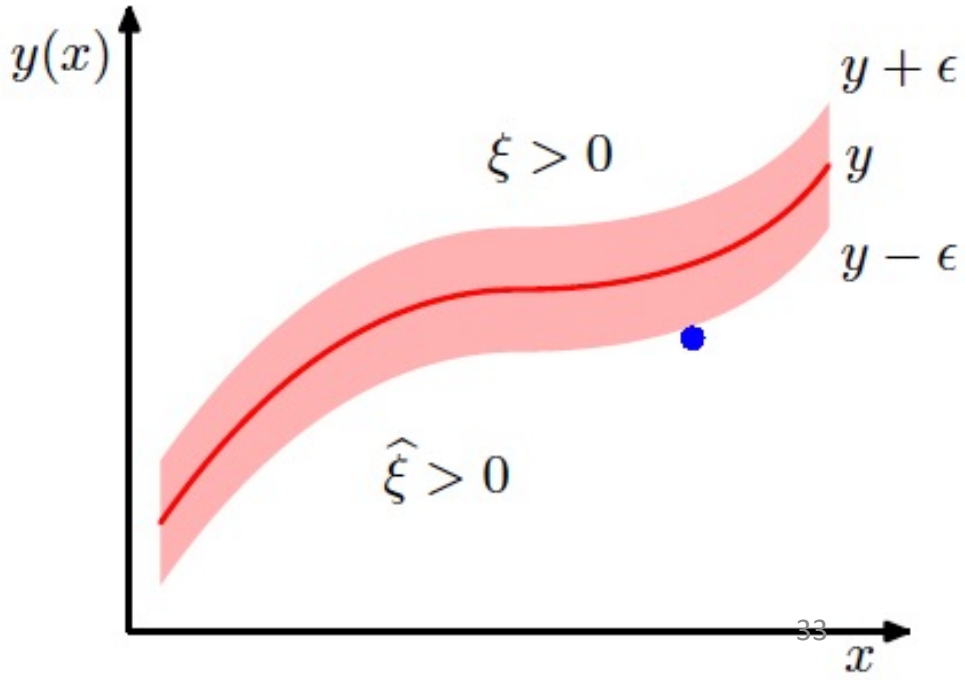
$$E_\epsilon(y(x) - t) = \begin{cases} 0 & if\ |y(x) - t| < \epsilon \\ |y(x) - t| & \text{Otherwise} \end{cases}$$



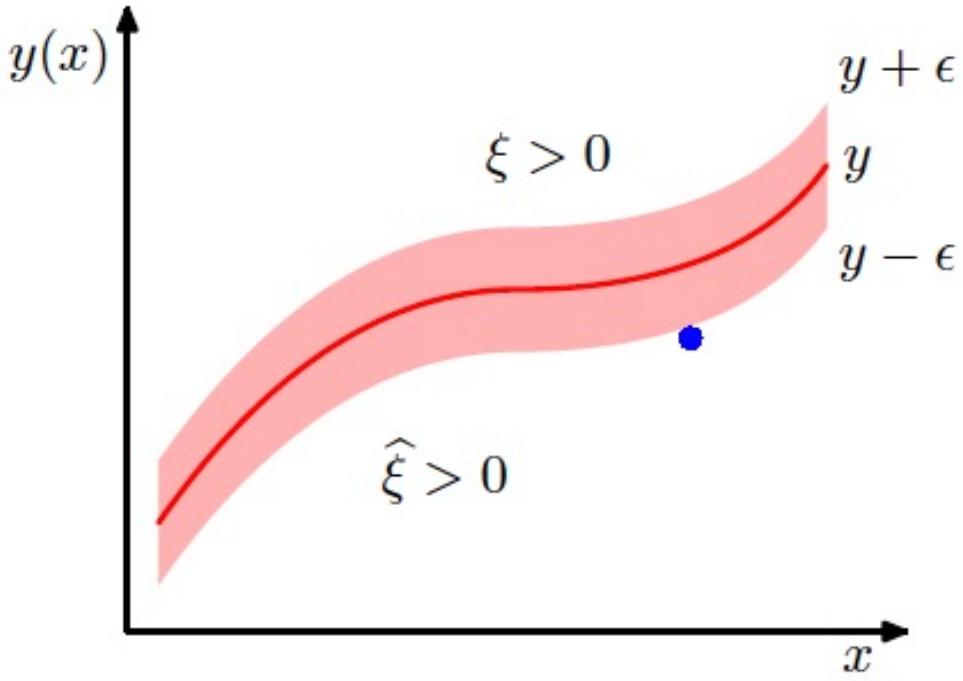
$$C\sum_{n=1}^{N} E_\epsilon(y_n(x_n) - t_n) + \frac{\lambda}{2}||w||^2 \qquad (24)$$

$\xi_n > 0$ corresponds to a point $t_n > y(x_n) + \epsilon$ and $\hat{\xi}_n \geq 0$ for $t_n < y(x_n) - \epsilon$

$$C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}||w||^2 \qquad (25)$$



SVM for regression



$\xi_n > 0$ corresponds to a point $t_n > y(x_n) + \epsilon$ and $\hat{\xi}_n \geq 0$ for $t_n < y(x_n) - \epsilon$

$$L = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}||w||^2 \qquad (25)$$

Eqn 25 must be minimized subject to $\xi_n, \hat{\xi}_n \geq 0$.
How about t_n?

$$t_n \leq y_n + \epsilon + \xi_n$$
$$t_n \geq y_n - \epsilon - \hat{\xi}_n$$

How can we optimize L?

SVM for regression - Optimize by slack variable

$$L = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}||w||^2 - \sum_{n=1}^{N}(\mu_n\xi_n + \hat{\mu}_n\hat{\xi}_n) - \sum_{n=1}^{N}a_n(\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N}\hat{a}_n(\epsilon + \hat{\xi}_n - y_n + t_n) \quad (26)$$

where $a_n, \hat{a}_n, \mu_n, \hat{\mu}_n \geq 0$.

The derivative w.r.t. w, b, ξ_n, and $\hat{\xi}_n$:

1) $\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N}(a_n - \hat{a}_n)\phi(x_n)$

2) $\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0$

3) $\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n$

4) $\frac{\partial L}{\partial \hat{\xi}_n} = 0 \Rightarrow \hat{a}_n = C - \hat{\mu}_n$

SVM for regression - Optimize by slack variable

$$\tilde{L}(\boldsymbol{a}, \hat{\boldsymbol{a}}) = -\frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(\boldsymbol{x_n}, \boldsymbol{x_m}) - \epsilon\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a}_n)\,t \quad (27)$$

$$y(x) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(x, x_n) + b$$

$$a_n(\epsilon + \xi_n + y_n - t_n) = 0$$
$$\hat{a}_n(\epsilon + \hat{\xi}_n - y_n + t_n) = 0$$
$$(C - a_n)\xi_n = 0$$
$$(C - \hat{a}_n)\hat{\xi}_n = 0$$

$$b = t_n - \epsilon - \boldsymbol{w^T\phi}(\boldsymbol{x_n}) = t_n - \epsilon - \sum_{m=1}^{N}(a_m - \hat{a}_m)k(\boldsymbol{x_n}, \boldsymbol{x_m}) \quad (28)$$

$$L(a, \hat{a}) = -\frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(x_n, x_m) + \sum_{n=1}^{N}(a_n - \hat{a}_n)\,t_n \quad (29)$$

Kernel SVM

- Outputs represents decisions (can be an advantage but also can be a disadvantage)
- Originally formulated for two classes
- Predictions are linear combination of kernel functions **but centered on training data points and that are required to be positive-definite**.

- How can we overcome the limitation of centered kernel?

- The relevance vector machine or RVM (Tipping 2001)
  - Faster performance on test data while the generalization error is maintained.

RVM - Regression

The modified prior:

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x), \beta^{-1}) \tag{6-50}$$

where $\beta = \sigma^{-2}$ is the noise precision.

The mean of the linear model

$$y(x) = \sum_{i=1}^{M} w_i \phi_i(x) = w^T \phi(x)$$

In RVM, we use kernels with one kernel associated with each of the data points in training set.

$$y(x) = \sum_{i=1}^{N} w_n k(x, x_n) + b \tag{6-52}$$

where b is the bias parameter and $M = N + 1$.

Here, **there is no restriction of positive-definite and the number nor the location to the training data sets.**

RVM - Regression

Suppose we have N observations of data matrix X. The likelihood function is

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta) = \prod_{n=1}^{N} p(t_n|\boldsymbol{x}_n,\boldsymbol{w},\beta)$$

and a weight prior distribution is

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{n=1}^{M} \mathcal{N}(w_i|0,\alpha_i^{-1})$$

α_i is a separate hyperparameter for each of weight parameter.

If the posterior distribution for the weights is Gaussian, then it is

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m},\boldsymbol{\Sigma})$$

Mean: $\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{t} = \beta\boldsymbol{\Sigma}\mathbf{K}^{\mathbf{T}}\mathbf{t}$

Covariance: $\Sigma = \left(\boldsymbol{A} + \beta\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi}\right)^{-1}$

$= \left(\boldsymbol{A} + \beta\boldsymbol{K}^{\boldsymbol{T}}\boldsymbol{K}\right)^{-1}$

The maximization of likelihood function by integration of the weight parameters

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta) = \int p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w} \qquad (6\text{-}53)$$

RVM - Regression

If the posterior distribution for the weights is Gaussian, then it is

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m},\boldsymbol{\Sigma})$$

The maximization of likelihood function by integration of the weight parameters

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta) = \int p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w} \quad (6\text{-}54)$$

The log likelihood also can be formatted as

$$\ln p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta) = \ln \mathcal{N}(\boldsymbol{t}|\boldsymbol{0},\boldsymbol{C}) - \frac{1}{2}\{N\ln(2\pi) + \ln|\boldsymbol{C}| + \boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}\} \quad (6\text{-}55)$$

where $\boldsymbol{C} = \beta^{-1}\boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T$.

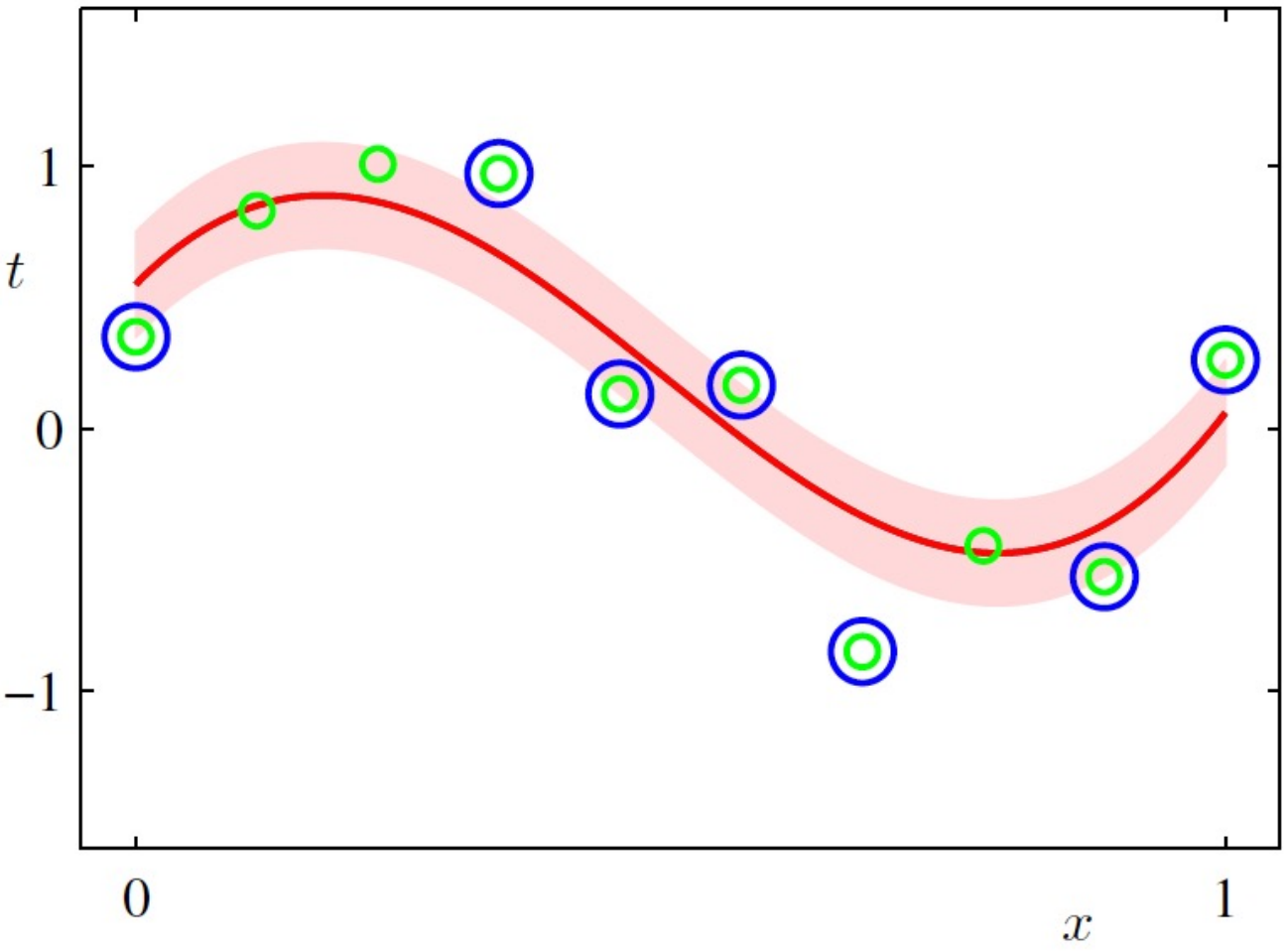
Set the required derivatives of the likelihood to zero and find the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

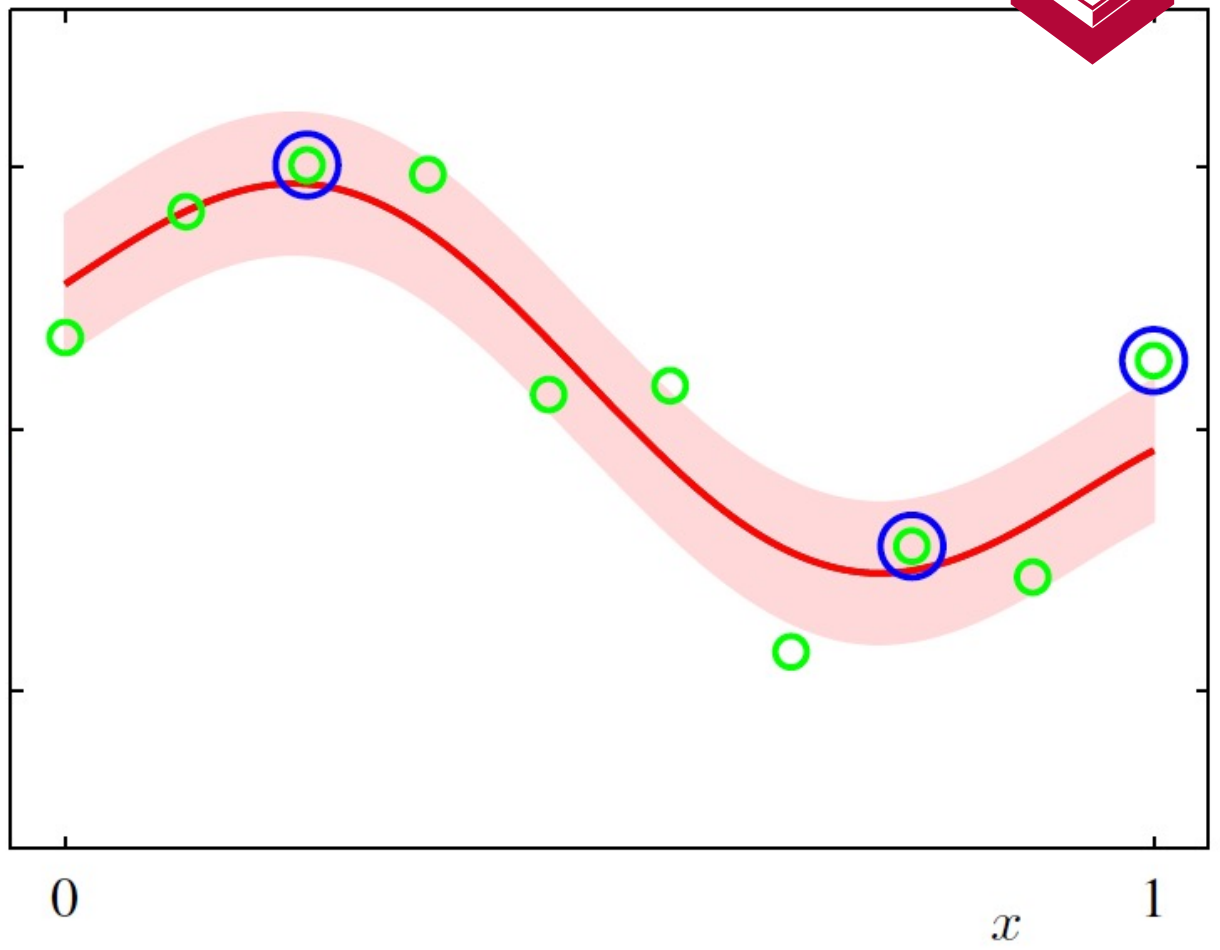
$$\alpha_i^{new} = \frac{\gamma_i}{m_i^2}$$

$$(\beta^{new})^{-1} = \frac{\left||\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{m}|\right|^2}{N - \sum_i \gamma_i}$$

where m_i is the i^{th} component of the poster mean $\boldsymbol{m}$ and the quantity $\gamma_i = 1 - \alpha_i\Sigma_{ii}$ is the measurement of how well w_i is measured.

RVM - Regression





RVM - Regression

Linear Regression – the predictive variance becomes small in regions of input space.

SVM Regression - ϕ is centered on data points and the model will become increasingly certain of its predictions when extrapolating outside the domain of the data.

Gaussian Process – the computation cost is high.

RVM – A significant improvement in the speed of processing on test data. This greater sparsity is achieved with little or no reduction in generalization error compared with SVM.

But… Training involves optimizing a nonconvex function and takes longer.

RVM - Classification

Let's begin from the two-class problem with a binary target $t \in \{0,1\}$.

If the model takes the linear combination format of transformation function $\phi(\cdot)$ by a logistic sigmoid function

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(w^T \phi(\boldsymbol{x})\right).$$

The posterior distribution over w (assume it is a Gaussian prior) is

$$p(\boldsymbol{w}|\boldsymbol{t}) \propto p(\boldsymbol{w}) p(\boldsymbol{t}|\boldsymbol{w}).$$

We can solve for $\boldsymbol{\alpha}$ by following the way we saw in RVM for regression using Hassian matrix.

RVM - Classification

$$\ln p(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha}) = \ln\{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})\} - \ln p(\boldsymbol{t}|\boldsymbol{\alpha})$$

$$= \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} - \frac{1}{2}\boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w} + const.$$

From logistic regression lecture,
Refer to equation 22.

Regularization comes from the prior in the previous slide,
$p(\boldsymbol{w}|\boldsymbol{t}) \propto p(\boldsymbol{w})p(\boldsymbol{t}|\boldsymbol{w})$

Note: $\ln p(w|t) = -\frac{1}{2}(\boldsymbol{w}-\boldsymbol{m}_0)^T\boldsymbol{S}_0^{-1}(\boldsymbol{w}-\boldsymbol{m}_0) + \sum_{n=1}^{N}\{t-y_n)\ln(1-y_n)\} + Const.$ when the Gaussian prior of **w** is $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m_0},\boldsymbol{S}_0)$

From here, we can take the gradient of $\ln p(w|t,\alpha)$ be zero to obtain the mean and α_i^{new} is

$$\alpha_i^{new} = \frac{\gamma_i}{m_i^2}$$

Empirical Risk – Loss Functions

Empirical Risk Minimization

$$\min_{w} \frac{1}{n} \sum_{i=1}^{N} \underbrace{l(h_w(x_i), y_i)}_{\text{Loss}} + \underbrace{\lambda r(w)}_{\text{Regularization}}$$

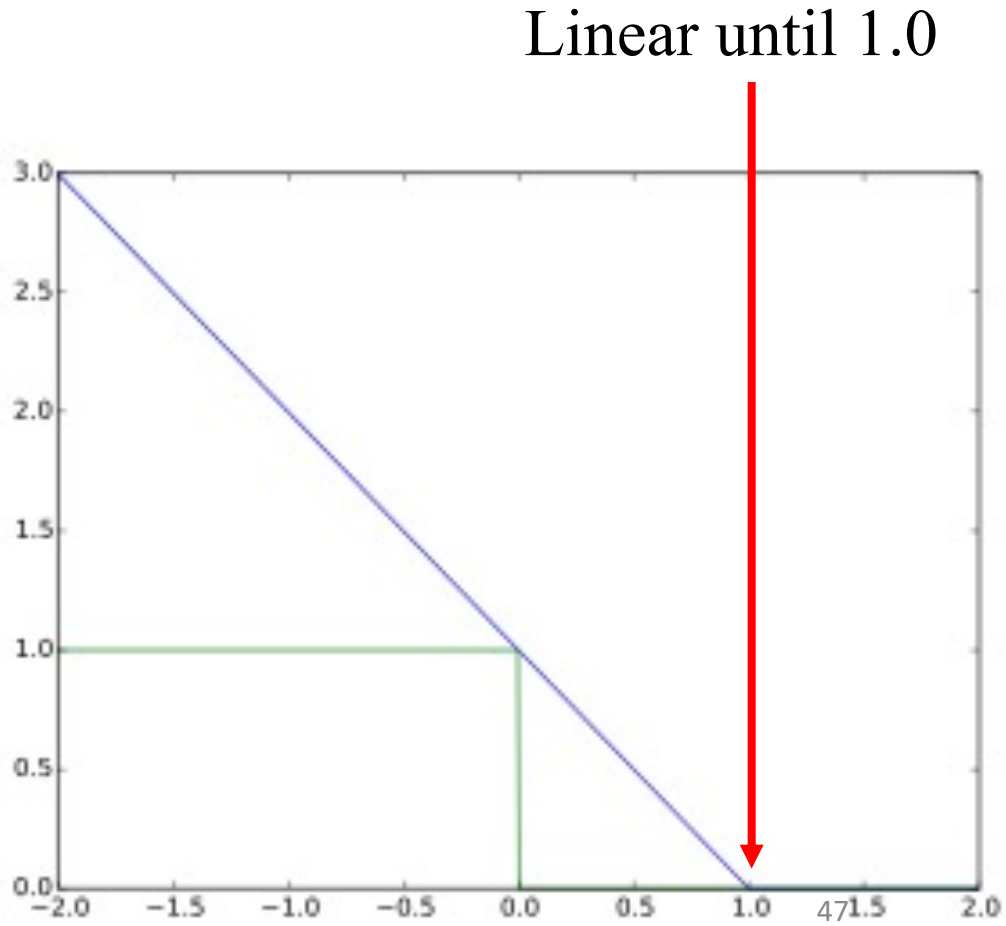
- The loss function – a continuous function penalizing training error
- The regularization – a continuous function penalizing classifier complexity

Classification: Hinge-Loss

$$\max[1 - h_w(x_i)y_i, 0]^p$$

- Usage: Supportive Vector Machine
- When $p = 1$, Standard SVM: the loss function denotes the size of the margin between linear separator and its closest points in either class.

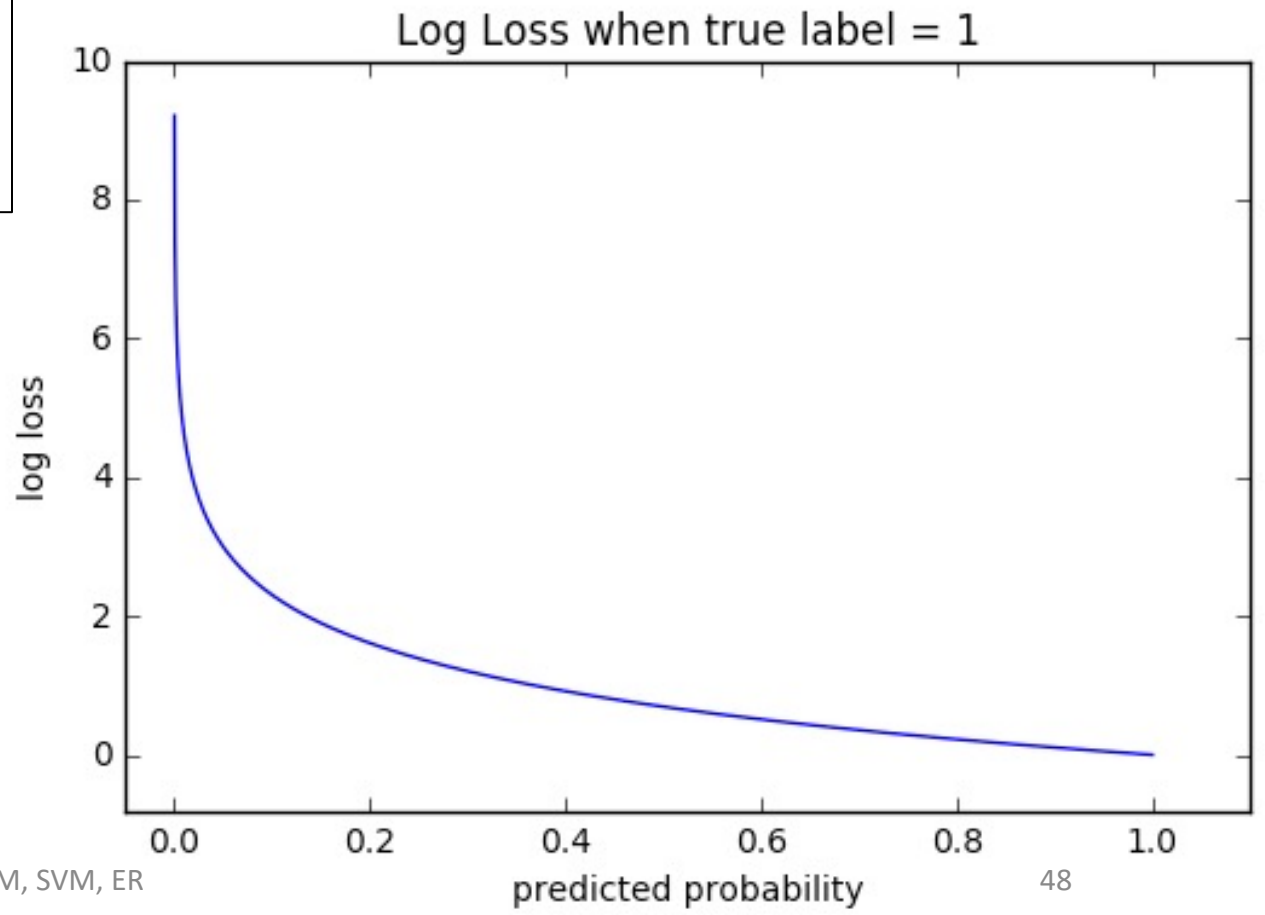
- When $p = 2$, Squared Hingeless SVM: Only differentiable everywhere with $p = 2$.



Classification: Log-Loss

$$\log\left(1 + e^{-h_w(x_i)y_i}\right)$$

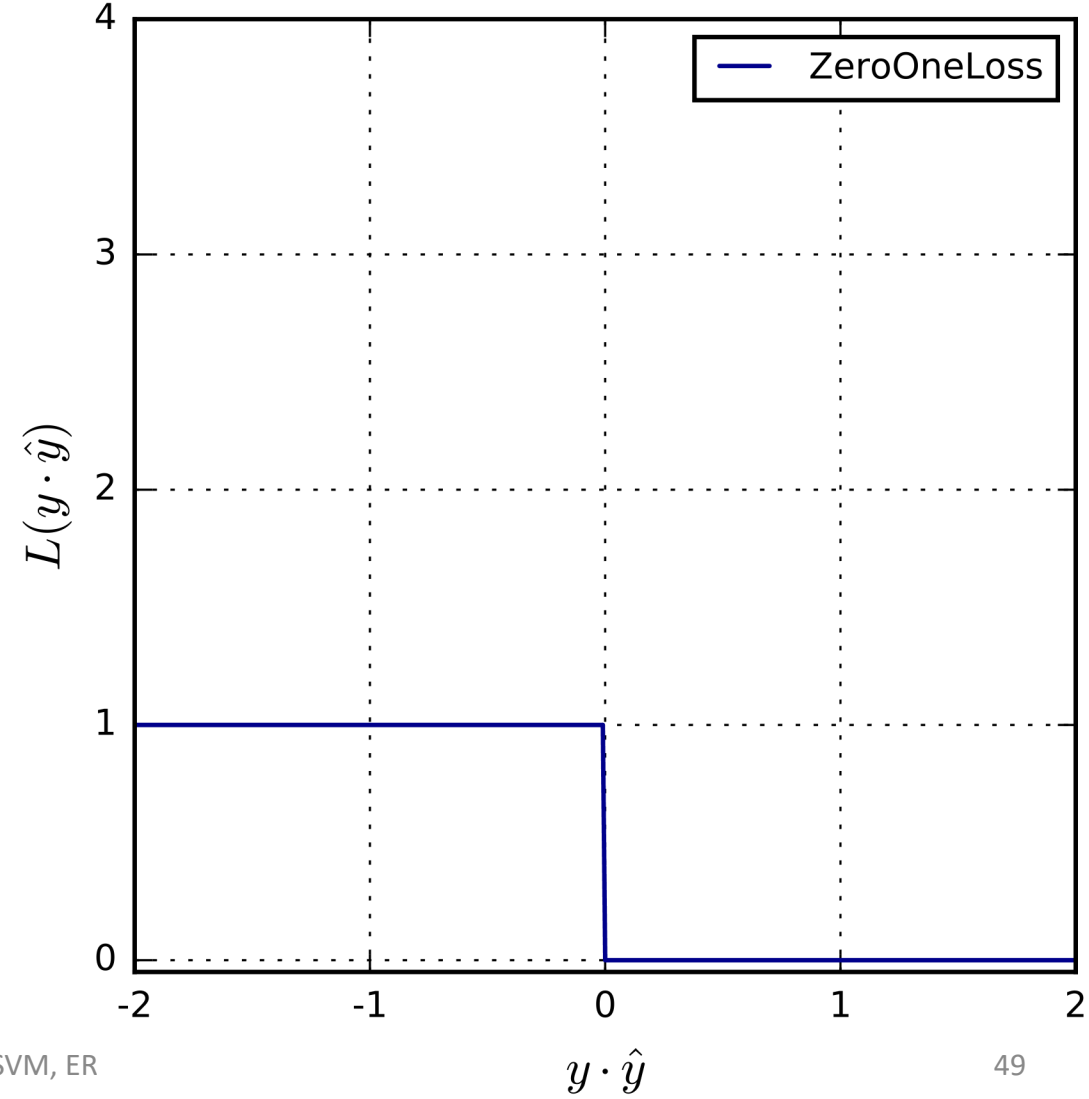
- Usage: Logistic Regression
- Its outputs are well-calibrated probabilities.
- Value range from 0 to 1.
- Goal: to minimize its value.



Classification: Zero-One Loss

$$\delta\big(sign\big(h_w(x_i)\big) \neq y_i\big)$$

- Usage: Actual Classification Loss
- Characteristic: Not convex nor continuous.



Loss Function: Classifications

Hinge-Loss:

$$\max[1 - h_w(x_i)y_i, 0]^p$$

- Usage: SVM
- When $p = 1$, Standard SVM: the loss function denotes the size of the margin between linear separator and its closest points in either class.
- When $p = 2$, Squared Hingeless SVM: Only differentiable everywhere with $p = 2$.

Log-Loss:

$$\log\left(1 + e^{-h_w(x_i)y_i}\right)$$

- Usage: Logistic Regression
- Its outputs are well-calibrated probabilities.
- Value range from 0 to 1.
- Goal: to minimize its value.

Zero-One Loss:

$$\delta\left(sign\left(h_w(x_i)\right) \neq y_i\right)$$

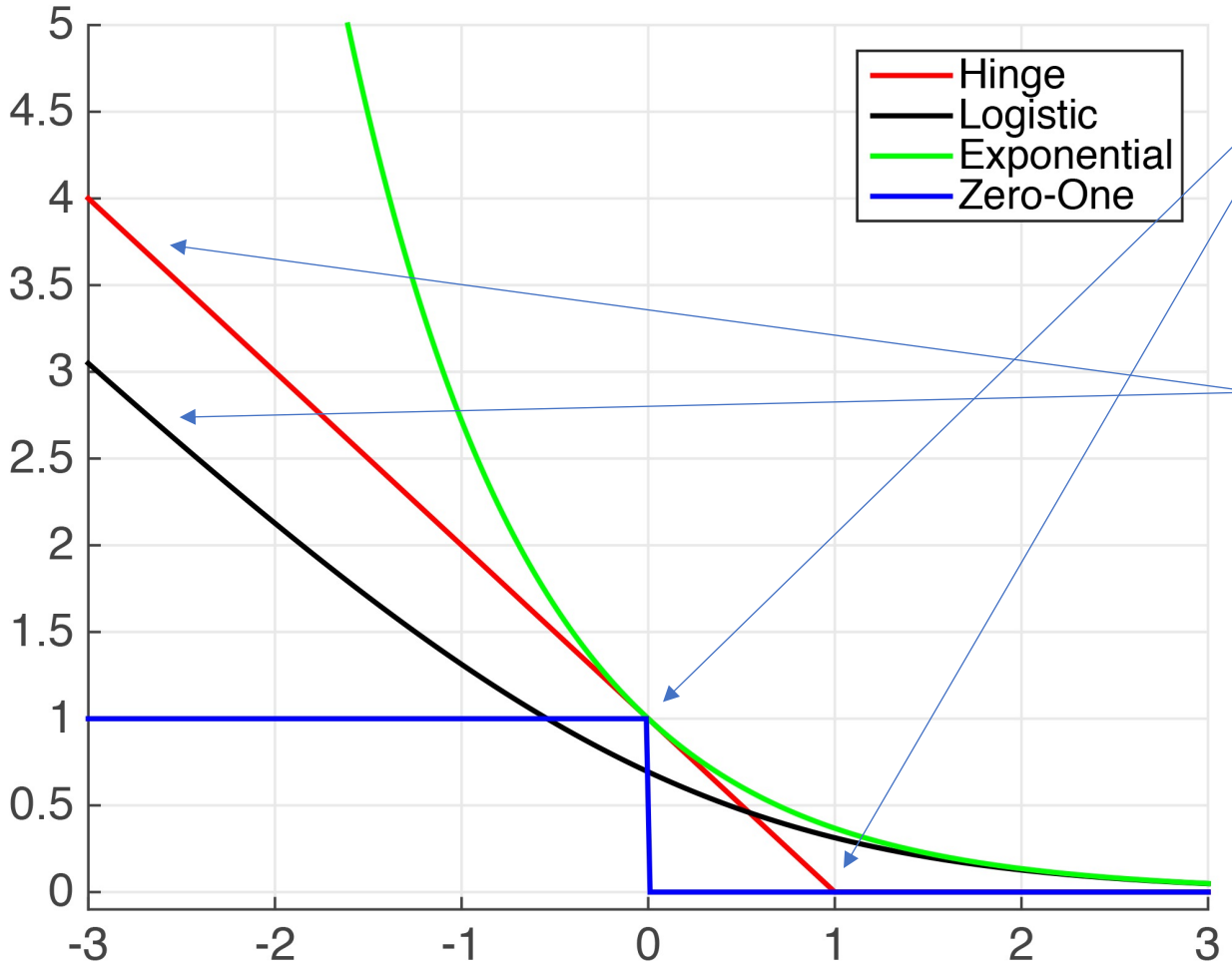
- Usage: Actual Classification Loss
- Characteristic: Not convex nor continuous.

Exponential Loss:

$$\exp(-h_w(x_i)y_i)$$

- Usage: AdaBoost
- Characteristic: Very aggressive
- Misprediction increases exponentially with the value of $-h_w(x_i)y_i$.
- May cause problems with noisy data

Classification Loss Functions Summary



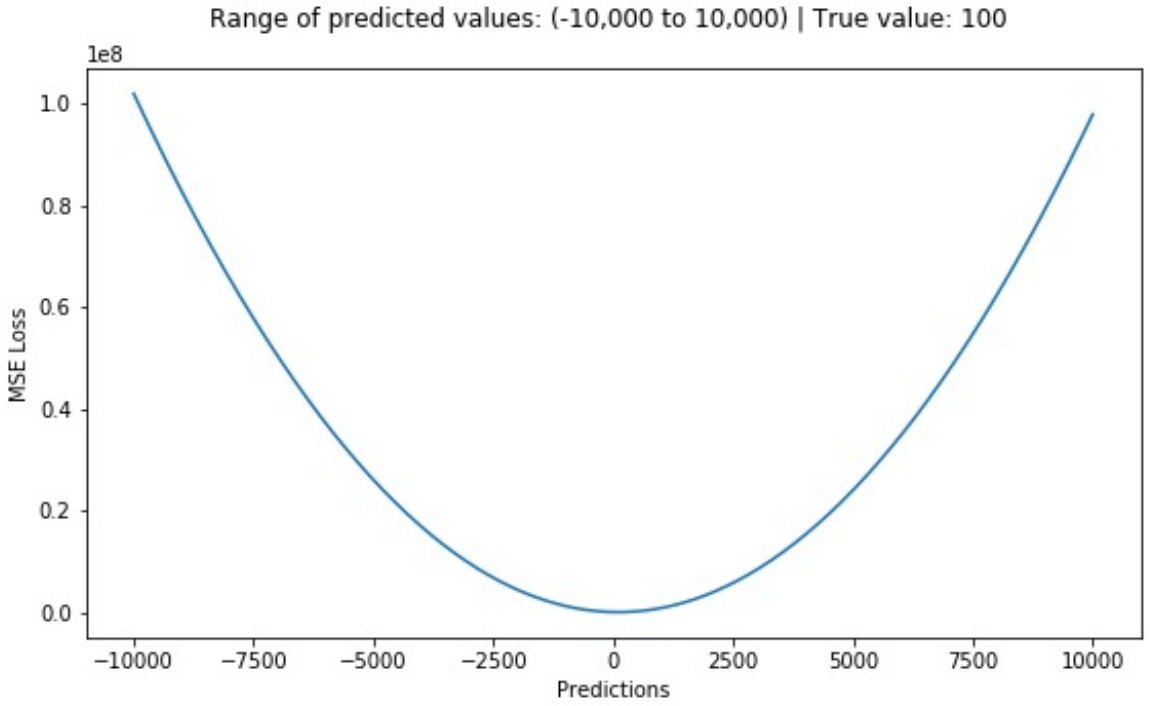
Which functions are strict upper bounds on the 0/1-loss?

What can you say about the hinge-loss and the log-loss as $z \to -\infty$?

Regression: Mean Square Loss

$$(h(x_i) - y_i)^2$$

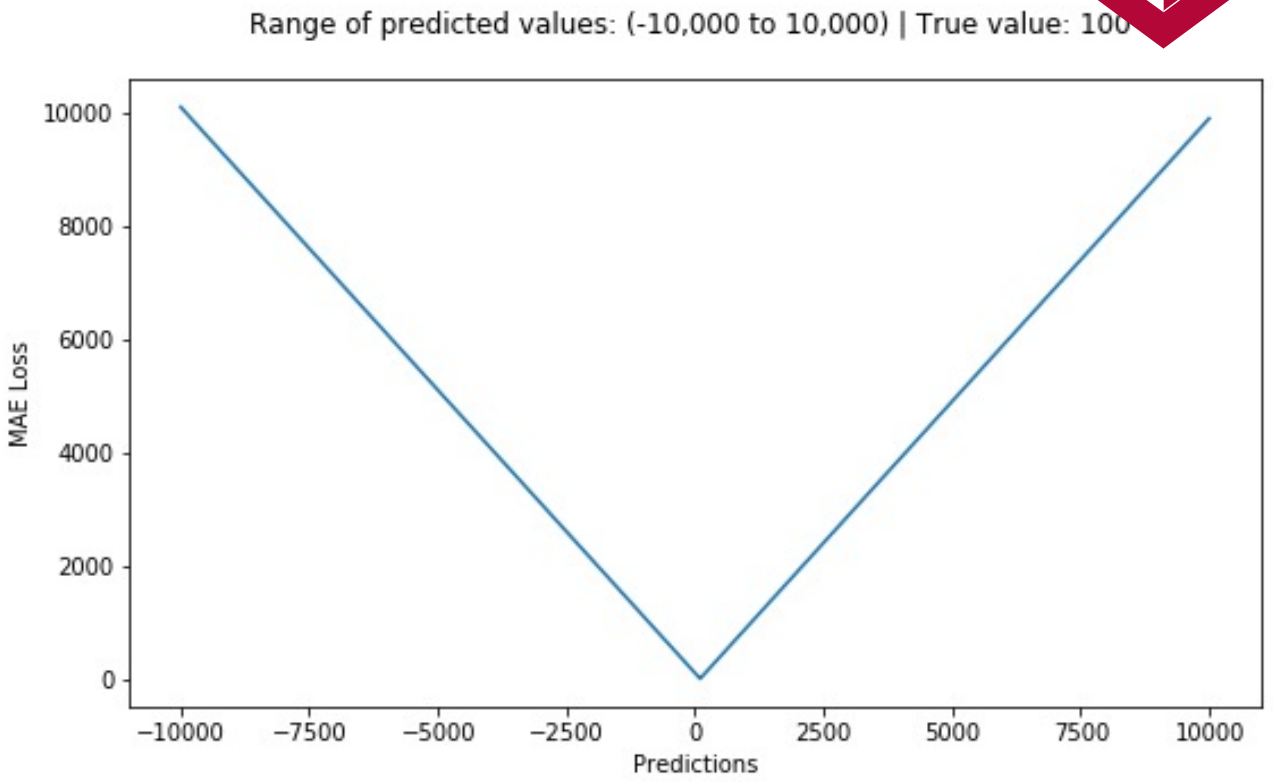
- Most popular regression loss function.
- Estimates Mean.
- It is differentiable everywhere.
- Sensitive to outliers.



Regression: Absolute Loss

$$|h(x_i) - y_i|$$

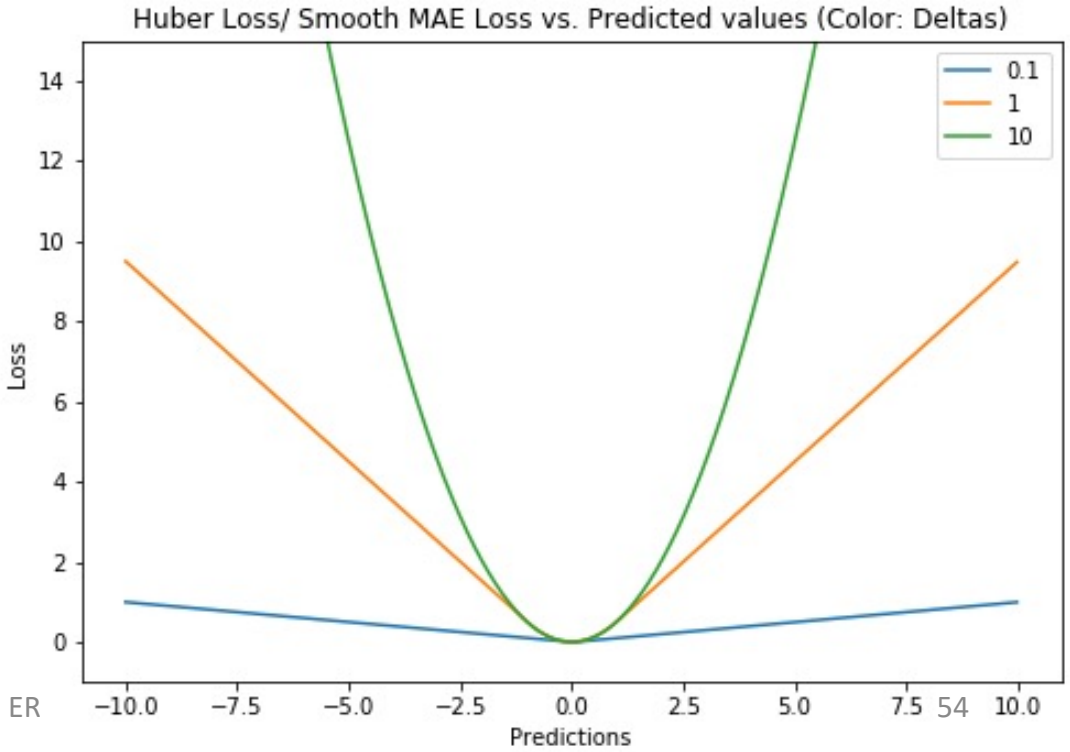
- Another Most popular regression loss function.
- Estimates Median.
- Less sensitive than squared loss function.
- Not differentiable at 0.



Regression: Huber Loss

$$\begin{cases} \frac{1}{2}(h(x_i) - y_i)^2, & if\ |h(x_i) - y_i| < \delta \\ \delta\left(|h(x_i) - y_i| - \frac{\delta}{2}\right), & Otherwise \end{cases}$$

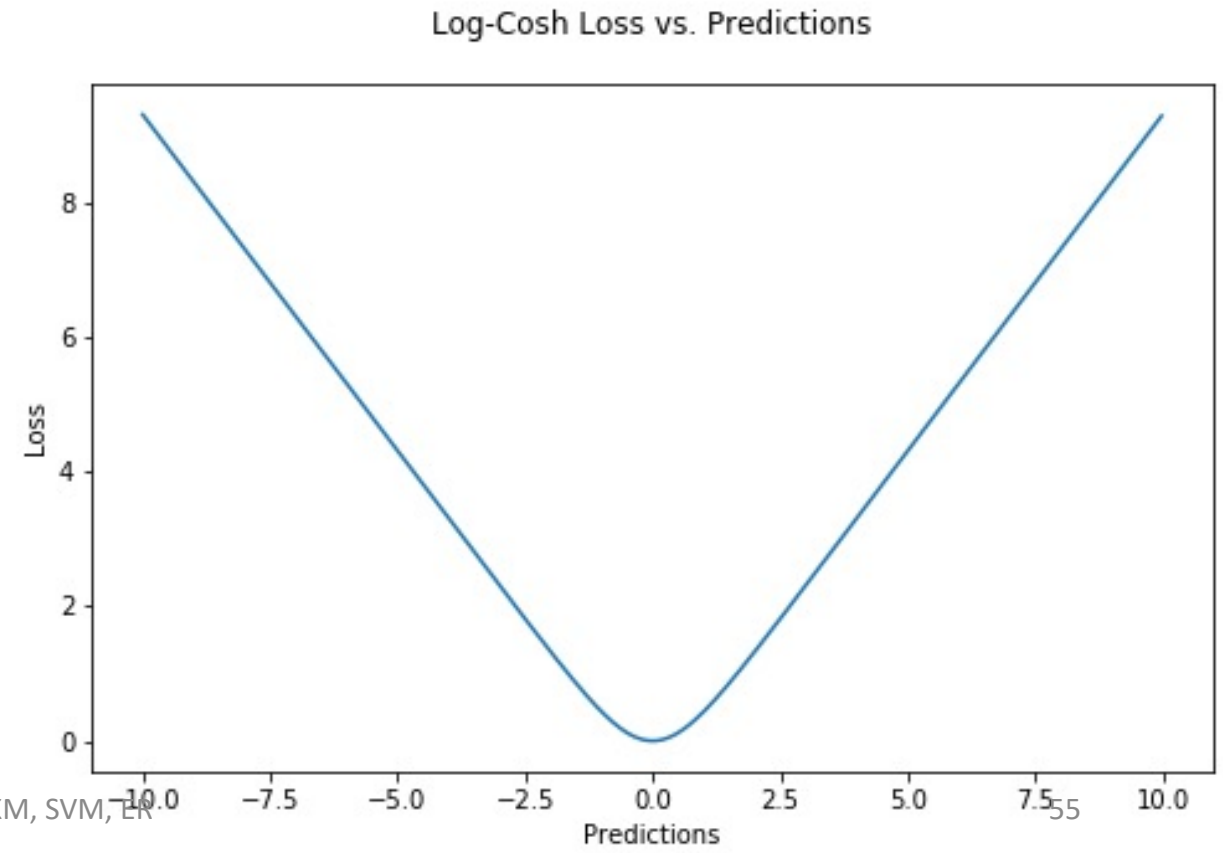
- Also known as Smooth Absolute Loss
- "Best of Both Worlds"
- Once-differentiable.
- When the loss is large, the squared loss.
- When the loss is small, the absolute loss.



Regression: Squared Loss

$$\log(\cosh(h(x_i) - y_i)), \cosh\left(\frac{e^x + e^{-x}}{2}\right)$$

- Similar to Huber Loss.
- Differentiable everywhere.



Loss Functions: Regressions

Squared Loss:

$$(h(x_i) - y_i)^2$$

- Most popular regression loss function.
- Estimates Mean.
- It is differentiable everywhere.
- Sensitive to outliers.

Absolute Loss:

$$|h(x_i) - y_i|$$

- Another Most popular regression loss function.
- Estimates Median.
- Less sensitive than squared loss function.
- Not differentiable at 0.

Huber Loss:

$$\begin{cases} \frac{1}{2}(h(x_i) - y_i)^2, & if\ |h(x_i) - y_i| < \delta \\ \delta(|h(x_i) - y_i| - \delta/2), & Otherwise \end{cases}$$

- Also known as Smooth Absolute Loss
- "Best of Both Worlds"
- Once-differentiable.
- When the loss is large, use absolute loss.
- When the loss is small, use squared loss.

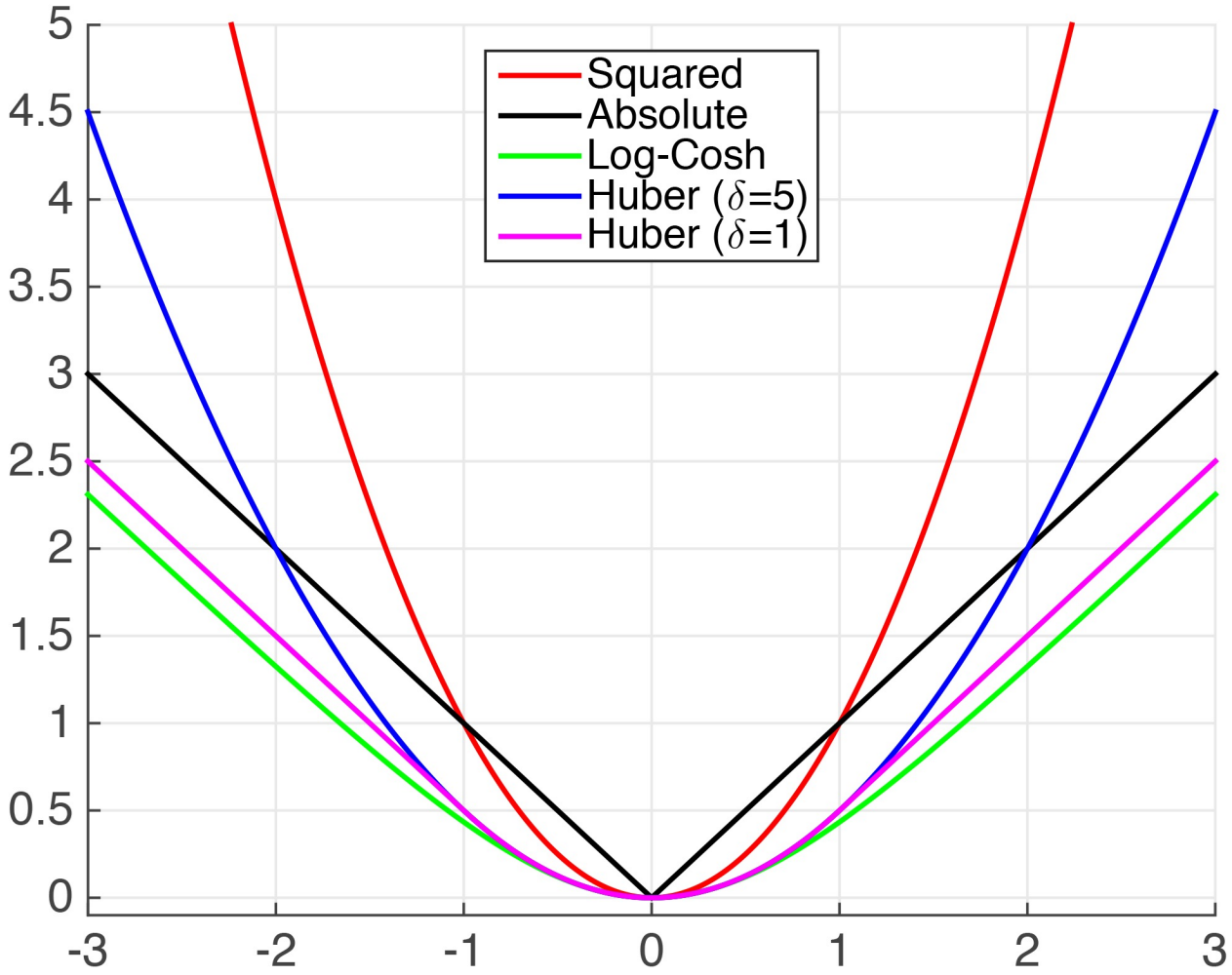
Log-Cosh Loss:

$$\log(\cosh(h(x_i) - y_i))$$

$$\cosh\left(\frac{e^x + e^{-x}}{2}\right)$$

- Similar to Huber Loss.
- Differentiable everywhere.

Regression: Loss Functions Summary



See how sensitive functions are!

Regularization

We can write Equation (1) into different formation of optimization problem.

$$\min_{w,b} \sum_{i=1}^{N} l(h_w(x_i), y_i) + \lambda r(w) \tag{2}$$

$$\Leftrightarrow \min_{w,b} \sum_{i=1}^{N} l(h_w(x_i), y_i) \;\; s.t. r(w) \leq B$$

Equation (2) indicates that for each $\lambda \geq 0$, the regularizer r(w) is less than or equal to a constant number B

Regularization – Lasso and Ridge

l_2 Regularization:

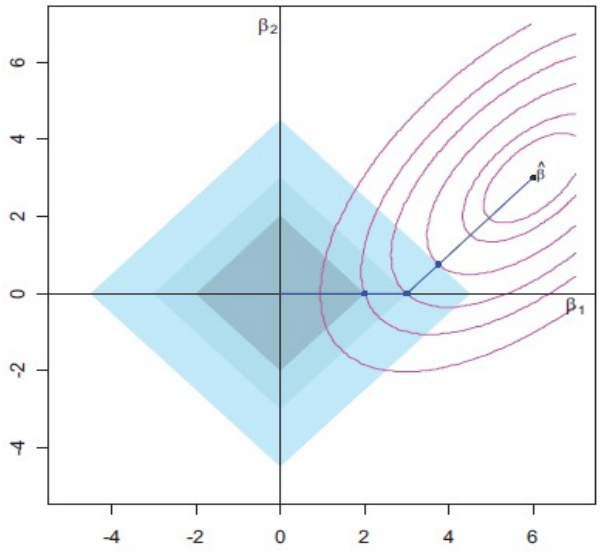
$$r(w) = w^T w = ||w||^2$$

- Strictly convex and differentiable
- Uses weights on all features – dense solutions

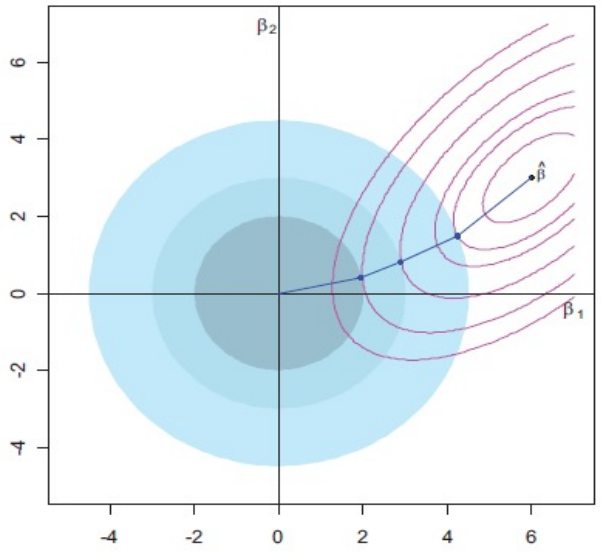
l_1 Regularization:

$$r(w) = ||w||$$

- Convex but not strictly
- Not differentiable at 0
- Sparse Solutions



l_1



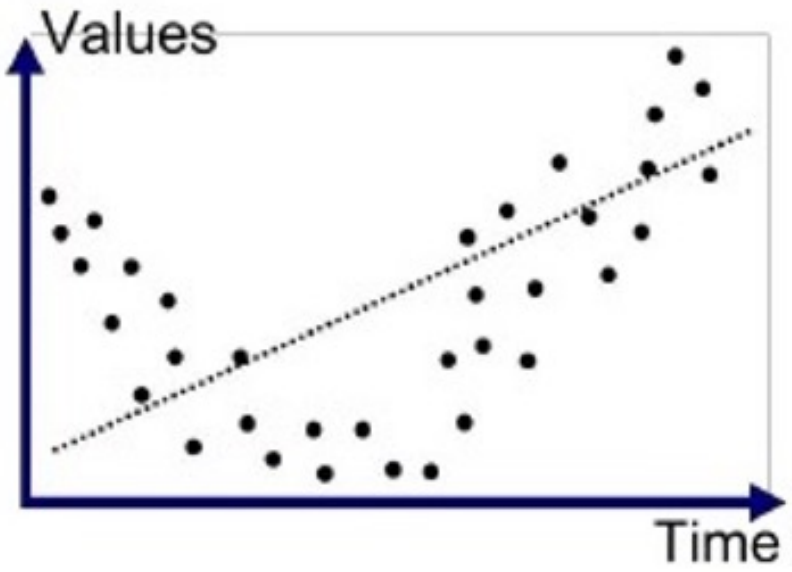
l_2

Elastic Net Regularization

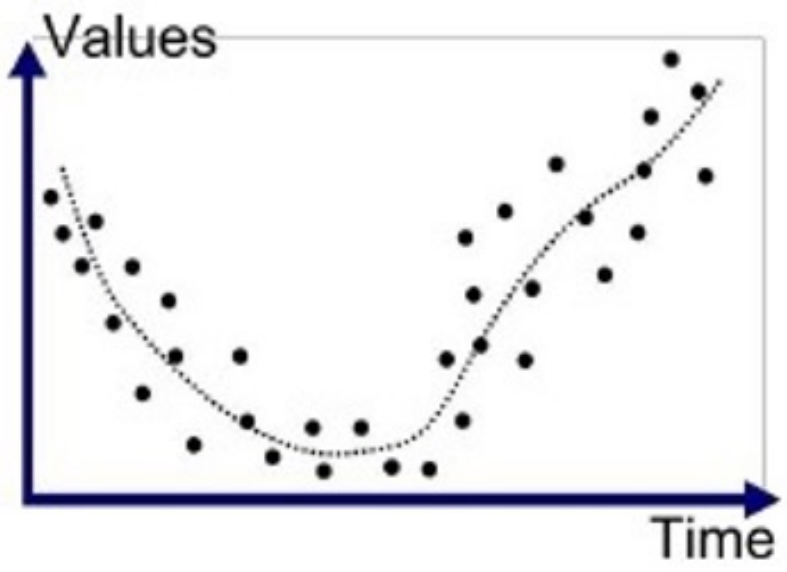
$$\lambda||w|| + (1-\lambda)||w||^2$$

Loss and Regularizer	Comments
Least Squares $\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$	• Square Loss • No Regularization • Closed form solution: $w = (XX^T)^{-1}Xy^T$ • $X = [x_1, \dots, x_n], y = [y_1, \dots, y_n]$
Ridge Regression $\min_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|\|w\|\|^2$	• Squared Loss • l_2 regularization • $w = (XX^T + \lambda I)^{-1}Xy^T$
Lasso Regression $\min_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|\|w\|\|$	• Sparsity inducing – good for feature selection • Convex but not strict (no unique solution) • Not differentiable at 0 • Solve with sub-gradient descent
Elastic Net $\min_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|\|w\|\| + (1-\lambda)\|\|w\|\|^2$	• Unique solution and sparsity inducing • Dual of squared-loss SVM • Non-differentiable
Logistic Regression $\min_{w,b} \frac{1}{n}\sum_{i=1}^{n}\log\left(1 - e^{-y_i(w^T x_i + b)}\right)$	• Solve with gradient descent • Probability estimation
Linear SVM $\min_w C\sum_{i=1}^{N}\max[1 - y_i(w^T x_i + b), 0] + \lambda\|\|w\|\|^2$	• Typically l_2 regularized sometimes l_1 • Quadratic program • Kernalize – Solve very efficiently

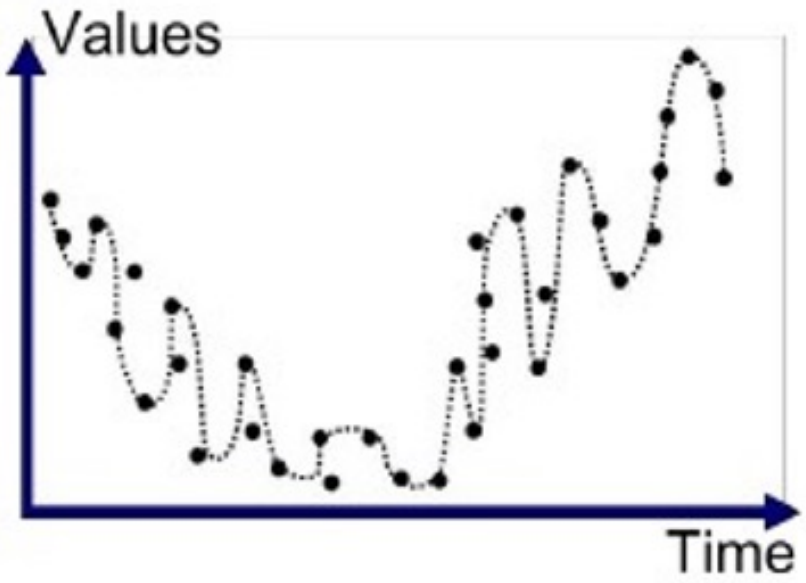
Overfitting vs. Underfitting



Underfitted



Good Fit/Robust



Overfitted

Overfitting vs. Underfitting

$$\min_w \frac{1}{n}\sum_{i=1}^{n} l(h_w(x_i), y_i) + \lambda r(w)$$

Loss Regularizer

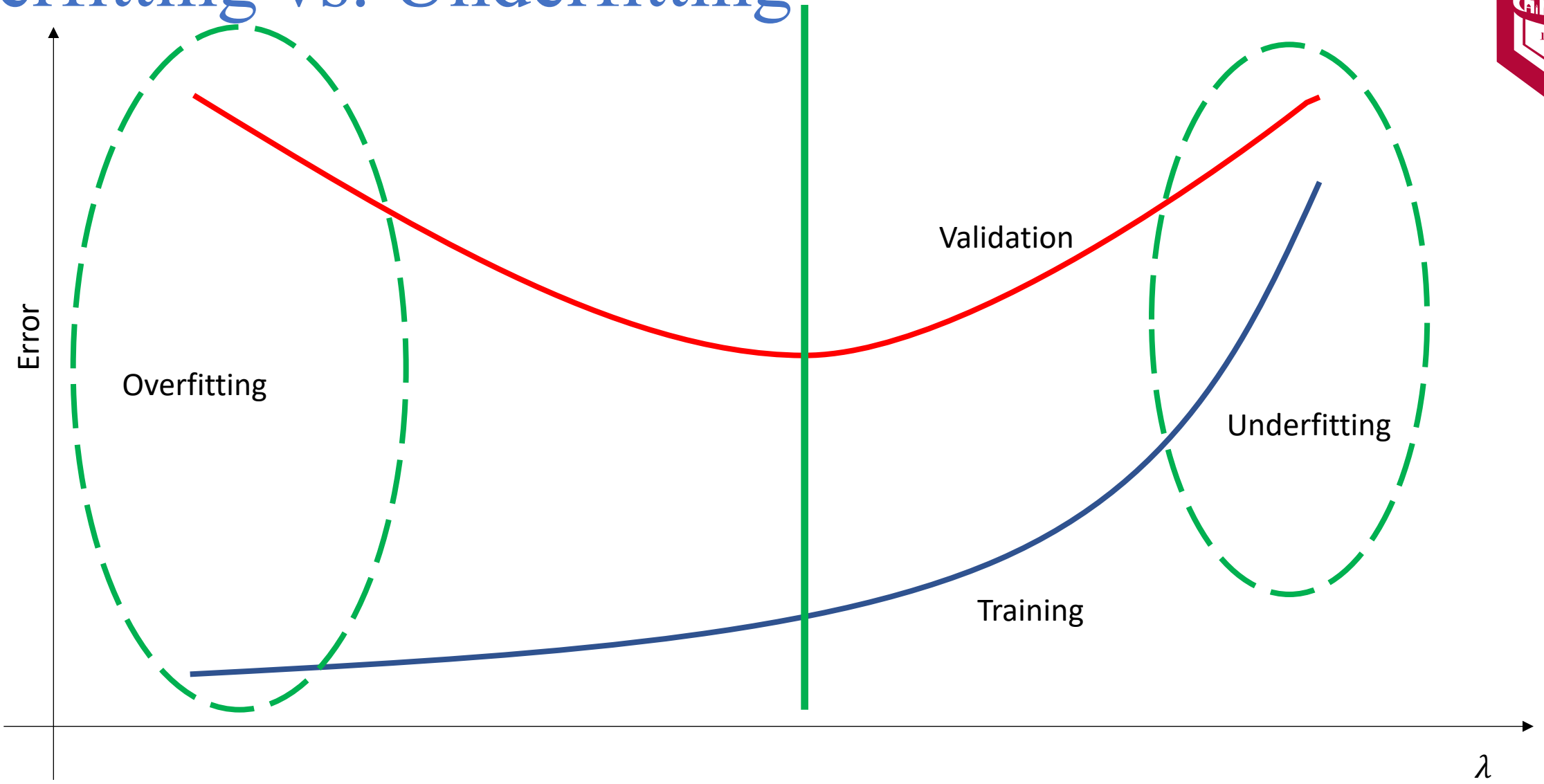
A question rise is on λ – how should we handle it?

Leads to two possible scenarios: **Underfitting** or **Overfitting**

Underfitting – The model does not learn enough through the training process. Both errors on training and test will be high. (The model is relatively simpler than what it supposed to be.)

Overfitting – The model learns too much from training data set. The test error may rise because the model will reflects on the exact patterns of training data set in the test set. (The model is relatively complex than what it should be.)

Overfitting vs. Underfitting



Overfitting vs. Underfitting

The big question is…. **HOW do we find the exact point to stop?**

Typical range: $10^{-5}, 10^{-4}, 10^{-3}, \cdots, 10^{1}, 10^{2}, 10^{3}, \cdots$

K-fold Cross Validation

- Divide the training data k portion equally.
- Train on k-1 of them and leave k^{th} one for the validation.
- Do this k times and find the best λ that gives the smallest error on the validation set.
  - When the data is too small, you leave only one observation for the validation.

For a specific search on λ

- Do with $\lambda = \{\cdots, 10^{-3}, 10^{-2}, \cdots, 10^{1}, 10^{2}, \cdots\}$ to find the best magnitude.
- Then do for the specific value of λ value in the best chosen magnitude
- Example: Let's say you learned that $\lambda = 10^{-2}$ is the best magnitude on the 1^{st} try. Then do with $\lambda = \{0.01, 0.015, 0.02, \cdots\}$ on the 2^{nd} try.

Bias-Variance Tradeoff

$$E_{x,y,D}[(h_D(x)-y)^2] = E_{x,y,D}\left[\left[\left(h_D(x)-\bar{h}(x)\right)+\left(\bar{h}(x)-y\right)\right]^2\right]$$

$$= E_{x,D}\left[\left(h_D(x)-\bar{h}(x)\right)^2\right] + 2E_{x,y,D}\left[\left(h_D(x)-\bar{h}(x)\right)\left(\bar{h}(x)-y\right)\right] + E_{x,y}\left[\left(\bar{h}(x)-y\right)^2\right] \quad (1)$$

2nd Term

$$E_{x,y,D}\left[\left(h_D(x)-\bar{h}(x)\right)\left(\bar{h}(x)-y\right)\right] = E_{x,y}\left[E_D\left[h_D(x)-\bar{h}(x)\right]\left(\bar{h}(x)-y\right)\right]$$
$$= E_{x,y}\left[\left(E_D[h_D(x)]-\bar{h}(x)\right)\left(\bar{h}(x)-y\right)\right]$$
$$= E_{x,y}[0]$$

$$E_D[h_D(x)] = \bar{h}(s)$$

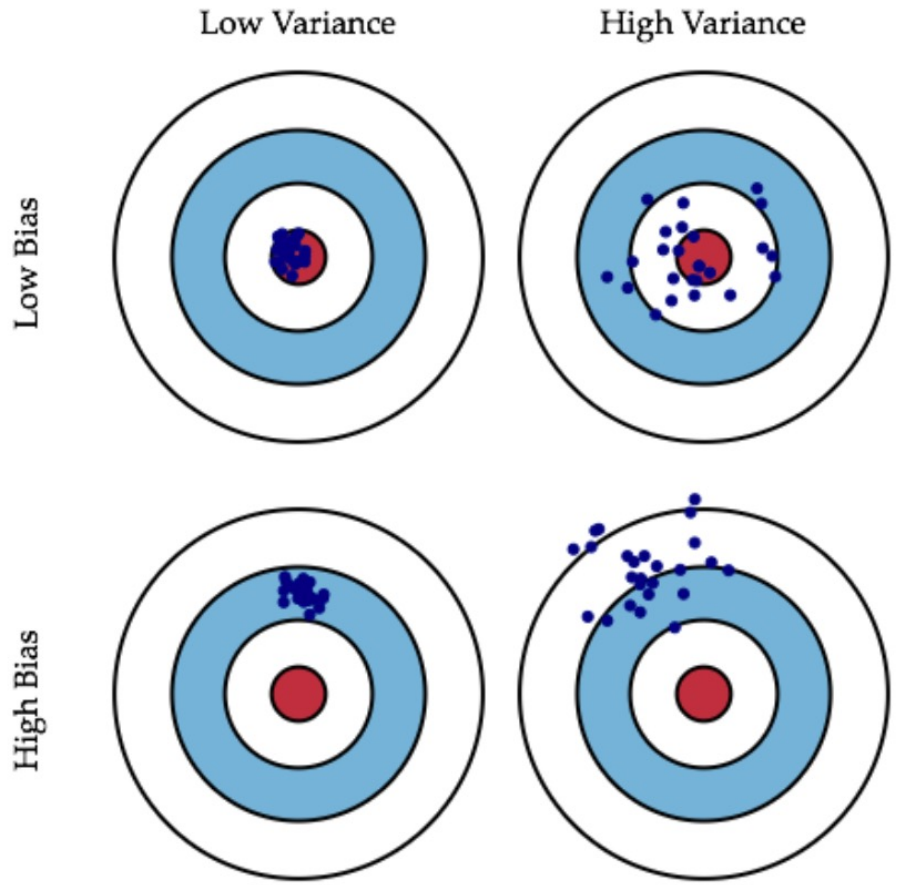
3rd Term

$$E\left[\left(\bar{h}(x)-y\right)^2\right] = E\left[\left(\left(\bar{h}(x)-\bar{y}(x)\right)+\left(\bar{y}(x)-y\right)\right)^2\right]$$
$$= E[(\bar{y}(x)-y)^2] + E\left[\left(\bar{h}(x)-\bar{y}(x)\right)^2\right] + 2E\left[\left(\bar{h}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]$$

$$E_{x,y,D}[(h_D(x)-y)^2] = E_{x,D}\left[\left(h_D(x)-\bar{h}(x)\right)^2\right] + E_{x,y}\left[\left(\bar{h}(x)-\bar{y}(x)\right)^2\right] + E_x[(\bar{y}(x)-y)^2] \quad (2)$$

$Var(x)$ =Variance Bias² ϵ =Noise

Bias-Variance Tradeoff



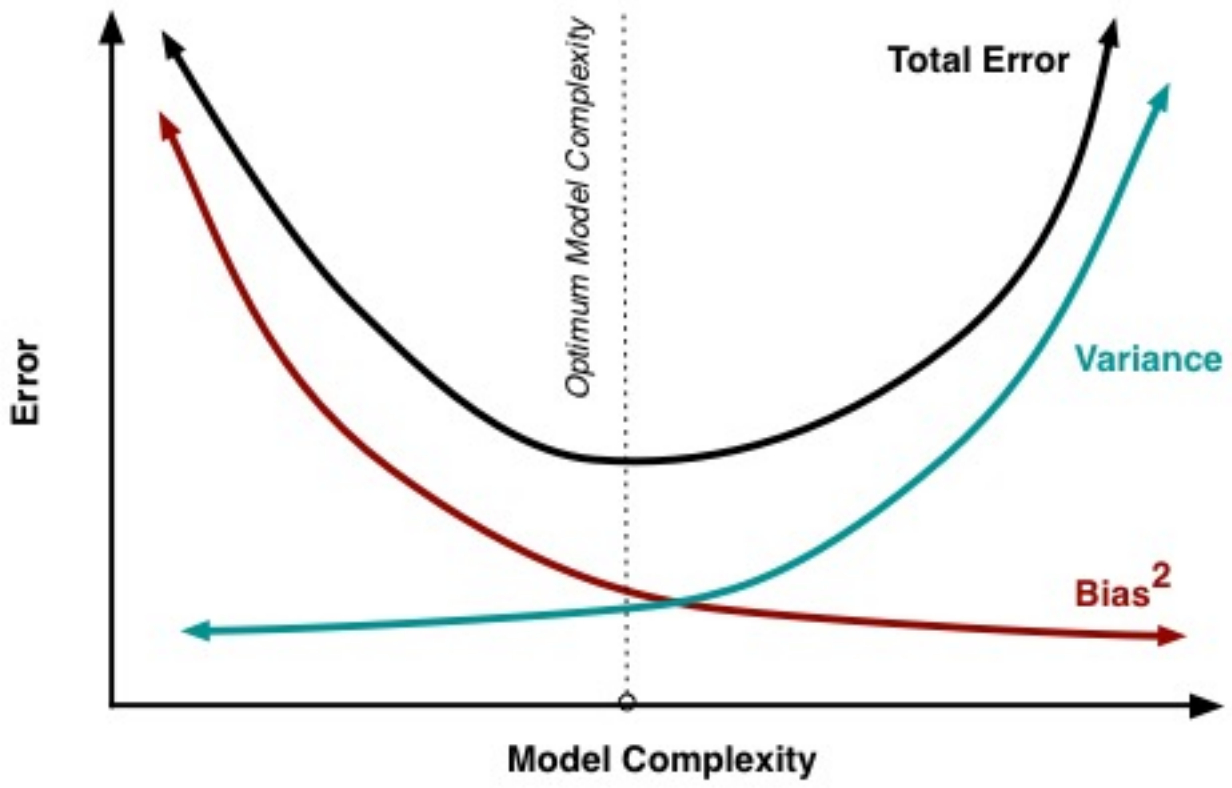


Figure: Graphical illustration of bias vs. variance