CS 559: Classification

Lecture 5

In Jang

ijang@stevens.edu

Lecture Outline

- ML Challenge
  - Model Selection
    - AIC vs BIC
    - Feature Selection
  - Optimization
- Classification I
  - KNN
  - Linear Discriminant Analysis (LDA)
  - Perceptron
  - Naïve Bayes
- Classification II
  - Probabilistic Generative Models
  - Probabilistic Discriminative Models (Logistic Regression)
  - Generalization

What is Model Selection?

Given a set of models $M = \{M_1, M_2, \ldots, M_R\}$, choose the model that is expected to do the best on the test data. M may consist of:

- Same learning model with different complexities or hyperparameters.
  - Nonlinear regression: polynomials with different degrees
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - SVM: Different choices of the misclassification penalty
  - Regularized models: Different choices of the regularization parameter
  - Kernel based methods: Different choices of kernels ...and
  - almost any learning problem
- Different learning models (e.g. SVM, KNN, DT, etc)

Note: usually considered in supervised learning but unsupervised learning faces this issue too.

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(L)$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(L)$$

- k: # of model parameters
- n: # of data examples
- L: maximum value of the model likelihood
- Applicable for probabilistic models
- AIC/BIC penalize model complexity

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(L)$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(L)$$

Factors	AIC	BIC
Penalty Weight	Weak: $2k$	Strong: $k \cdot \log(N)$
Dependency	Independent to sample size	Depends on sample size
Most Likely Error	Over-fitting	Under-fitting
Emphasis	Good future prediction	Explanatory model
Function	Add more variables results in over-fitting and penalize penalty but does not combat over-fitting	Add more variables results in over-fitting and penalize penalty - combat over-fitting but ends with under-fitting
Limit	Cannot not generalize to new data	Cannot catch variations in train data
Bias-Variance	Prone to high bias and low variance	Prone to high bias and low variance
Asymptotically Equivalence	Leave one out validation	K-fold
Modification	Small Sample: use AICc $$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$	

Feature Selection

Selecting a useful subset from all the features. Why?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
  - Note: Feature Selection is different from Feature Extraction. The latter transforms original features to get a small set of new features
  - More on feature extraction when we cover Dimensionality Reduction

Feature Selection Methods

- Methods agnostic to the learning algorithm
  - Preprocessing based methods
    - E.g., remove a binary feature if its ON in very few or most examples
  - Filter Feature Selection methods
    - Use some ranking criteria to rank features Select the top ranking features
- Wrapper Methods (keep the learning algorithm in the loop) Requires repeated runs of the learning algorithm with different set of features
  - Can be computationally expensive

Optimization – How to optimize?

- Gradient:
- $\nabla_{\boldsymbol{w}}\text{TrainLoss}(\boldsymbol{w})$
- The direction that increases the loss the most.
- Algorithm: Gradient Descent
  - Initialize w
  - For $t = 1 \ldots, T$

$$w \leftarrow w - \eta \nabla_{\boldsymbol{w}}\text{TrainLoss}(\mathbf{w})$$

where η is the constant learning rate (usually less than 1)

  - Objective Function

$$\text{TrainLoss}(\text{w}) = \frac{1}{\text{N}} \sum_{(x,y)\in D_{train}} \left(\mathbf{w^T x} - \text{y}\right)^2$$

Optimization – How to optimize?

- Gradient:

$$\nabla_{\boldsymbol{w}}\text{TrainLoss}(\boldsymbol{w}) = \frac{1}{\text{N}} \sum_{(x,y)\in D_{train}} 2\left(\mathbf{w}^{\mathbf{T}}\mathbf{x} - \text{y}\right)^2\mathbf{x}$$

- Gradient descent update:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla_{\boldsymbol{w}}\text{TrainLoss}(\mathbf{w})$$

- Each iteration requires going over all training examples – expensive when have lots of data

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

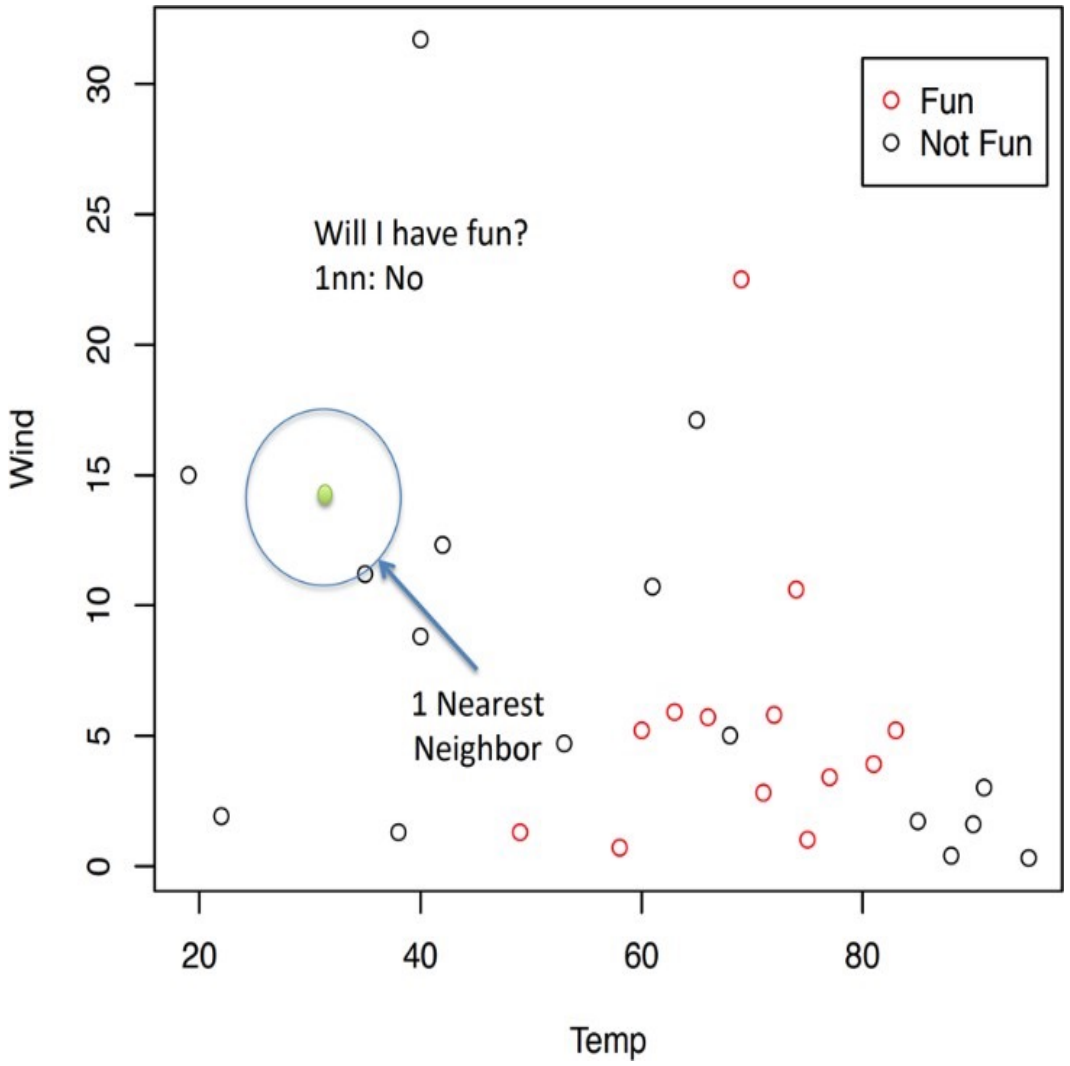
- Is it a good day to go for a run?

- 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)

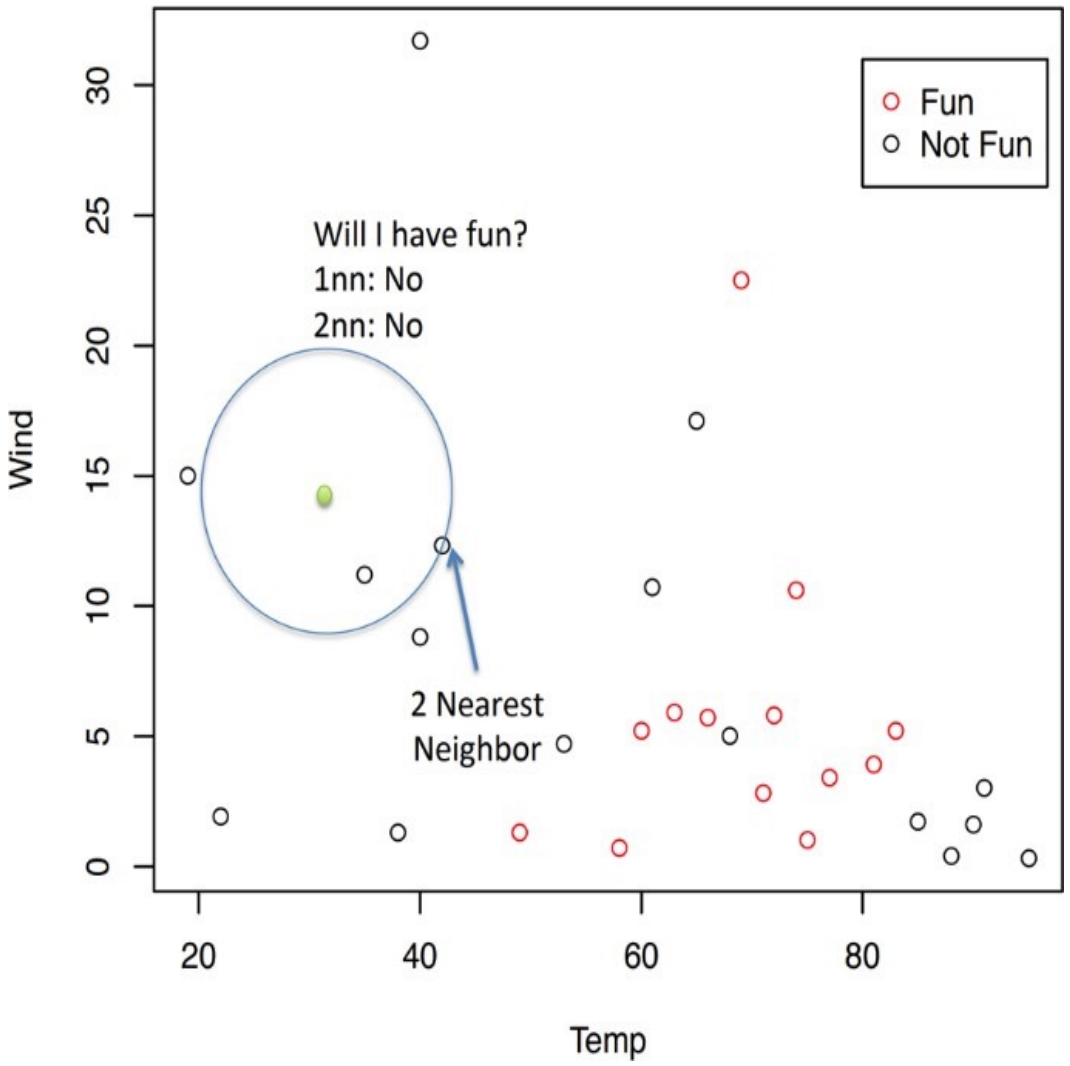
- It is now 65 degrees and the wind is 9 mph. Will a run be fun?

KNN classification

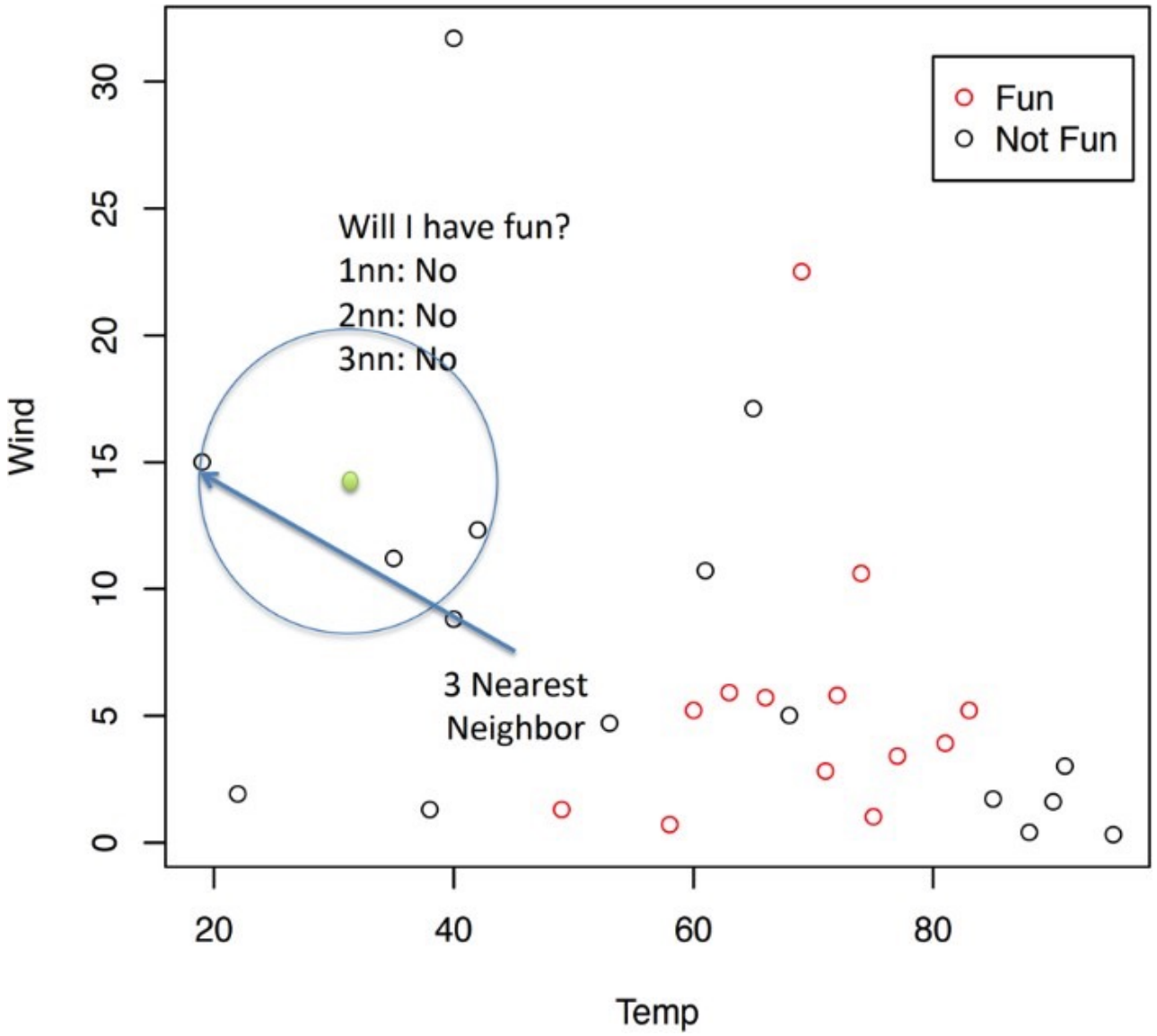
KNN classification: k=1



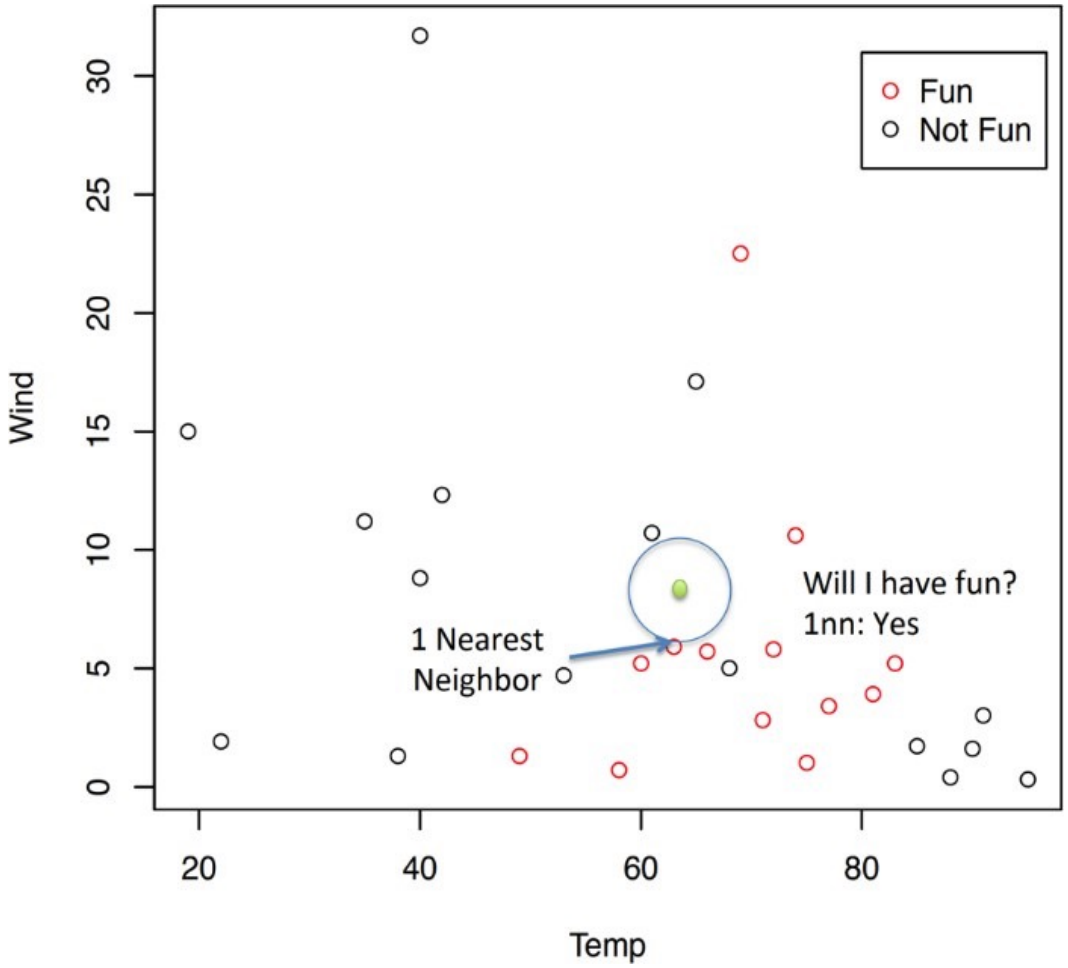
KNN classification: k=2

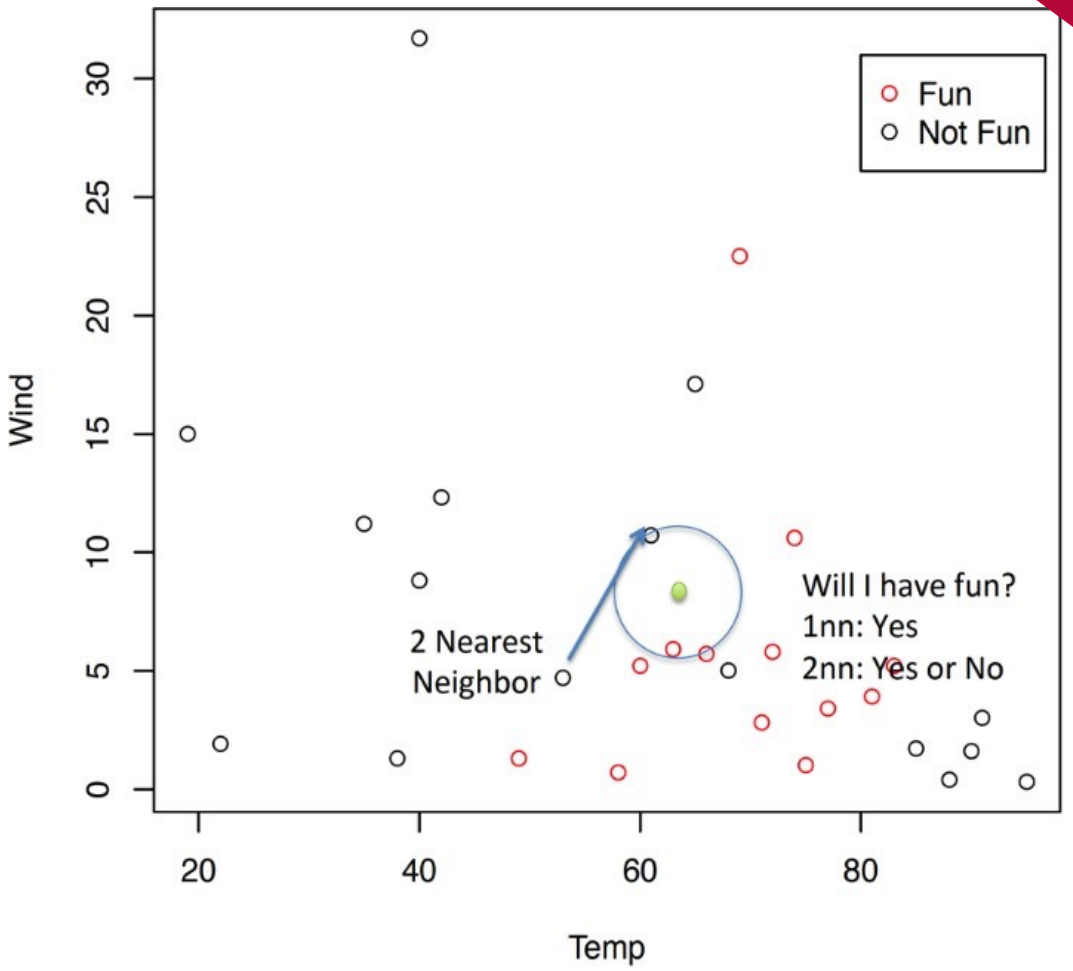


KNN classification: k=3

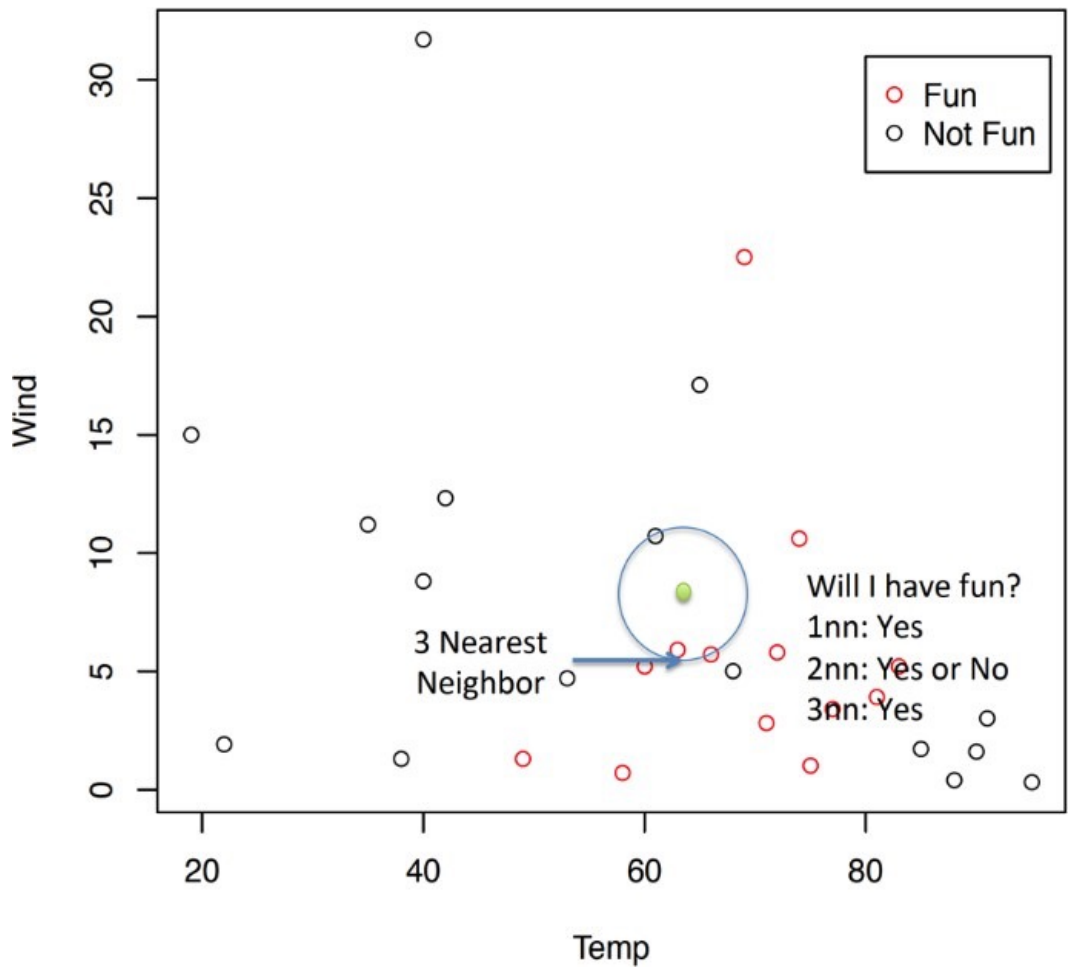


KNN classification: k=1





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 $p(Ck|x)$
- 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

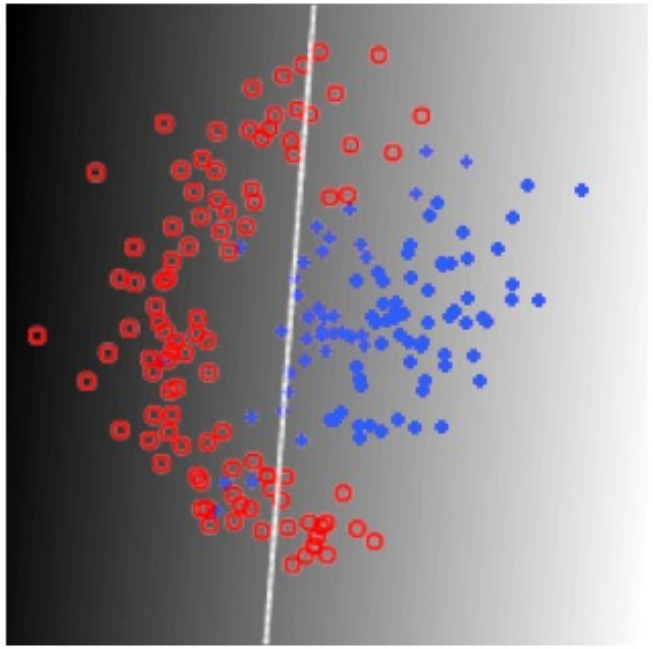
$$p(x) = \sum_k p(\boldsymbol{x}|C_k)p(C_k)$$

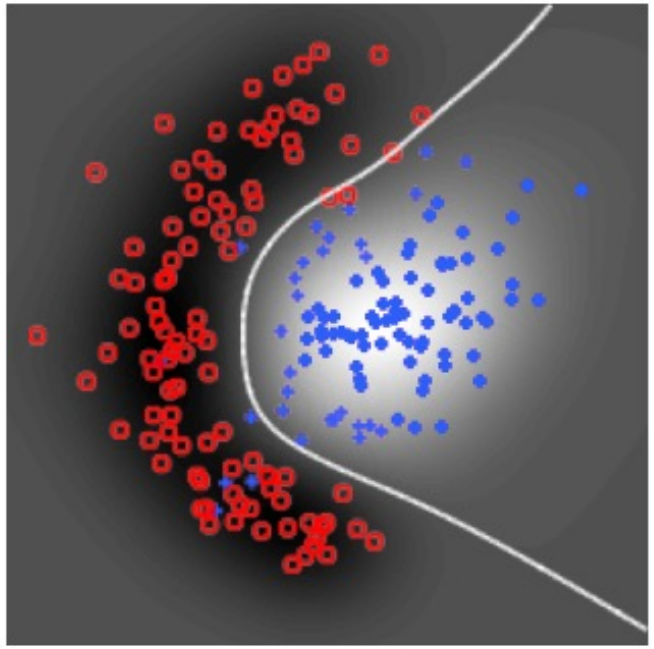
- Use decision theory to determine class membership for each new input *x*.

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

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

- $\boldsymbol{w}$ represents the orthogonal direction to the decision boundary.

Linear Discriminant Functions-Geometrical Properties Cont.

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

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

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

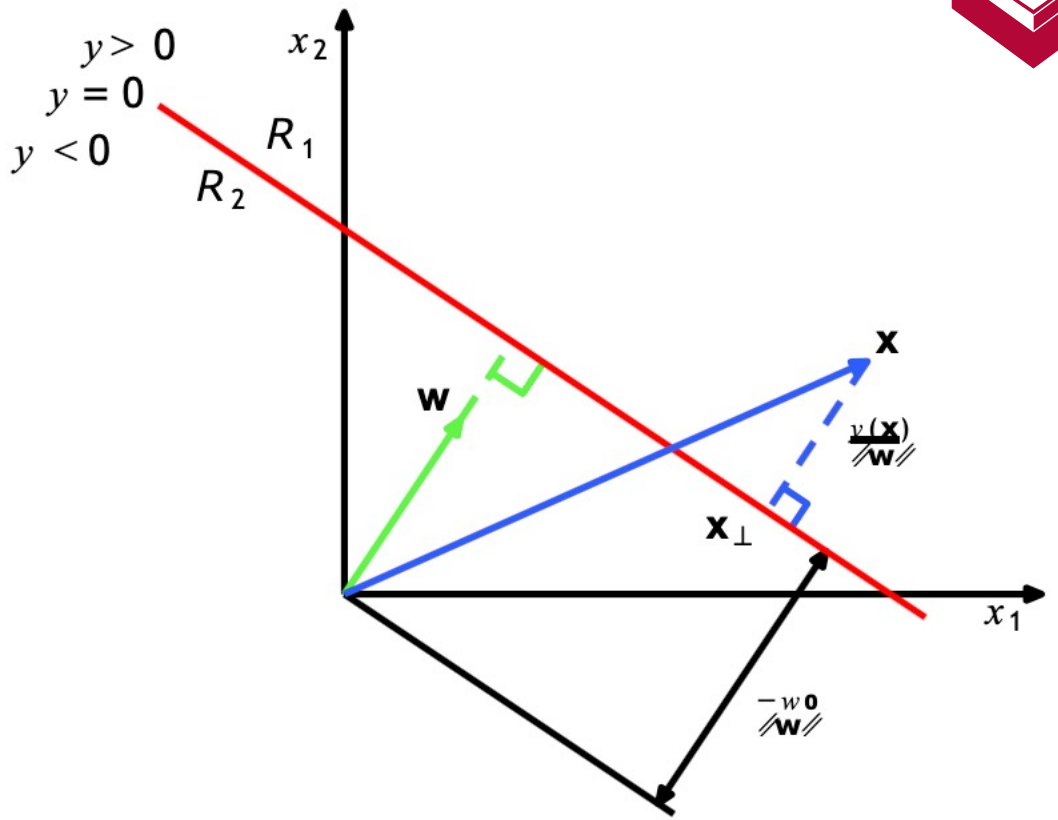
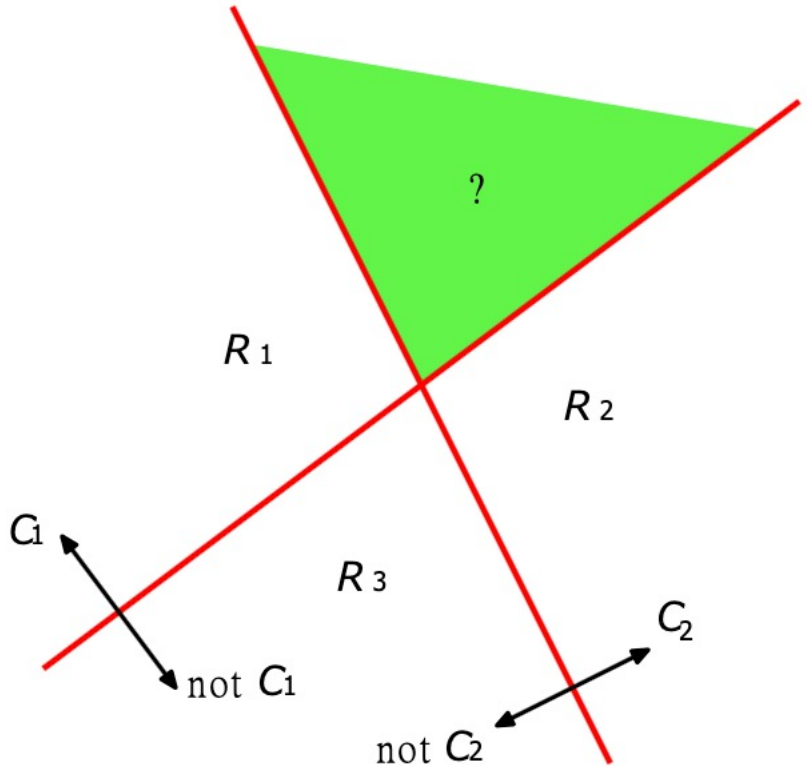


Figure: Signed orthogonal distance of the origin from the decision

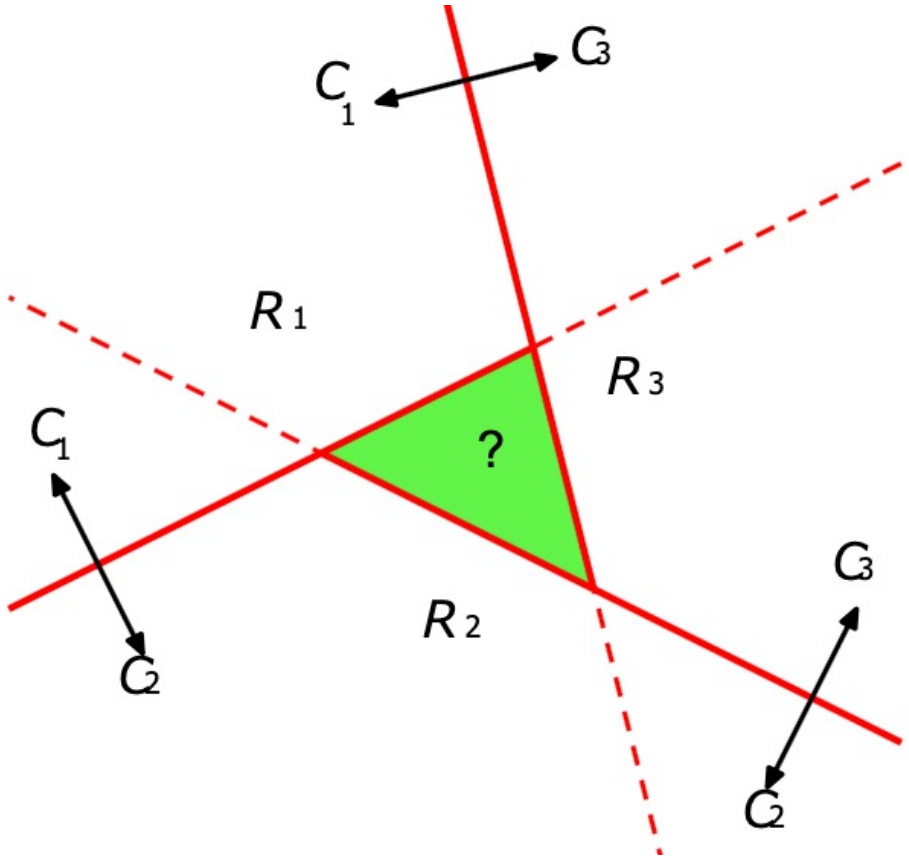
Linear Discriminant Functions: Multiple classes

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.



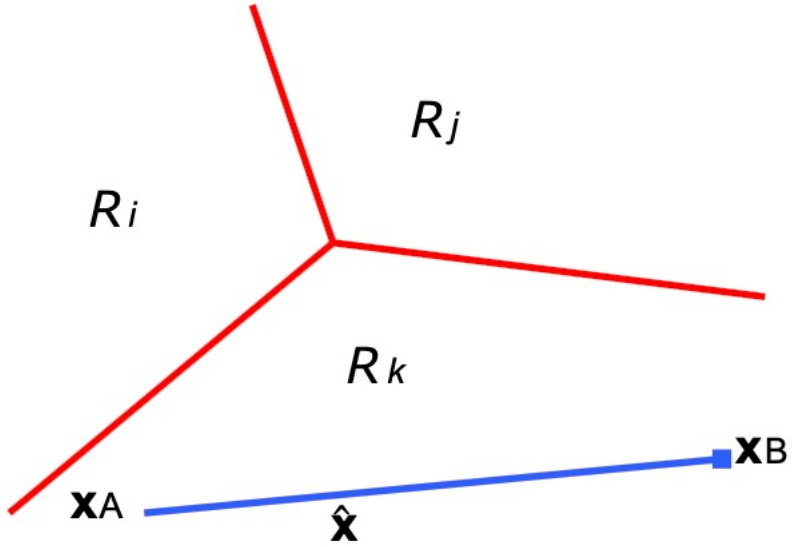
Linear Discriminant Functions: Multiple classes

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

$$f_k(x) = f_j(x) \Rightarrow (wk - w_j)^T x + (wk_0 - w_{j0}) = 0$$

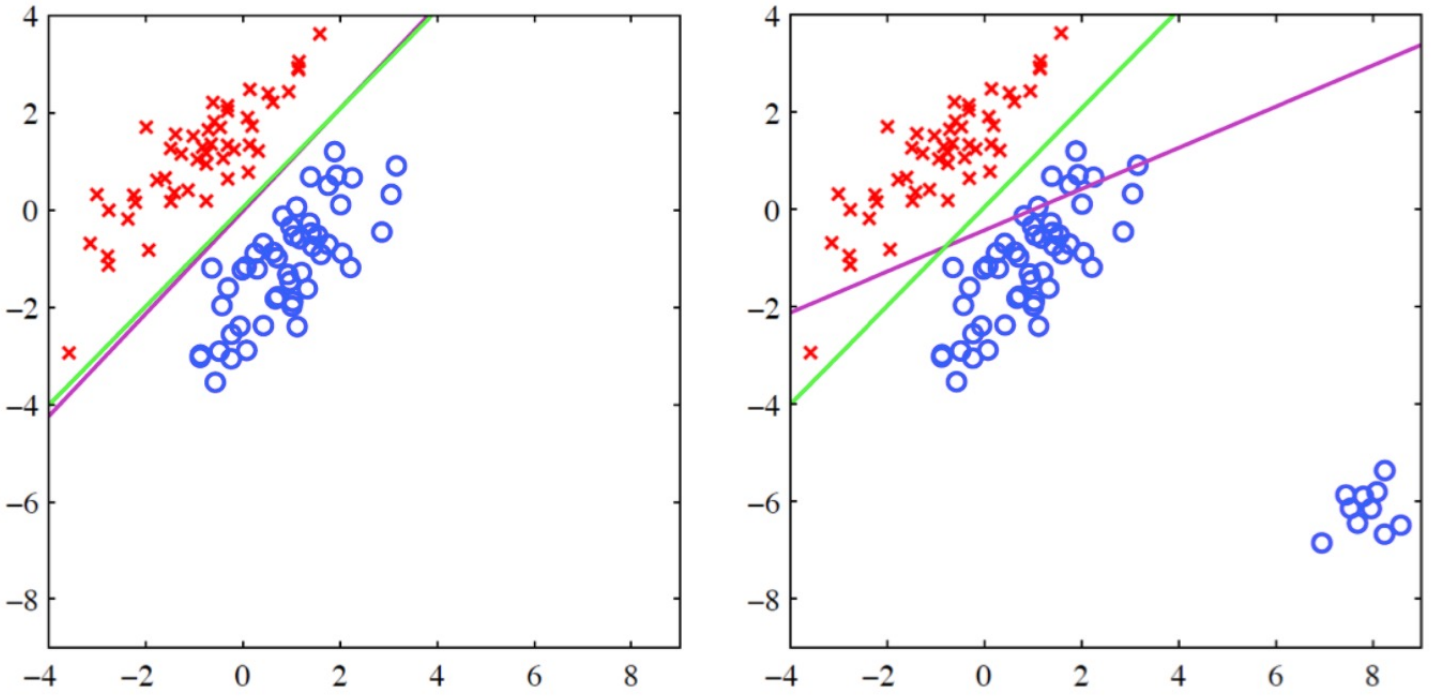


Least square classification

- 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

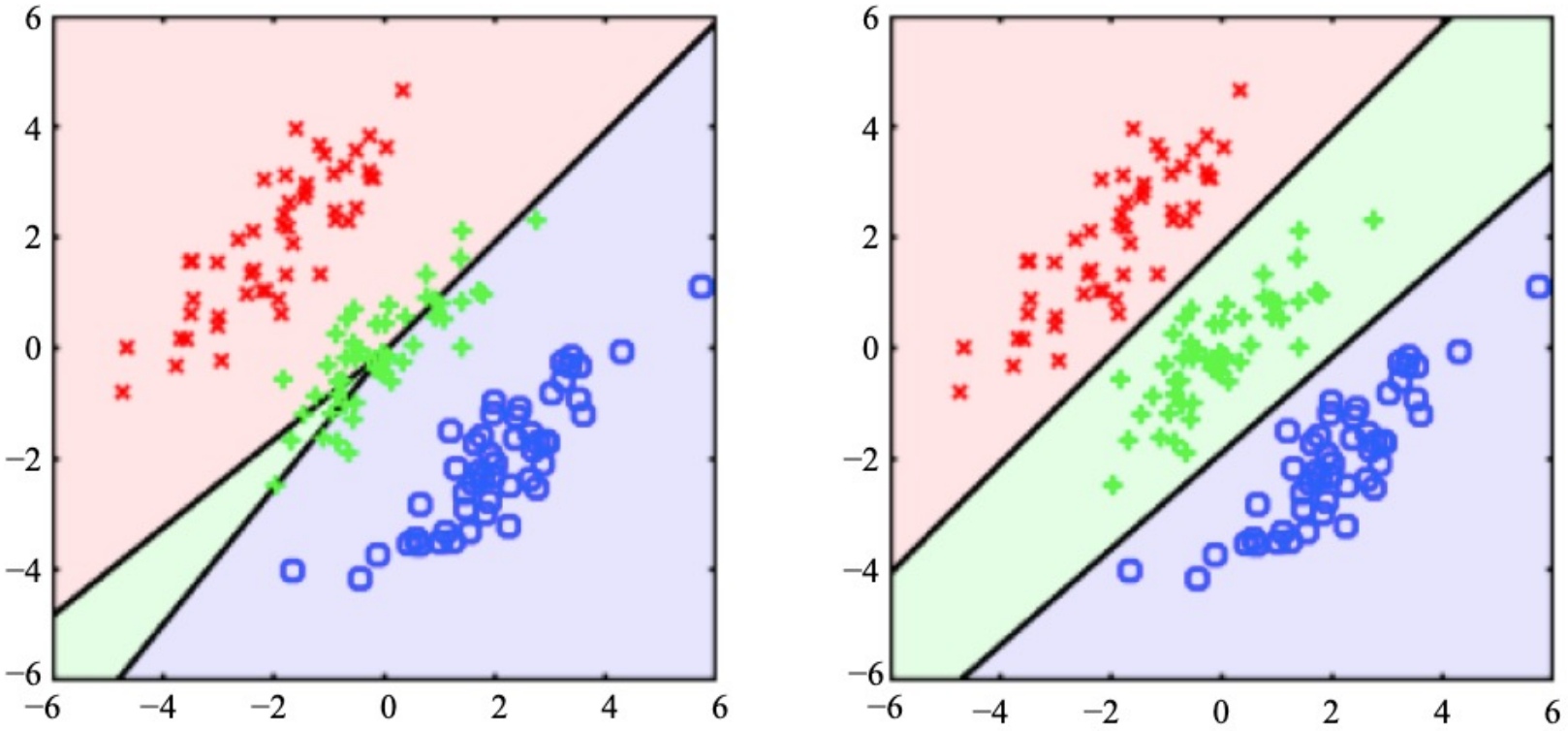
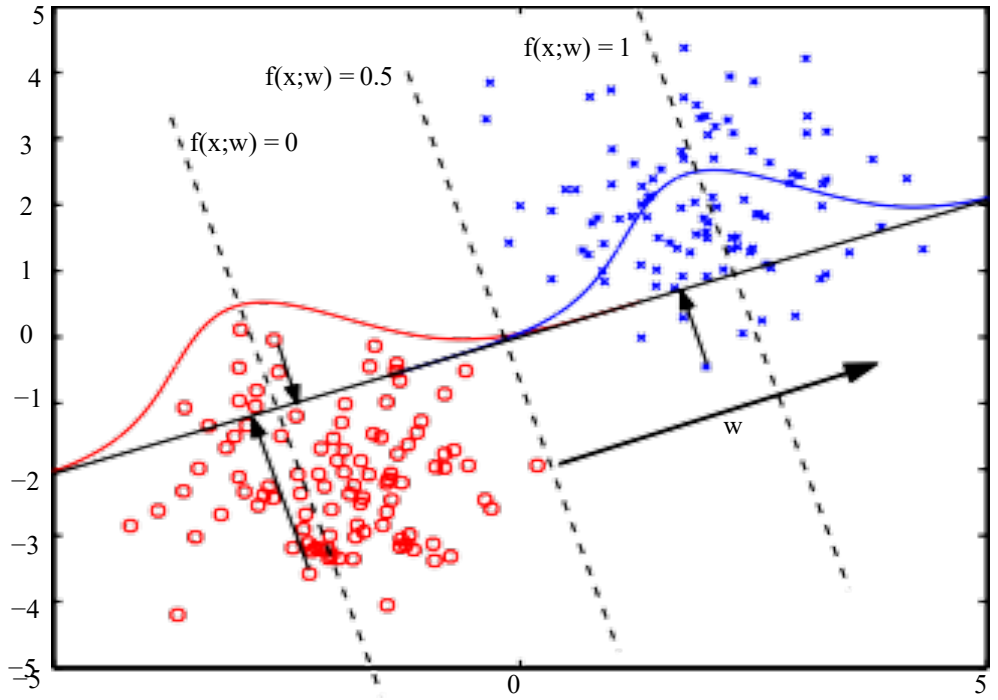


Figure: Left: using a least-squares discriminant; Right: using logistic regression

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 $\boldsymbol{w}$:

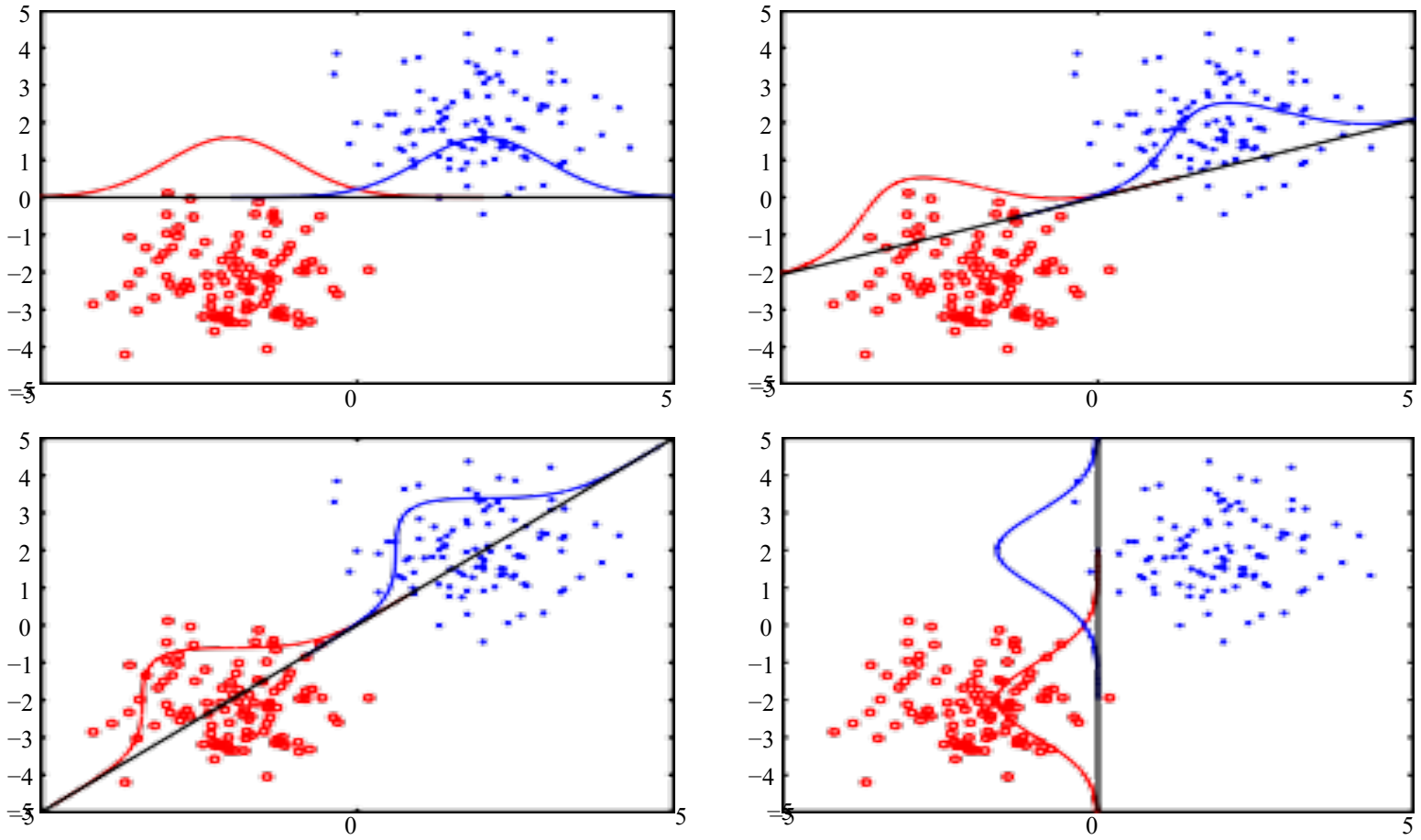
Point in $\boldsymbol{R^d}$	Projected point in R
x_1	$z_1 = w^T x_1$
x_2	$z_2 = w^T x_2$
$\vdots$	$\vdots$
x_n	$z_n = w^T x_n$



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

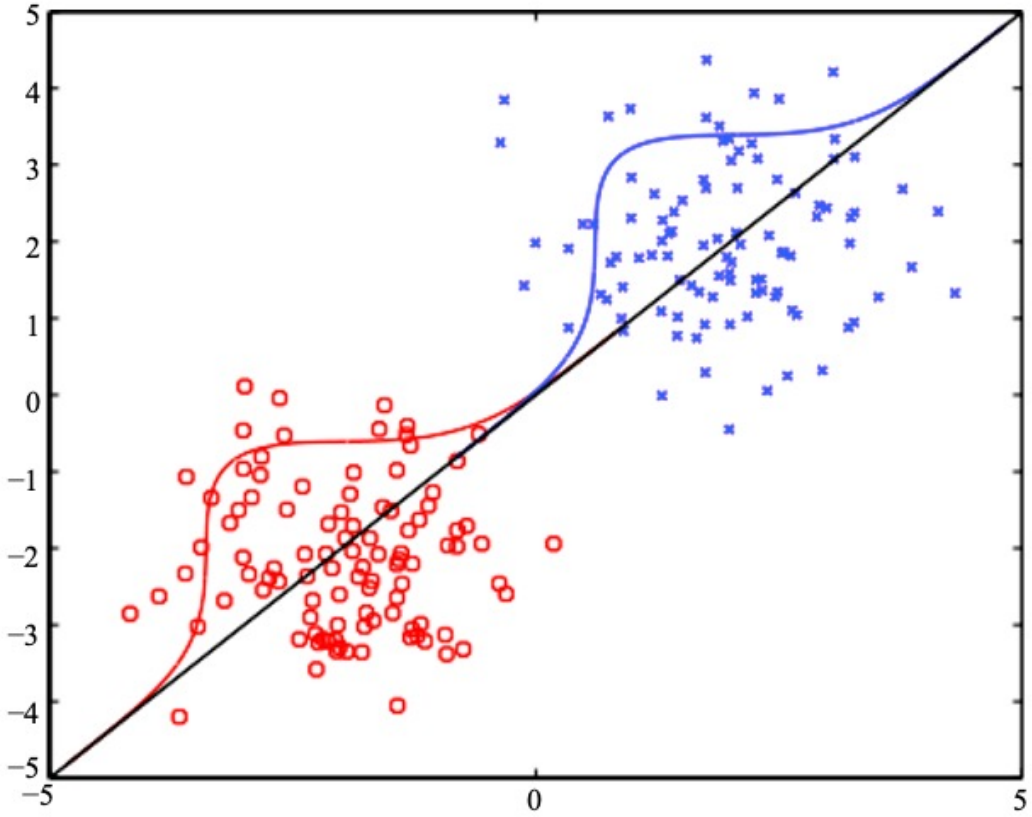
Classification via projection

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

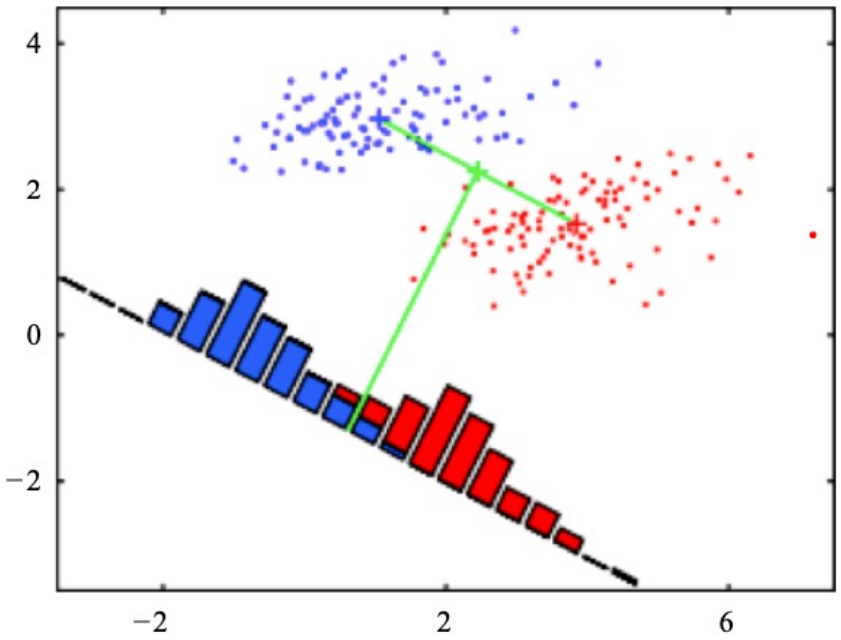
- One way to view a linear classification model is in terms of dimensionality reduction.
- Two class case: suppose we project x onto one dimension:

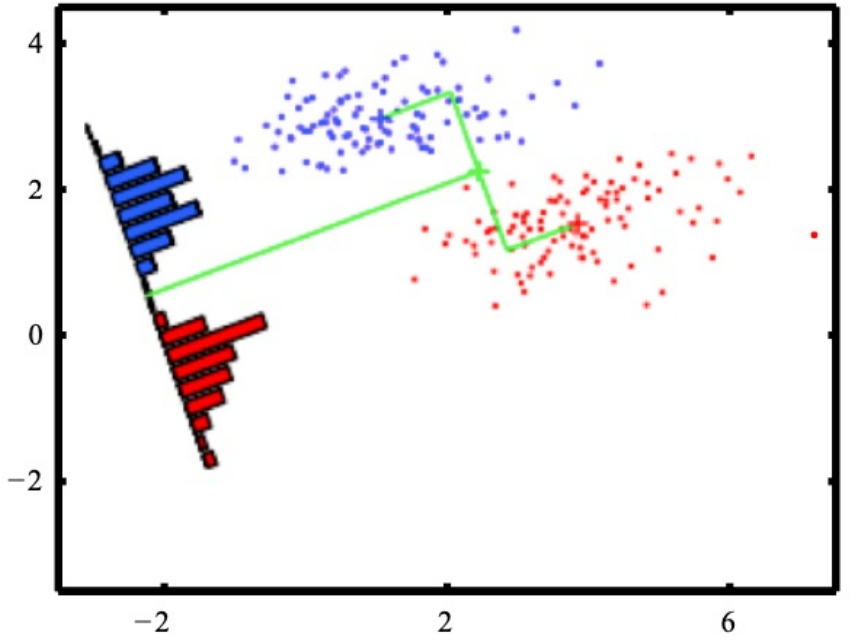
$$f = \boldsymbol{w}^T \boldsymbol{x}$$

- 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:

$$m_1 = \frac{1}{N_1} \sum_{x \in C_1} x$$

- Sample mean for the projected points:

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

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

- 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:

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

- Total within-class scatter of the projected samples:

$$s_1^2 + s_2^2$$

- Fisher linear discriminant analysis:

$$\arg\max_{\mathrm{w}} \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

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:

$$S_1 = \sum_{x \in C_1} (\boldsymbol{x} - \boldsymbol{m_1})(\boldsymbol{x} - \boldsymbol{m_2})^T$$

Within-class scatter matrix:

$$S_w = S_1 + S_2$$

Then:

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

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

Fisher's linear discriminant

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

Thus,

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

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

Fisher's linear discriminant

We have obtained:

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

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

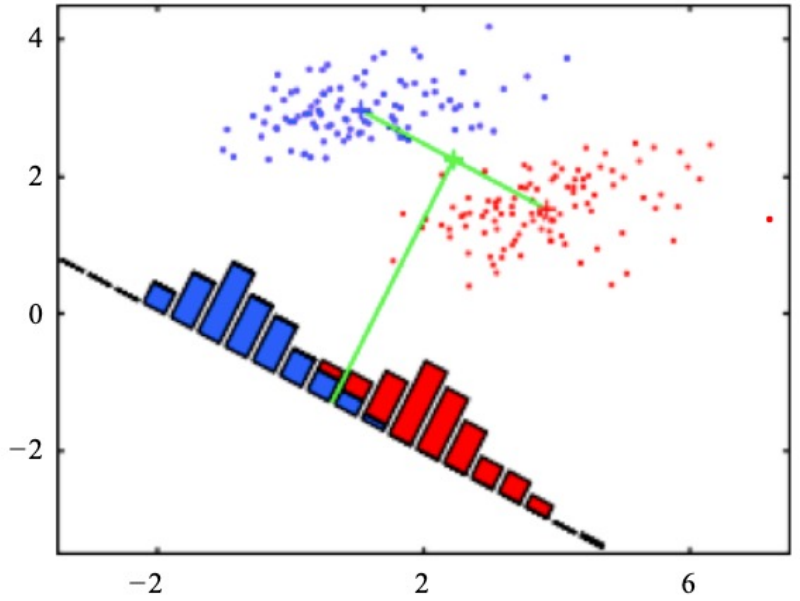
$$\boldsymbol{S_B w = (m_1 - m_2)(m_1 - m_2)^T w}$$

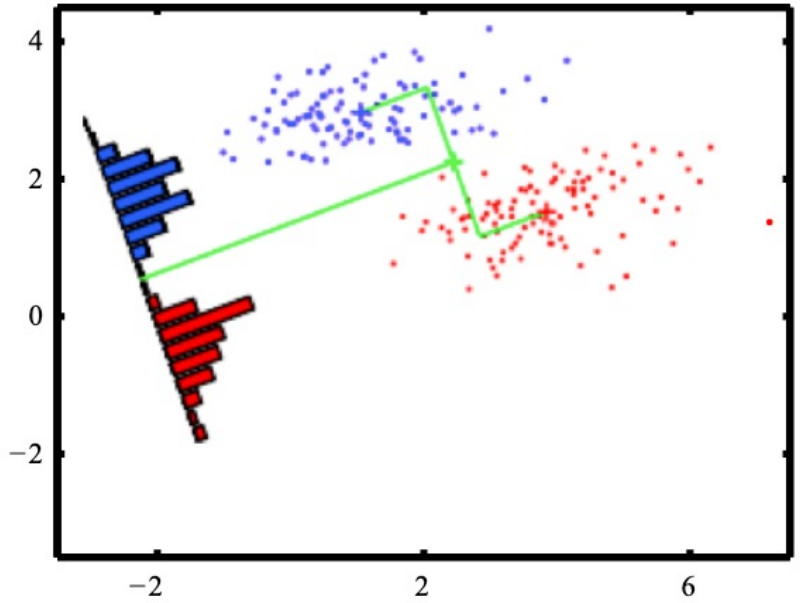
where $\boldsymbol{(m_1 - m_2)^T w}$ is a scalar and always in the direction of $\boldsymbol{(m_1 - m_2)}$.

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):

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

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:

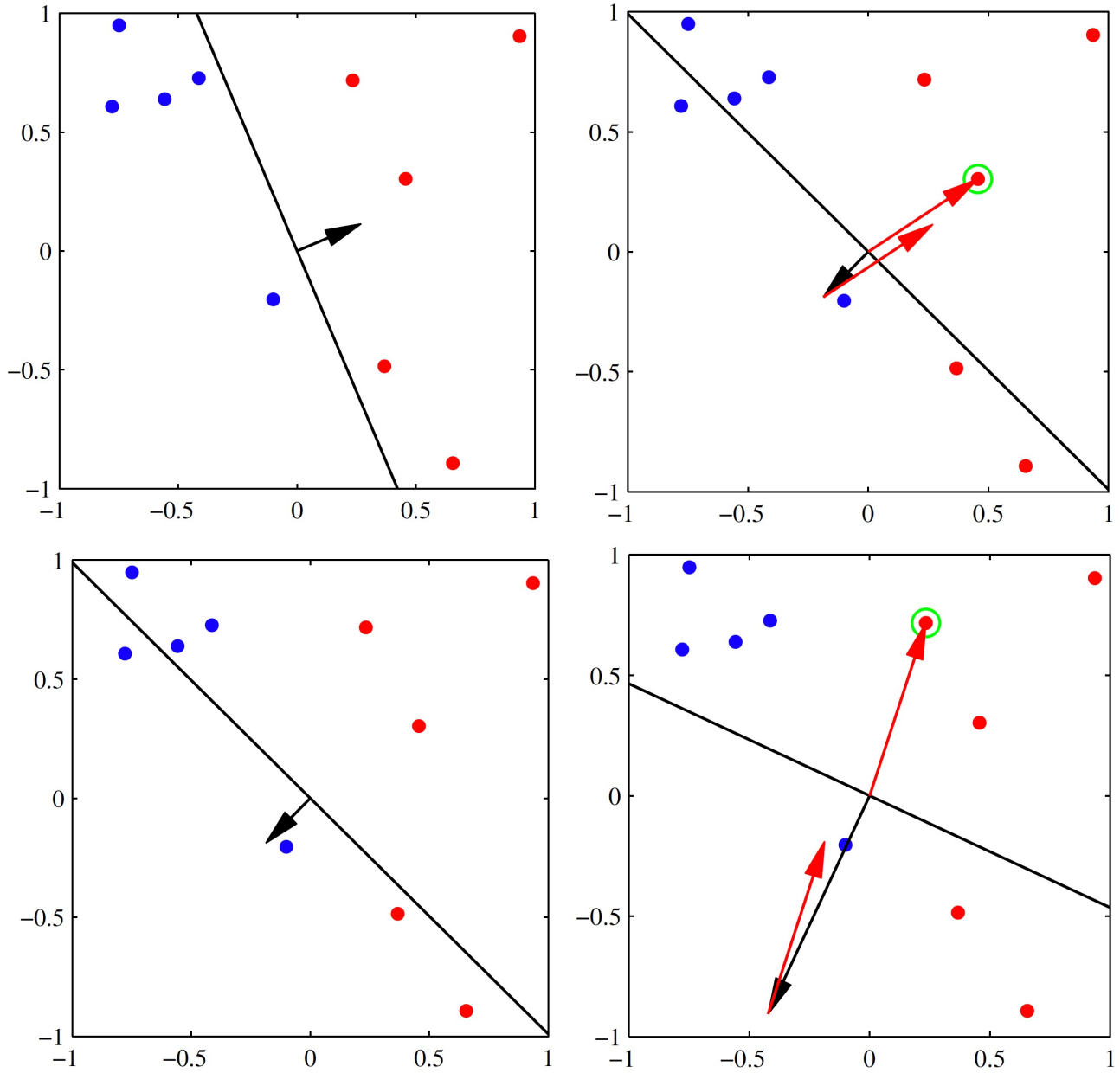
$$\mathbf{w_{k+1}} = \begin{cases} \boldsymbol{w_k} + \boldsymbol{x_k}\ if\ \boldsymbol{x_k} \in C_1 \\ \boldsymbol{w_k} - \boldsymbol{x_k}\ if\ \boldsymbol{x_k} \in C_2 \end{cases}$$

- 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

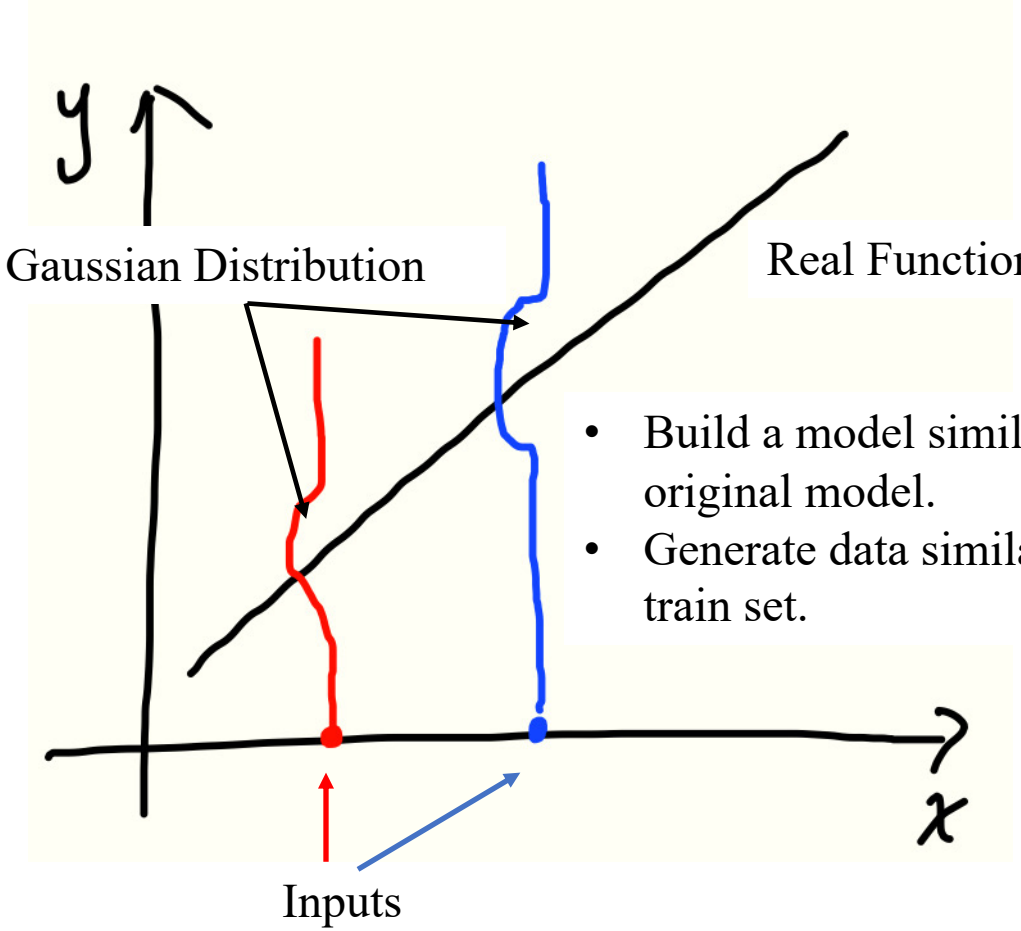
- A and B are independent iff $p(A, B) = p(A)p(B)$
- A and B are conditionally independent given C iff

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

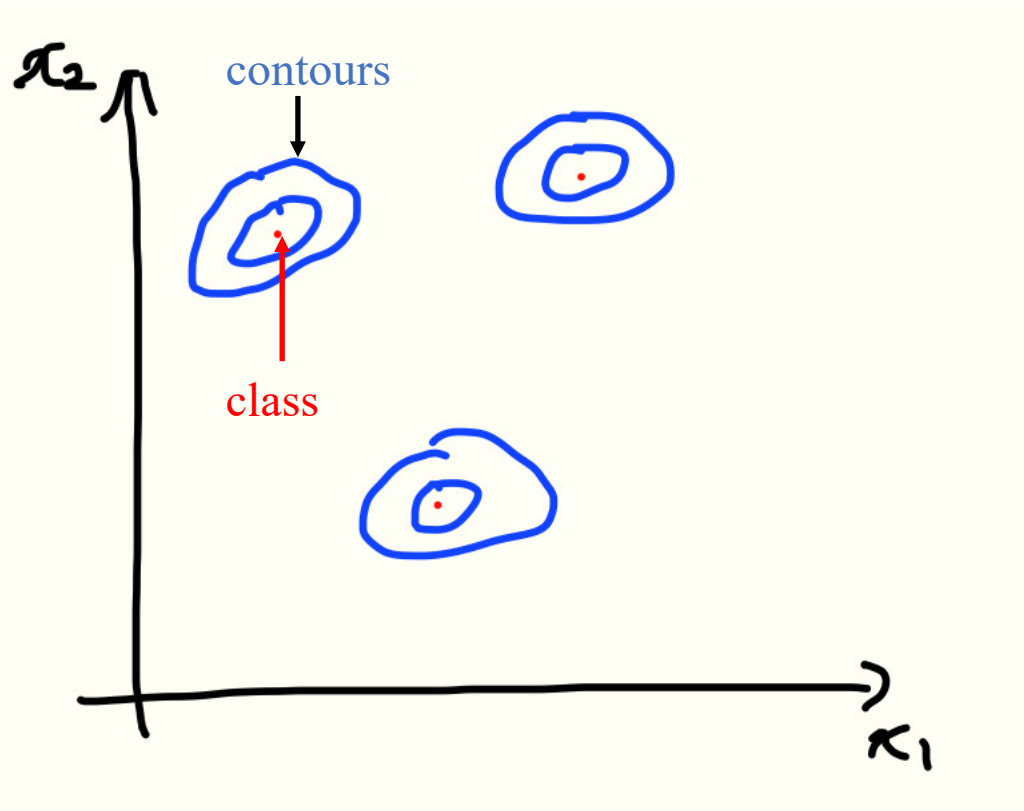
Linear Classification II

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 $x = e^{\ln x}$

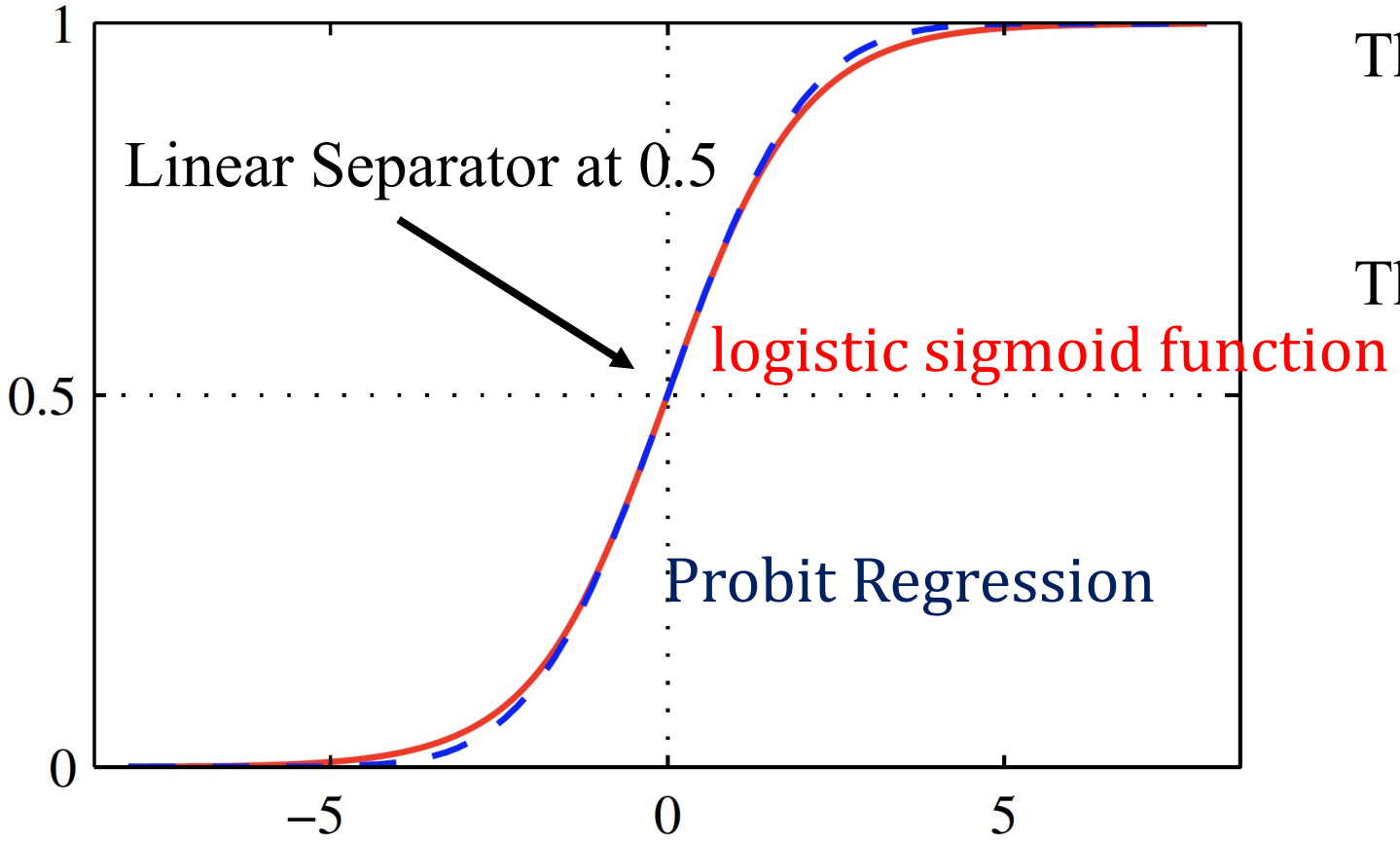
$$= \frac{1}{1 + \pi_j \exp\left\{\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j\right\} / \pi_k \exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right\}}$$

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

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

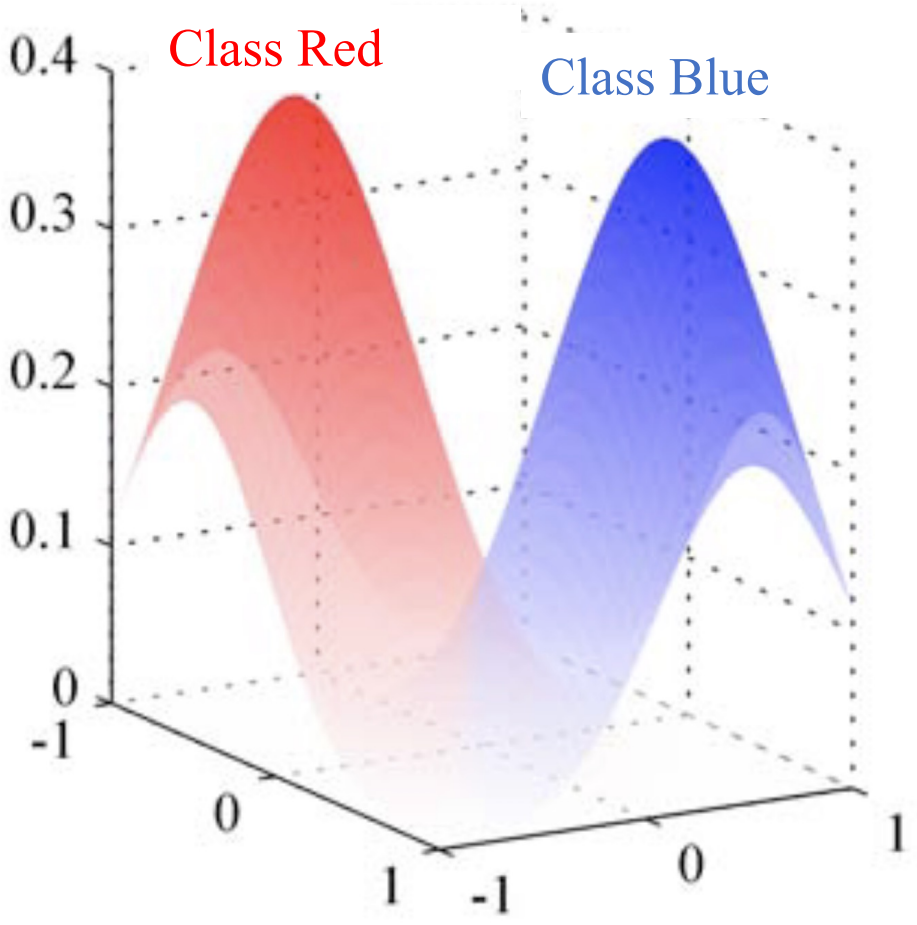
$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

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

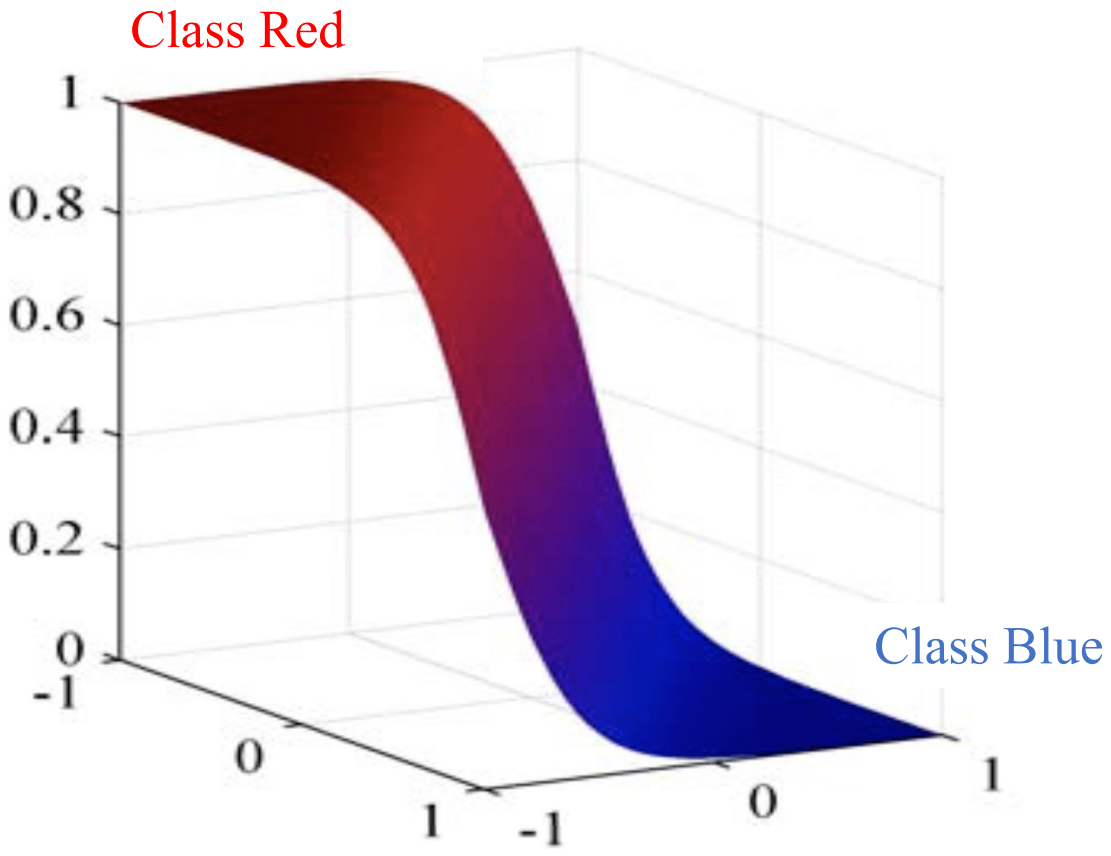
Logistic sigmoid function

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

Class Conditional



Posterior



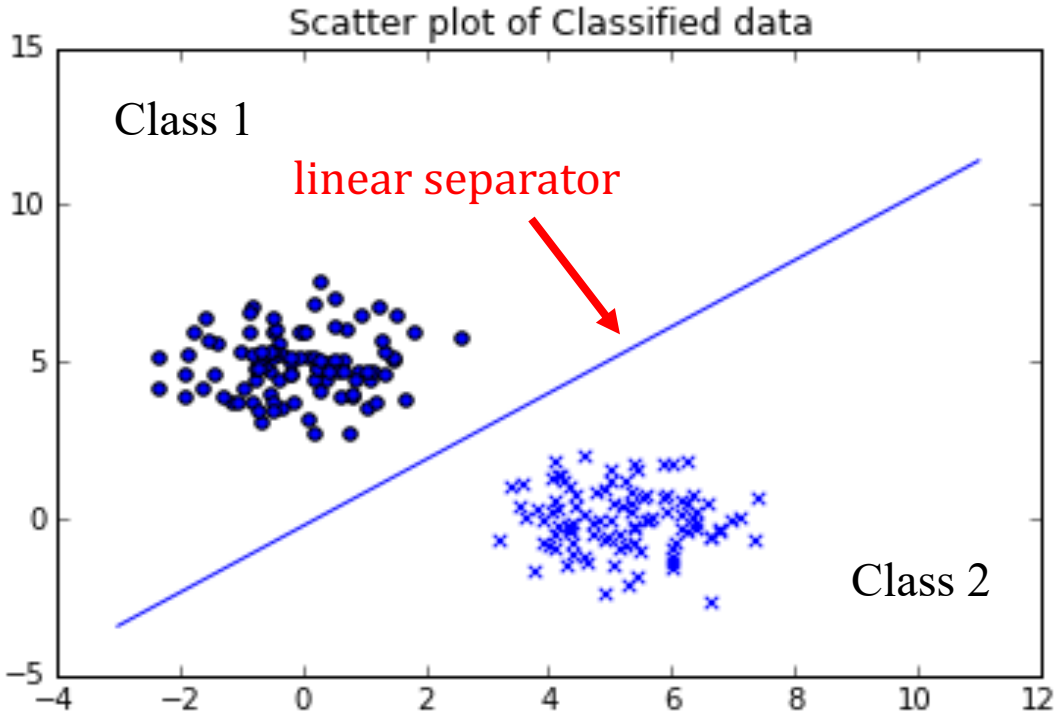
Two Class Prediction

- $best\ class = argmax_k P(C_k|\mathbf{x})$
- $= \begin{cases} c_1 & \sigma(a) \geq 0.5 \\ c_2 & \text{otherwise} \end{cases}$
- 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:

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

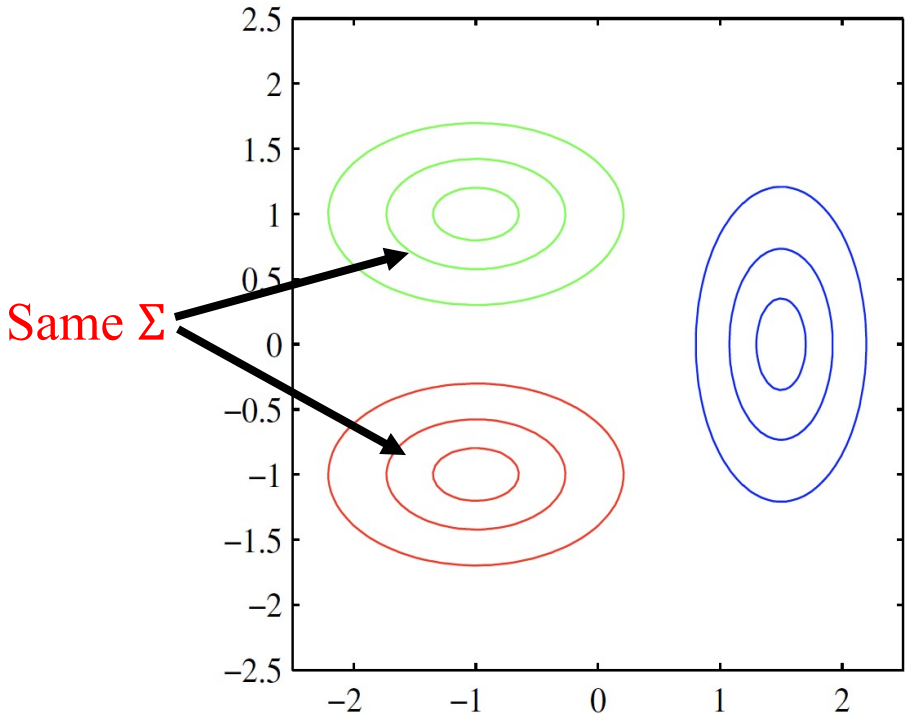
- argmax distribution

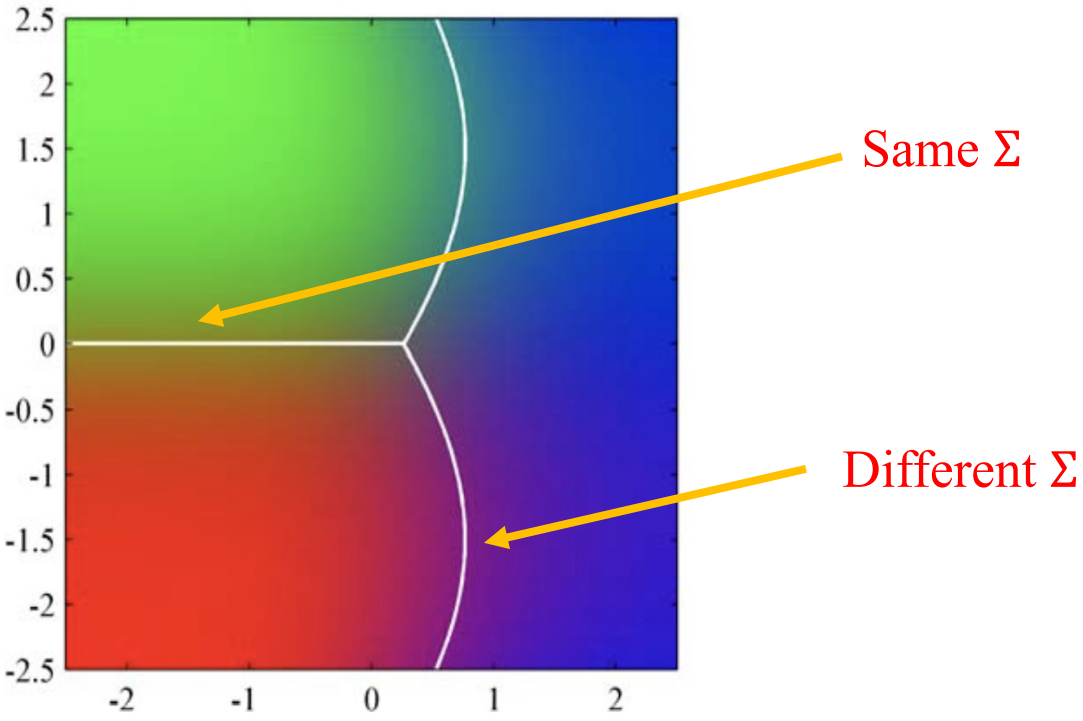
$$p(C_k \mid \boldsymbol{x}) = \begin{cases} 1 & \text{if } k = argmax_j\ f_j(x) \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

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

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

Probabilistic Discriminative Models

Logistic Regression & Generalization

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.

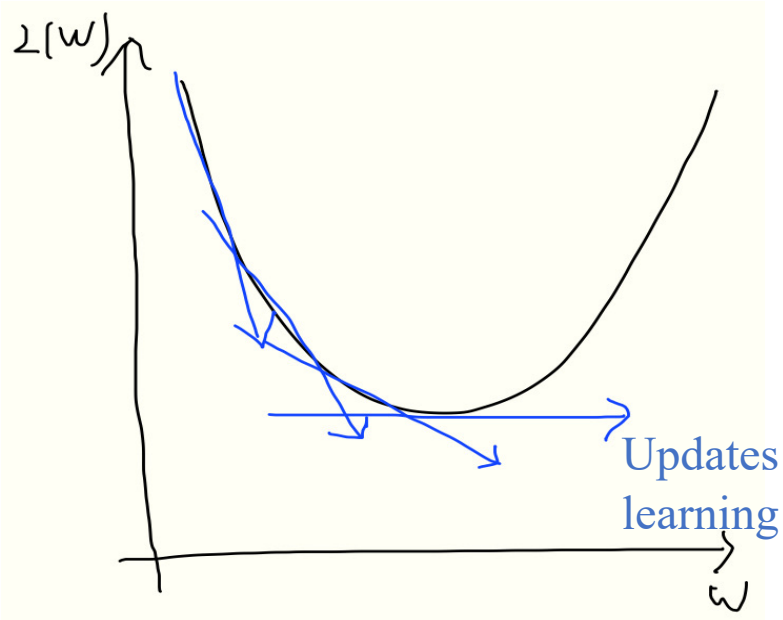
Maximum Likelihood

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

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

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

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



Updates by the linear gradient and the learning rate

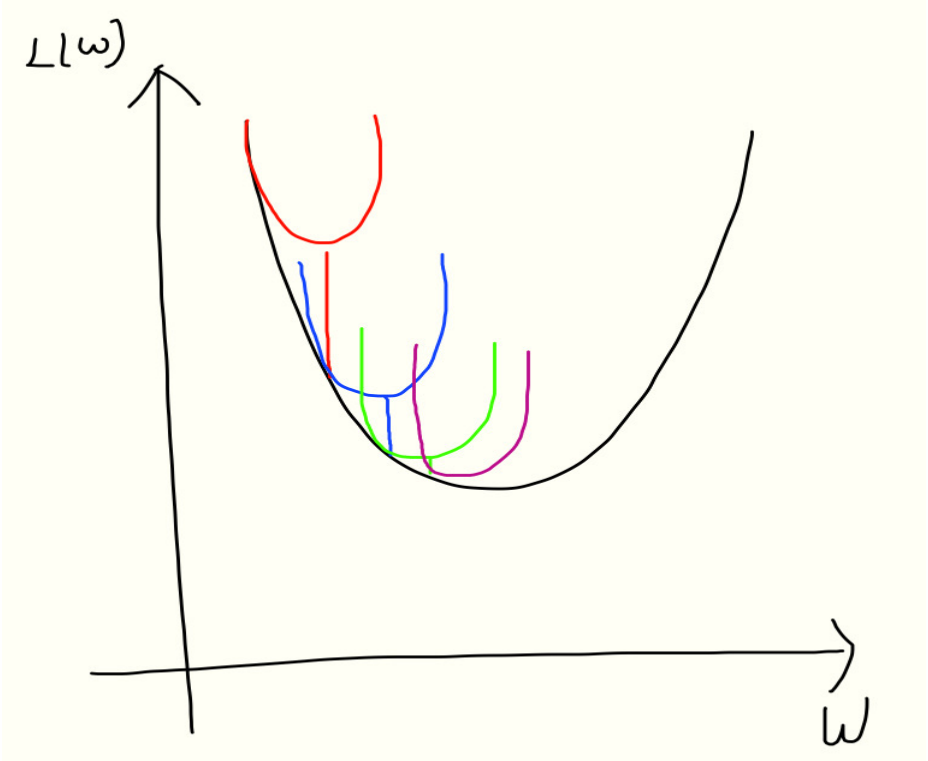
Newton's Method

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

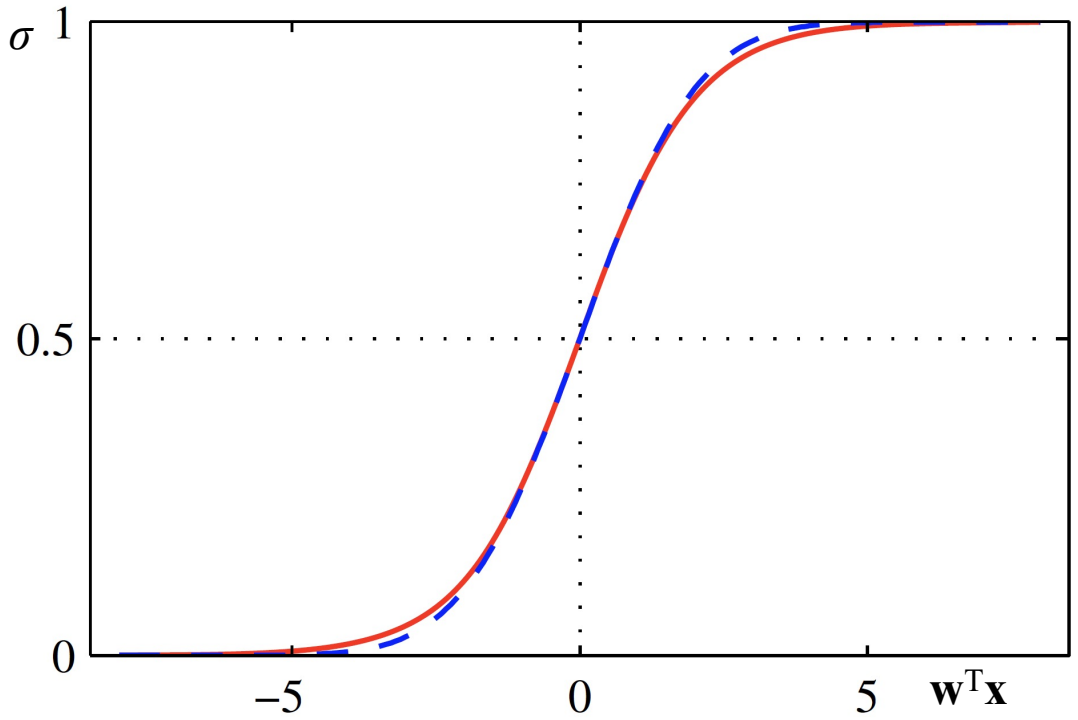
Issues

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, $\mathbf{w}^T\mathbf{x} \rightarrow \infty$ 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

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

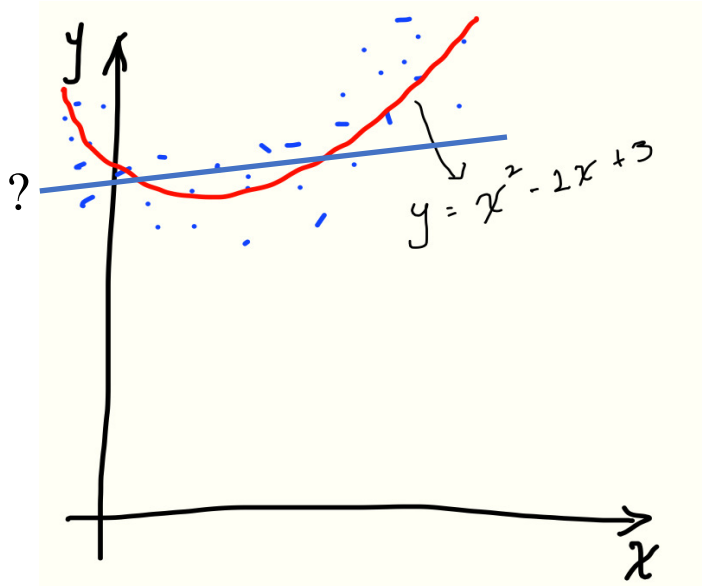
$$= \min_{w}\left\{-\sum_{n=1}^{N}\left[y_n \ln \sigma\left(\mathbf{w^T}\bar{\mathbf{x}}\right) + (1-y_n)\ln\left(1-\sigma\left(\mathbf{w^T}\bar{\mathbf{x}}\right)\right)\right]\right\} + \frac{1}{2}\lambda\mathbf{w^T}\mathbf{w} \tag{28-1}$$

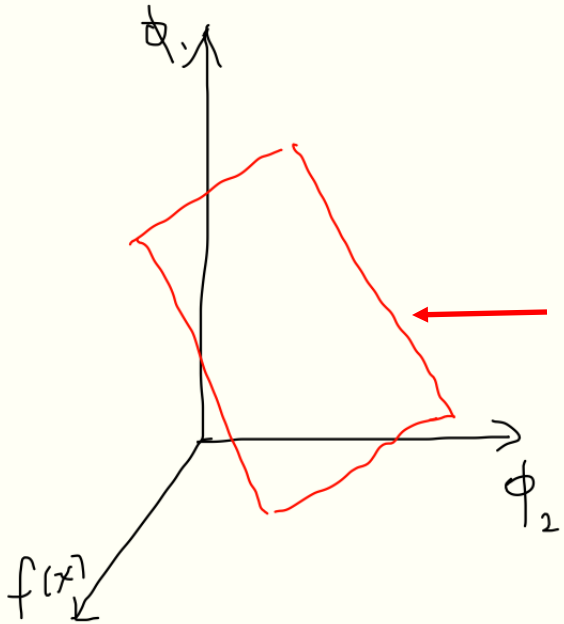
- Hessian: $$H = \bar{\mathbf{X}}\mathbf{R}\bar{\mathbf{X}}^T + \lambda\mathbf{I} \tag{29}$$
- The term $\lambda\mathbf{I}$ ensures that H is not singular (eigenvalues $\geq \lambda$)

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