



LUND UNIVERSITY

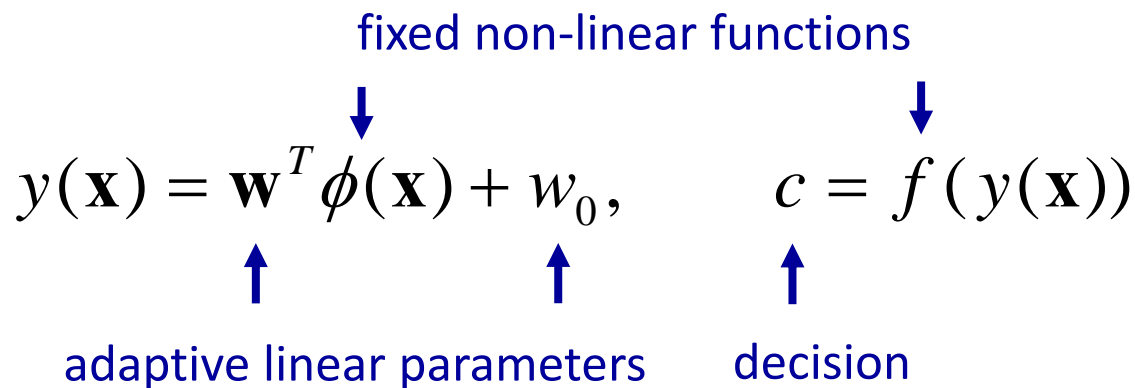
FMAN-45: Machine Learning

Lecture 4: Linear Models for Classification

Cristian Sminchisescu

Linear Classification

- Classification is intrinsically non-linear because of the training constraints that *place non-identical inputs in the same class*
- Differences in the input vector sometimes causes 0 change in the answer
- Linear classification means that the adaptive part \mathbf{w} is linear
 - The adaptive part is cascaded with a fixed non-linearity f
 - It may also be preceded by a fixed non-linearity ϕ when nonlinear basis functions are used


$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0, \quad c = f(y(\mathbf{x}))$$

adaptive linear parameters decision

Approach 1: Discriminant Function

- Use discriminant functions directly, and do not compute probabilities
- Convert the input vector into one or more real values so that a simple process (thresholding, or a majority vote) can be applied to assign the input to the class
- The real values should be chosen to maximize the useable information about the class label present in the real value
- Given discriminant functions $f_1(\mathbf{x}), \dots, f_K(\mathbf{x})$
Classify \mathbf{x} as class C_k , **iff** $f_k(\mathbf{x}) > f_j(\mathbf{x}), \forall j \neq k$

Approach 2: Class-conditional Probabilities

- Infer conditional class probabilities $p(C_k|\mathbf{x})$
- Use conditional distribution to make optimal decisions, e.g. by minimizing some loss function
- Example, 2 classes

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi), \quad p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$$
$$\text{where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$

Approach 3: Class Generative Model

- Compare the probability of the input under separate, class-specific, generative models
- Model both the class conditional densities $p(\mathbf{x}|C_k)$, and the prior class probabilities $p(C_k)$

- Compute posterior using Bayes' theorem

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)}$$

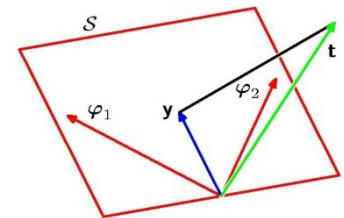
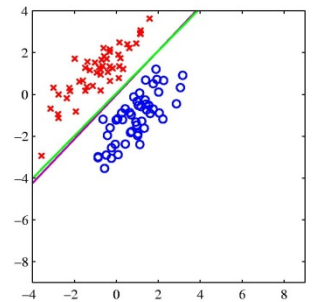
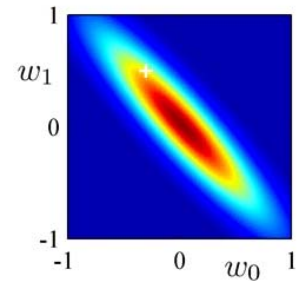
class conditional density class prior

posterior for class

- Example: fit a multivariate Gaussian to the input vectors corresponding to each class, model class prior probabilities by training data frequency counts, and see which Gaussian makes a test data vector most probable using Bayes' theorem

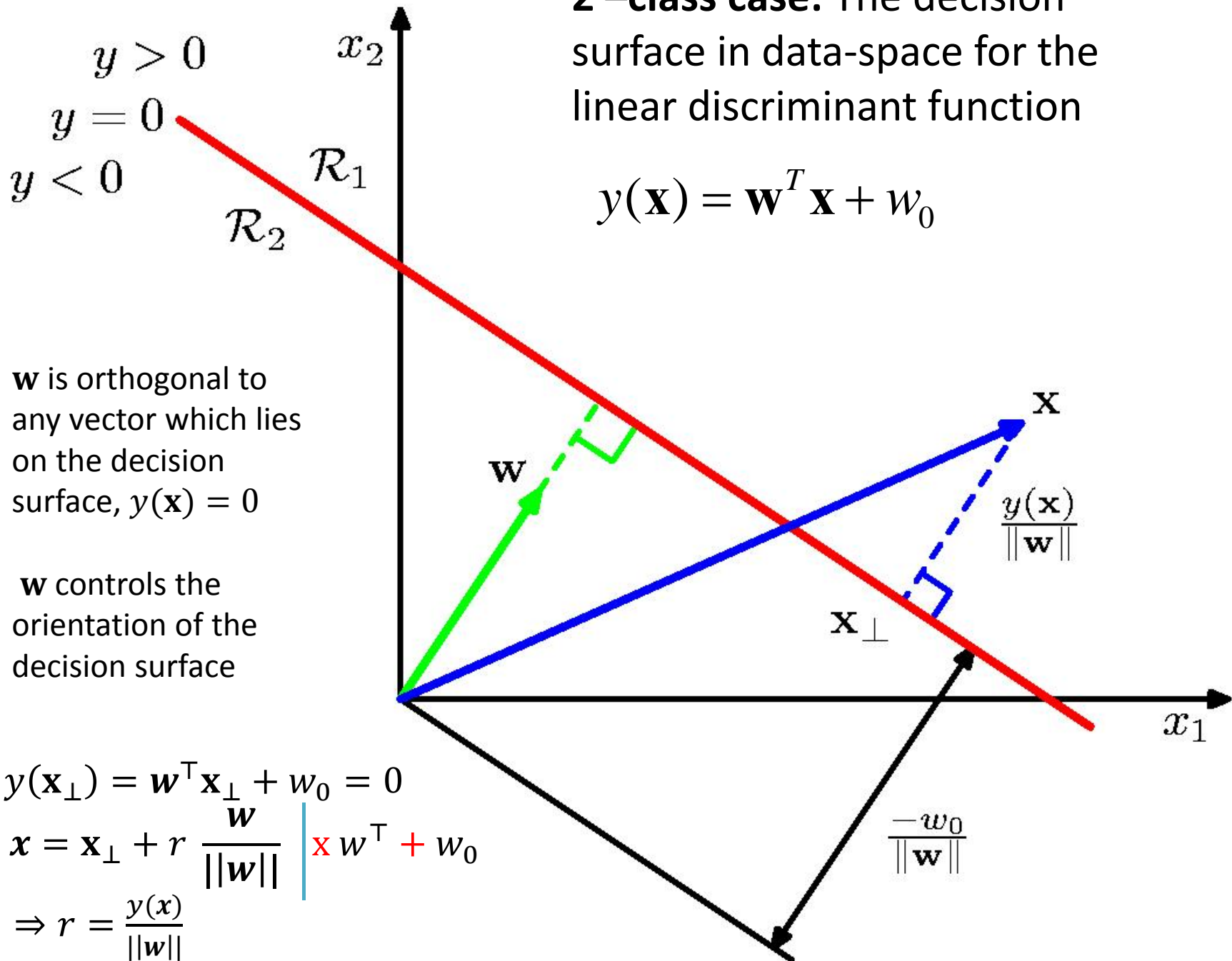
Different Types of Plots in the Course

- **Weight-space**
 - Each axis corresponds to a weight
 - A point is a weight vector
 - Dimensionality = #weights +1 extra dimension for the loss
- **Data-space**
 - Each axis corresponds to an input value
 - A point is a data vector. A decision surface is a plane.
 - Dimensionality = dimensionality of a data vector
- **Case-space** (used for the geometric interpretation of least squares, L3)
 - Each axis corresponds to a training case
 - Dimensionality = #training cases



2 –class case: The decision surface in data-space for the linear discriminant function

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$



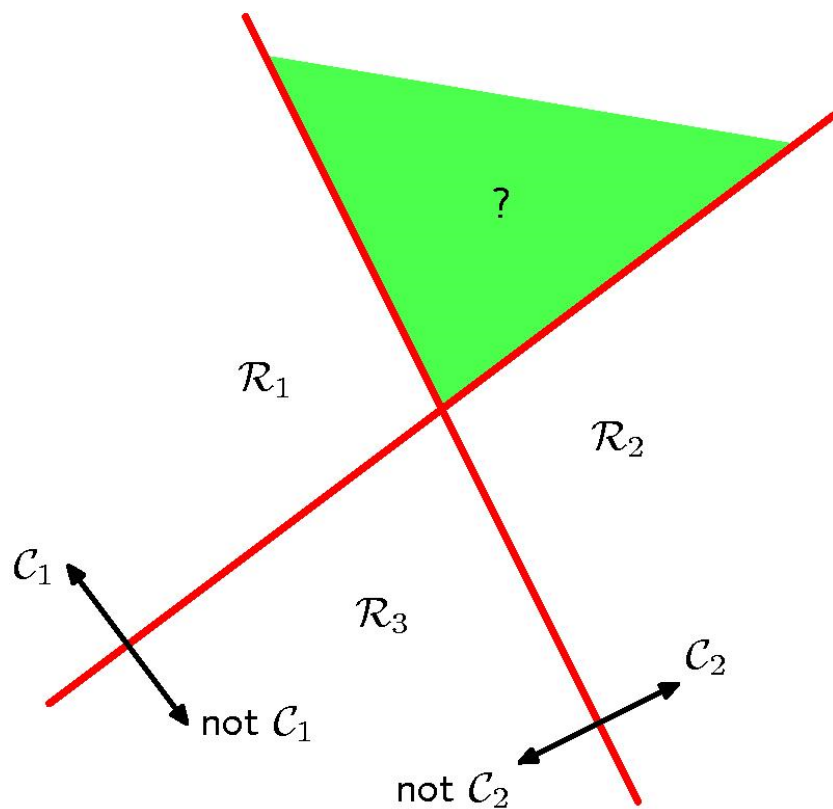
Represent Target Values: *Binary* vs. *Multiclass*

- **Two classes ($N=2$):** typically use a single real valued output that has target values of 1 for the *positive class* and 0 (or -1) for the *negative class*
 - For probabilistic class labels, the target can be the probability of the positive class and the output of the model can be the probability the model assigns to the positive class
- **For the multiclass ($N>2$),** we use a vector of N target values containing a single 1 for the correct class and zeros elsewhere
 - For probabilistic labels we can then use a vector of class probabilities as the target vector

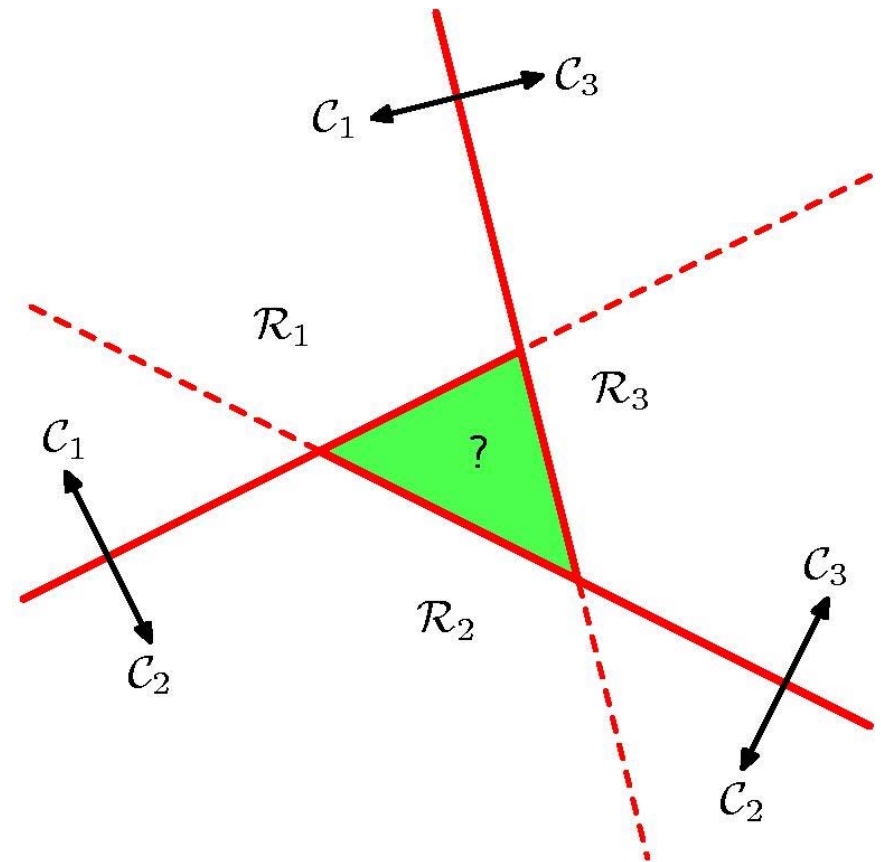
Discriminant Functions for Multiclass

- One possibility is to use N binary (2-way) discriminants
 - Each function separates one class from the rest
- Another possibility is to use $\frac{N(N-1)}{2}$ binary (2-way) discriminants
 - Each function discriminates between two specific classes. We have 1 discriminant for each class pair
- Both methods have ambiguities

Problems with Multi-class Discriminant Functions Constructed from Binary Classifiers



1 - vs.- all



1 vs. 1

If we base our decision on binary classifiers, we can encounter ambiguities

Simple Solution

Use N discriminant functions, y_i, y_j, y_k, \dots , and **take the *max* over their response. Why?**

- Consider linear discriminants y

$$y_k(\mathbf{x}) = w_k^T \mathbf{x} + w_{k0}$$

- The decision boundary between class k and j is given by the $D - 1$ hyperplane

$$(w_k^T - w_j^T) \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

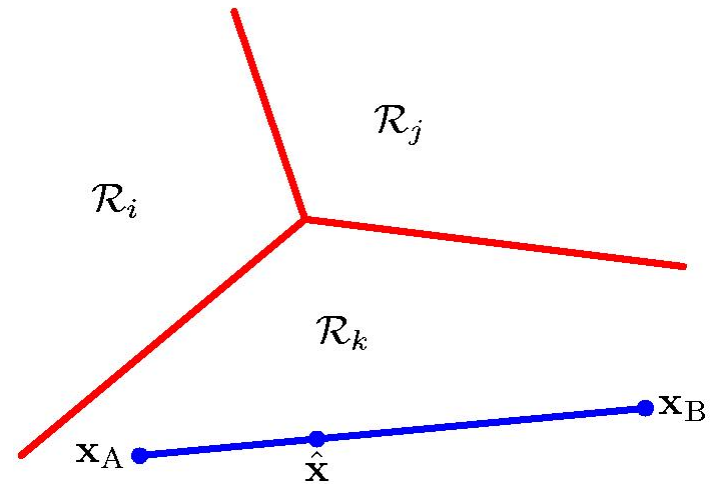
In this linear case the decision regions are convex

$$\mathbf{x}_A, \mathbf{x}_B \in R_k, \hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda) \mathbf{x}_B, 0 \leq \lambda \leq 1$$

$$\text{From the linearity of } y \Rightarrow y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B)$$

$$\text{But } y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A) \text{ and } y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B) \quad \forall j \neq k \Rightarrow y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}}) \\ \Rightarrow \hat{\mathbf{x}} \text{ also lies inside } R_k$$

Hence R_k is convex



Least Squares for Classification

- This is not necessarily the right approach in principle, and it does not work as well as more advanced methods, but is simple
 - It reduces classification to least squares regression
 - We already know how to do regression. We can solve for the optimal weights using the normal equations (L3)
- We set the target to be the conditional probability of the class given input
 - When more than two classes, we treat each class as a separate problem
- Consider a general classification problem with K classes, with a 1-of- K binary coding scheme for the target vector \mathbf{t} .
- The justification for using least squares is that it approximates the conditional expectation $E[\mathbf{t}|\mathbf{x}]$. For the binary coding scheme, this expectation is given by the vector of posterior probabilities. Unfortunately these are approximated rather poorly — e.g. values outside the range $(0,1)$ —, due to the limited flexibility of the model

Least Squares Classification

Assume each class has its own linear model: $y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$

Then we can write:

$$y(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}, \text{ with } k\text{-th column a } D + 1\text{-dim vector } \tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^T)^T, \tilde{\mathbf{x}} = (1, \mathbf{x}^T)^T$$

Given $\{\mathbf{x}_n, \mathbf{t}_n\}, n = 1, \dots, N$; n -th row of \mathbf{T} is \mathbf{t}_n^T ; $\tilde{\mathbf{X}}$'s n -th row is $\tilde{\mathbf{x}}_n^T$

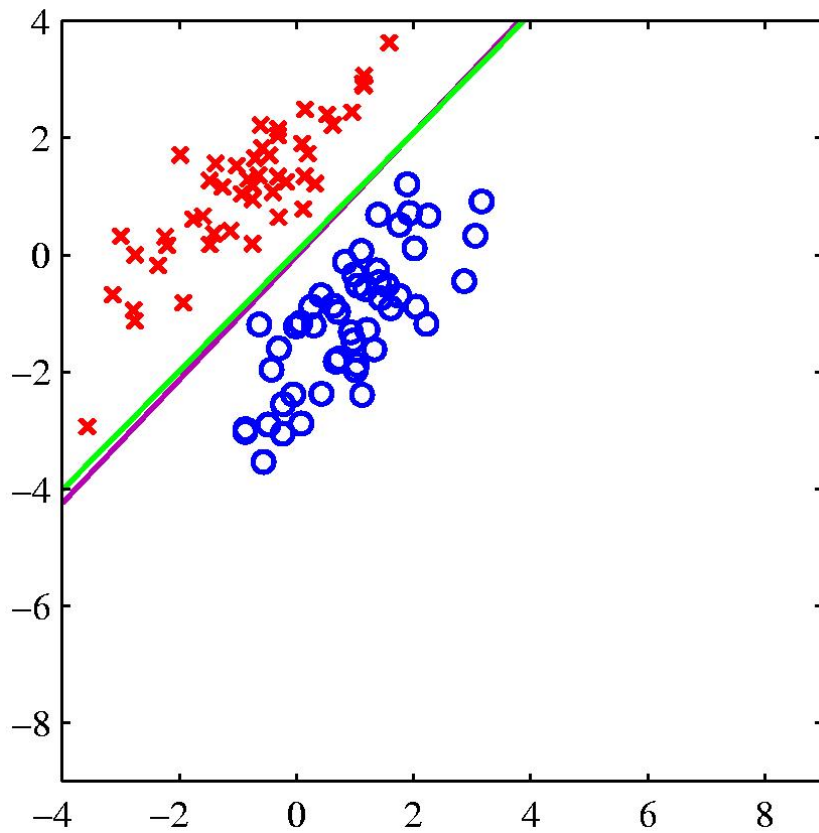
The sum of squares error function for classification is:

$$\begin{aligned} E_D(\tilde{\mathbf{W}}) &= \frac{1}{2} \text{Tr}\{(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})\} \\ \frac{\partial E_D}{\partial \tilde{\mathbf{W}}} &= 0 \Rightarrow \tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T} = \tilde{\mathbf{X}}^+ \mathbf{T} \quad \mathbf{X}^+ \text{ is the pseudoinverse of } \mathbf{X} \\ y(\mathbf{x}) &= \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = \mathbf{T}^T (\tilde{\mathbf{X}}^+)^T \tilde{\mathbf{x}} \quad \leftarrow \text{Closed-form solution} \end{aligned}$$

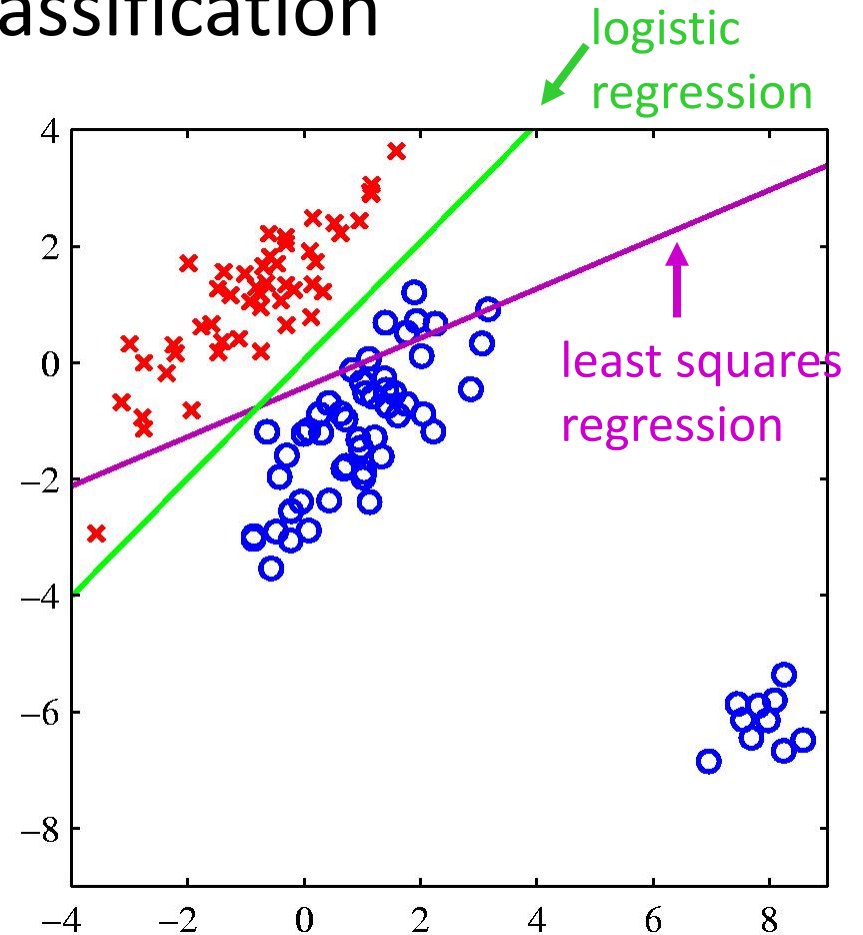
Property: every vector in the training set and the model prediction for any value of \mathbf{x} , satisfy some linear constraint:

$$\mathbf{a}^T \mathbf{t}_N + b = 0, \mathbf{a}^T y(\mathbf{x}) + b = 0, \text{ for some constants } \mathbf{a}, b.$$

Problems with using least squares for classification

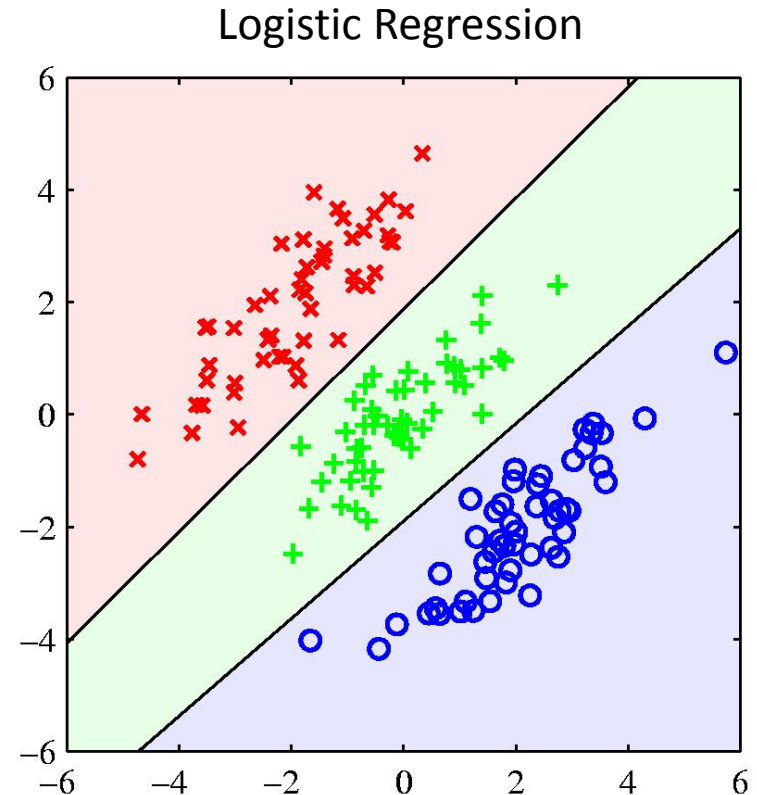
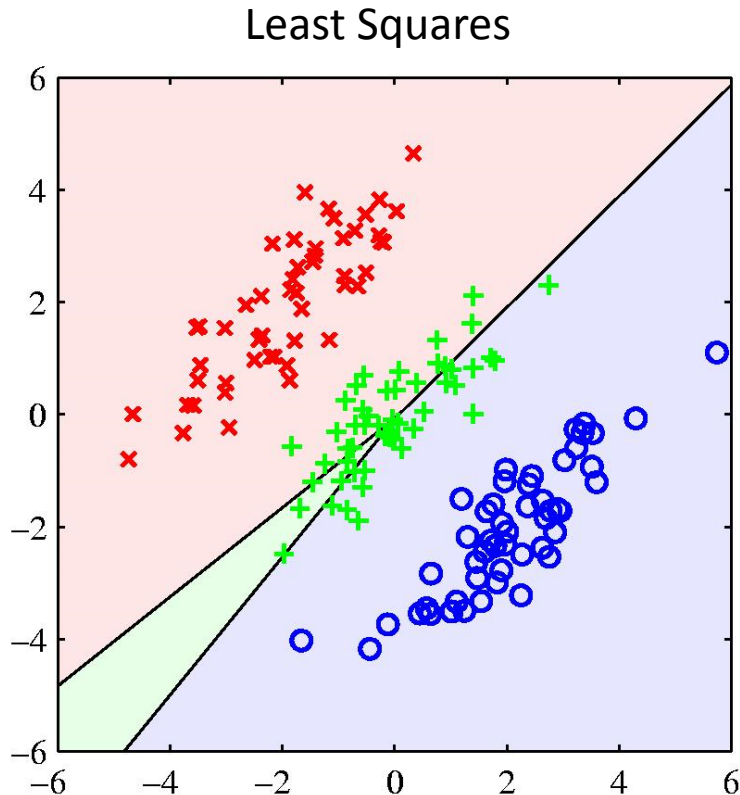


Least squares solutions lack robustness to outliers



If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being “too correct”

For non-Gaussian targets, least squares regression gives poor decision surfaces



Remember that least squares corresponds to the Maximum Likelihood under a Gaussian conditional distribution

Clearly the binary target vectors have a distribution that is far from Gaussian

Fisher's Linear Discriminant

- We can view classification in terms of dimensionality reduction
- A simple linear discriminant function is a projection of the D – dimensional data \mathbf{x} down to 1 dimension

Project: $y = \mathbf{w}^T \mathbf{x}$;

Classify: if $y \geq -w_0$ then C_1 else C_2

- However projection results in loss of information. Classes well separated in the original input space may strongly overlap in 1d
- We will adjust the projection (weight vector \mathbf{w}) to achieve the best separation among classes. But what do we mean by *best separation*?

Fisher's View of Class Separation (I)

- The simplest measure of class separation when projected onto \mathbf{w} is the separation of the projected class means. This suggests choosing \mathbf{w} so to minimize

$$m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1), m_k = \mathbf{w}^T \mathbf{m}_k, \mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathbf{x}_n, \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathbf{x}_n$$

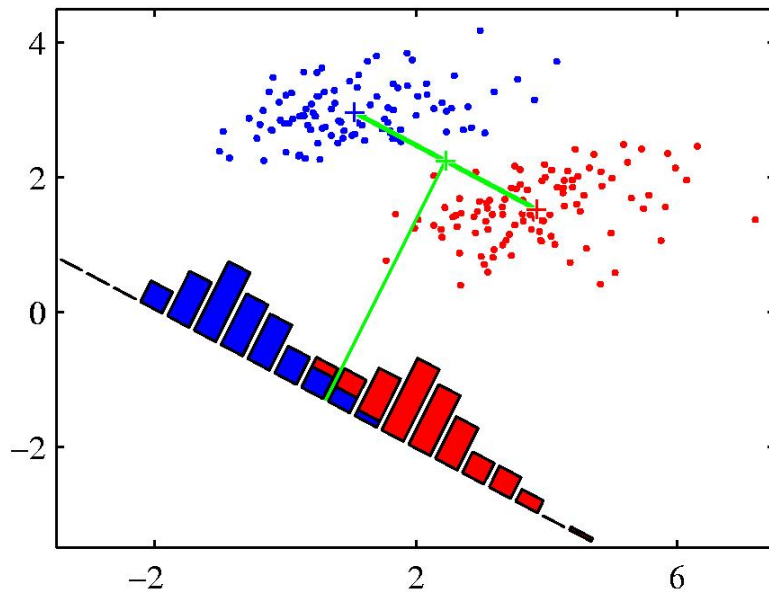
- This can be made arbitrarily large by increasing $\|\mathbf{w}\|$. We could handle this by imposing unit norm constraints using Lagrange multipliers. We get

$$\max_{\mathbf{w}} \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1), \text{ s. th. } \|\mathbf{w}\| = 1$$

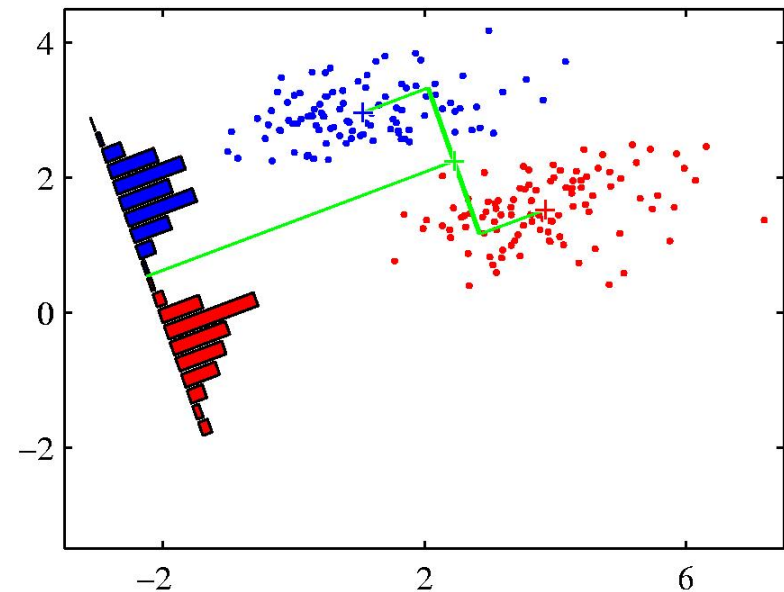
$$\Rightarrow \mathbf{w} = -\frac{1}{2\lambda}(\mathbf{m}_2 - \mathbf{m}_1) \propto \mathbf{m}_2 - \mathbf{m}_1$$

- However, still, if the main direction of variance in each class is not orthogonal to the direction between means, we will not get good separation (see next slide)

Advantage of using Fisher's Criterion



When projected onto the line joining the class means, the classes are not well separated



Fisher chooses a direction that makes the projected classes much tighter, even though their projected means are less far apart

Fisher's View of Class Separation (II)

- **Fisher:** maximize a function that gives a large separation between the projected class means, while also giving a small variance within each class, thereby minimizing class overlap
 - Choose direction maximizing the **ratio** of **between class** variance to **within class** variance
 - This is the direction in which the projected points contain the most information about class membership (under Gaussian assumptions)

Fisher's Linear Discriminant

- We seek a linear transformation that is best for discrimination

$$y = \mathbf{w}^T \mathbf{x}$$

- The projection onto the vector separating the class means seems right

$$\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$$

- But we also want small variance within each class

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2, \quad m_k = \mathbf{w}^T \mathbf{m}_k$$

- Fisher's objective function

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

← Between class

← Within class

Fisher's Linear Discriminant Derivations

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

where

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1) (\mathbf{m}_2 - \mathbf{m}_1)^T$$

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1) (\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2) (\mathbf{x}_n - \mathbf{m}_2)^T$$

$$\frac{d J(\mathbf{w})}{d \mathbf{w}} = 0 \Rightarrow (\overset{\text{scalar}}{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}) \mathbf{S}_W \mathbf{w} = (\overset{\text{scalar}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}) \underline{\mathbf{S}_B \mathbf{w}} \mid (\text{lx}) \mathbf{S}_W^{-1}$$

$$\text{Optimal solution: } \mathbf{w} \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1) \propto (\mathbf{m}_2 - \mathbf{m}_1)$$

The above result is known as Fischer's linear discriminant. Strictly *it is not a discriminant*, but rather a direction of projection that can be used for classification in conjunction with a decision (e.g. thresholding) operation.

Fischer's Linear Discriminant Computation

However, the objective $J(\mathbf{w})$ is invariant to rescaling $\mathbf{w} \rightarrow \alpha \mathbf{w}$. We can chose the denominator to be unity. We can then minimize

$$\min_{\mathbf{w}} -\frac{1}{2} \mathbf{w}^T \mathbf{S}_B \mathbf{w}$$

$$\mathbf{w}^T \mathbf{S}_W \mathbf{w} = 1$$

This corresponds to the (primal) Lagrangian

$$L_P = -\frac{1}{2} \mathbf{w}^T \mathbf{S}_B \mathbf{w} + \frac{1}{2} \lambda (\mathbf{w}^T \mathbf{S}_W \mathbf{w} - 1)$$

From the *KKT conditions*

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w} \Rightarrow \mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w} = \lambda \mathbf{w}$$

Generalized eigenvalue problem, as $\mathbf{S}_W^{-1} \mathbf{S}_B$ not symmetric

Fischer's Linear Discriminant Computation

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w} = \lambda \mathbf{w}$$

- Given that \mathbf{S}_B is symmetric positive definite, we can write

$$\mathbf{S}_B = \mathbf{S}_B^{1/2} \mathbf{S}_B^{1/2}$$

$$\text{where } \mathbf{S}_B = \mathbf{U} \mathbf{\Sigma} \mathbf{U}^T, \mathbf{S}_B^{1/2} = \mathbf{U} \mathbf{\Sigma}^{1/2} \mathbf{U}^T$$

- Defining $\mathbf{v} = \mathbf{S}_B^{1/2} \mathbf{w}$, we get

$$\mathbf{S}_B^{1/2} \mathbf{S}_W^{-1} \mathbf{S}_B^{1/2} \mathbf{v} = \lambda \mathbf{v}$$

- We have to solve a regular eigenvalue problem for a symmetric, positive definite matrix $\mathbf{S}_B^{1/2} \mathbf{S}_W^{-1} \mathbf{S}_B^{1/2}$

– We can find solutions λ_k and \mathbf{v}_k corresponding to $\mathbf{S}_B^{1/2} \mathbf{w}$

- Which eigenvector and eigenvalue should we choose? The largest! Why?*
- Transforming to dual

$$\mathbf{w}^T \mathbf{S}_B \mathbf{w} = 1, \mathbf{w}^T \mathbf{S}_W \mathbf{w} = \frac{1}{\lambda_k} \Rightarrow L_D = \text{const.} + \frac{1}{2} \lambda_k$$

(need to maximize over λ)

The Perceptron Model (cca. 1962)

- Linear discriminant model
- Input vector \mathbf{x} is first mapped using a fixed non-linear transformation, to give a feature vector $\phi(\mathbf{x})$, then used to construct linear model

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

where

$$f(a) = +1, a \geq 0$$

$$f(a) = -1, a < 0$$

Typically use $t = +1$ for class C_1 , $t = -1$ for C_2

Feature vector includes a bias component $\phi_0(\mathbf{x}) = 1$

Perceptron Criteria (I)

- Perceptron's algorithm can be motivated by error function minimization
- A natural error would be the number of misclassified patterns
- However this does not lead to a simple learning algorithm, because the error is a piecewise function of \mathbf{w}
- Discontinuities whenever a change in \mathbf{w} causes the decision boundary to move across one of the datapoints
- Gradient methods cannot be immediately applied, as the gradient is zero almost everywhere

Perceptron Criteria (II)

- Patterns \mathbf{x}_n in class C_1 will have $\mathbf{w}^T \phi(\mathbf{x}_n) > 0$
- Patterns \mathbf{x}_n in class C_2 will have $\mathbf{w}^T \phi(\mathbf{x}_n) < 0$
- Target coding $t \in \{-1, 1\}$
- Hence we would like all patterns to satisfy

$$\mathbf{w}^T \phi(\mathbf{x}_n) t_n > 0$$

- The perceptron associates 0 error to correctly classified patterns \mathbf{x}_n , whereas for a misclassified pattern \mathbf{x}_n , it tries to minimize the quantity $-\mathbf{w}^T \phi(\mathbf{x}_n) t_n$

Perceptron Criteria (III)

- The perceptron criterion is given by

$$E_P(\mathbf{w}) = - \sum_{n \in M} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

where M is the set of misclassified examples

- By applying stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

- Since perception's function f is invariant to the rescaling of \mathbf{w} , we can set $\eta = 1$
- As \mathbf{w} changes, so will the set of misclassified patterns

Algorithm

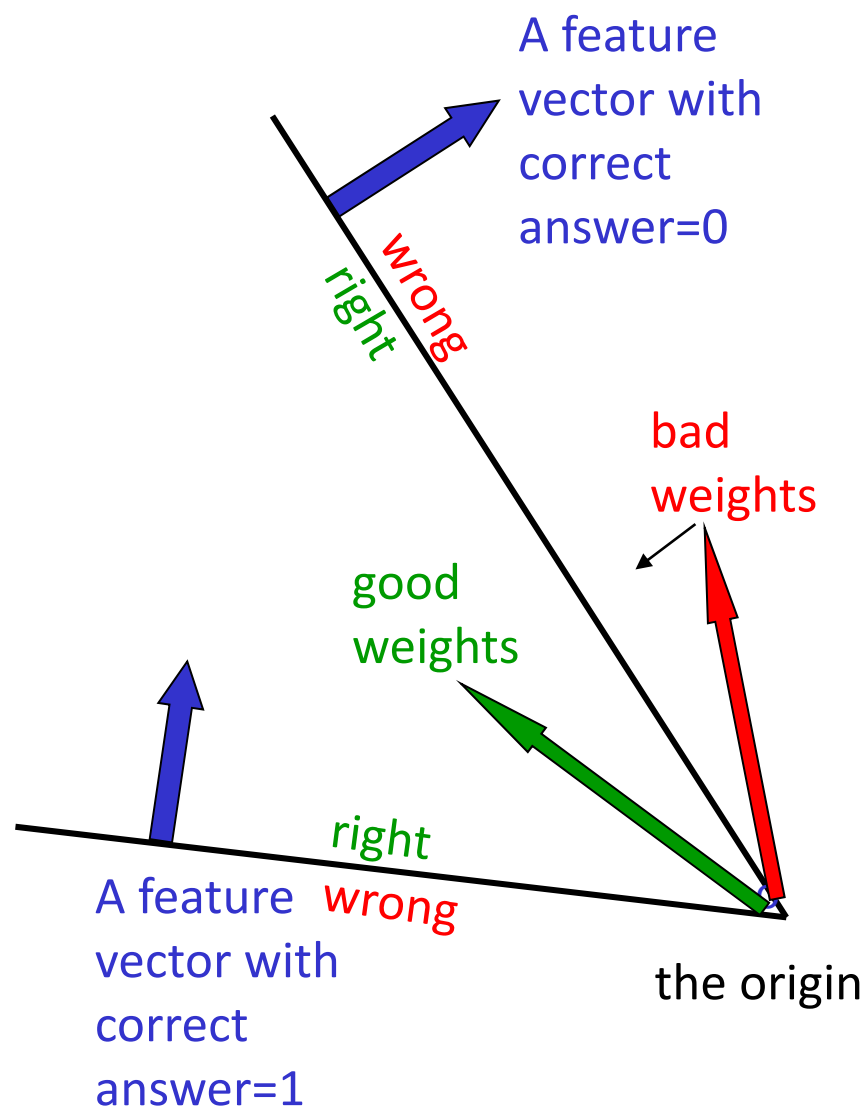
- We cycle through the training patterns in turn
- For each pattern we evaluate the perceptron function (output)

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

- If the pattern is *correctly classified*, then the weight vector \mathbf{w} remains unchanged
- If the pattern is *incorrectly classified*
 - For class C_1 we add vector $\phi(\mathbf{x}_n)$ to the current estimate of the weight vector \mathbf{w}
 - For class C_2 we subtract vector $\phi(\mathbf{x}_n)$ from the current estimate of the weight vector \mathbf{w}

Weight and Data Space

- Imagine a space in which each axis corresponds to a feature value or to the weight on that feature
 - A point in this space is a weight vector. Feature vectors are shown in blue translated away from the origin to reduce clutter.
- Each training case defines a plane.
 - On one side of the plane the output is **wrong**.
- To get all training cases right we need to find a point on the right side of all the planes.
 - This feasible region (if it exists) is a cone with its tip at the origin



Perceptron's Convergence

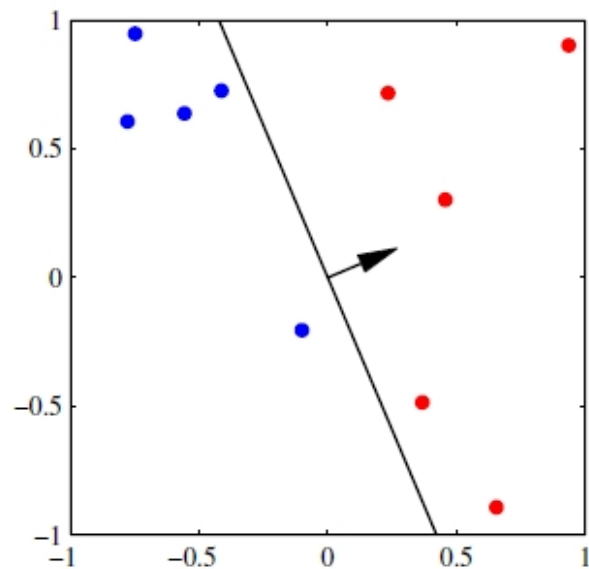
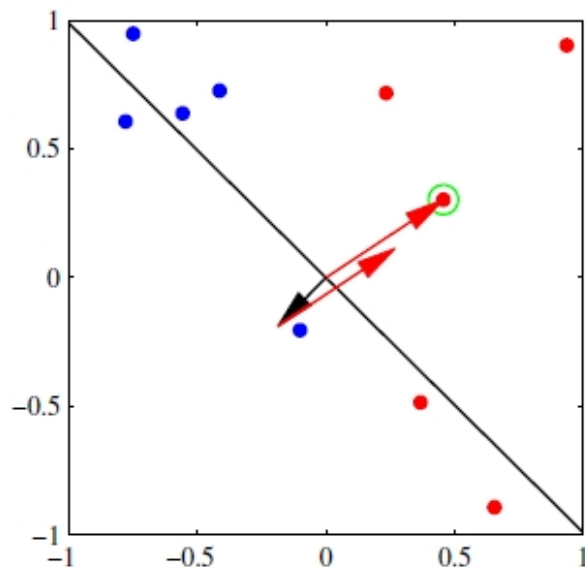
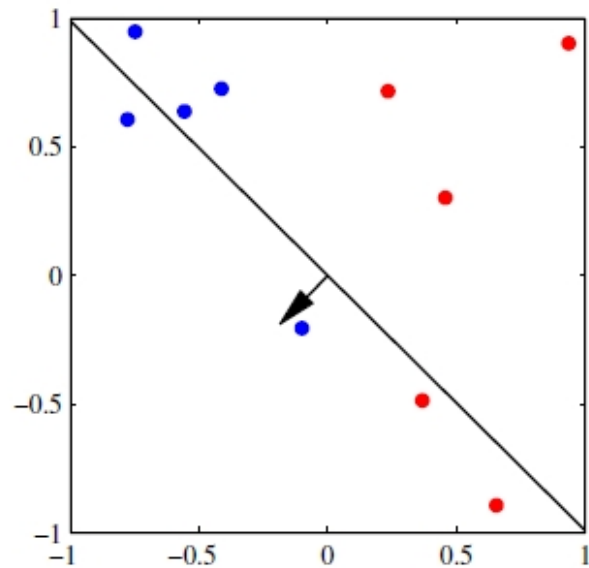
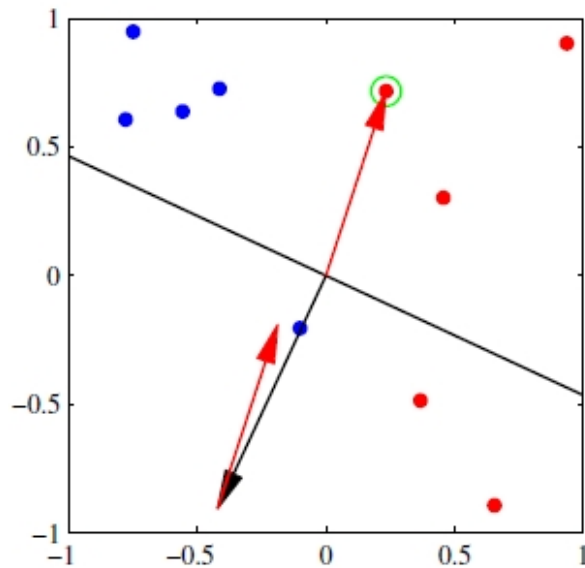
$$\begin{aligned} -\mathbf{w}^{(\tau+1)\text{T}} \phi_n t_n &= -\mathbf{w}^{(\tau)\text{T}} \phi_n t_n - (\phi_n t_n)^{\text{T}} \phi_n t_n \\ &< -\mathbf{w}^{(\tau)\text{T}} \phi_n t_n \end{aligned}$$

- Contribution to error function from a misclassified pattern \mathbf{x}_n is reduced
- However, this does not imply that contributions from other misclassified patterns will have been reduced as the change in weight vector may have caused some previously correctly classified patterns to become misclassified
- The perceptron rule is not guaranteed to reduce the total error function at each stage
- Novikoff (1962) proved that the perceptron algorithm converges after a finite number of iterations, **if the data set is linearly separable**
- The weight vector is always adjusted by a bounded amount in a direction it has a negative dot product with, and thus can be *bounded above* by $O(\sqrt{I})$ where I is the number of changes to \mathbf{w} . But it can also be *bounded below* by $O(I)$ because if there exists an (unknown) feasible \mathbf{w} , then every change makes progress in this direction by a positive amount that depends only on the input vector. This can be used to show that the number of updates I to the weight vector is bounded by $\frac{2R}{\gamma^2}$, where R is the maximum norm of an input vector.

Summary: Perceptron's Convergence

- **Perceptron's convergence theorem**: if there exists an exact solution (data is linearly separable), then the perceptron algorithm is guaranteed to find an exact solution in a finite number of steps
- The number of steps could be very large, though
- Until convergence we cannot distinguish between a non-separable problem, or one that is just slow to converge
- Even for linearly separable data, there may be many solutions, depending on the parameter initialization and the order in which datapoints are presented

Perceptron at Work



Other Issues with the Perceptron

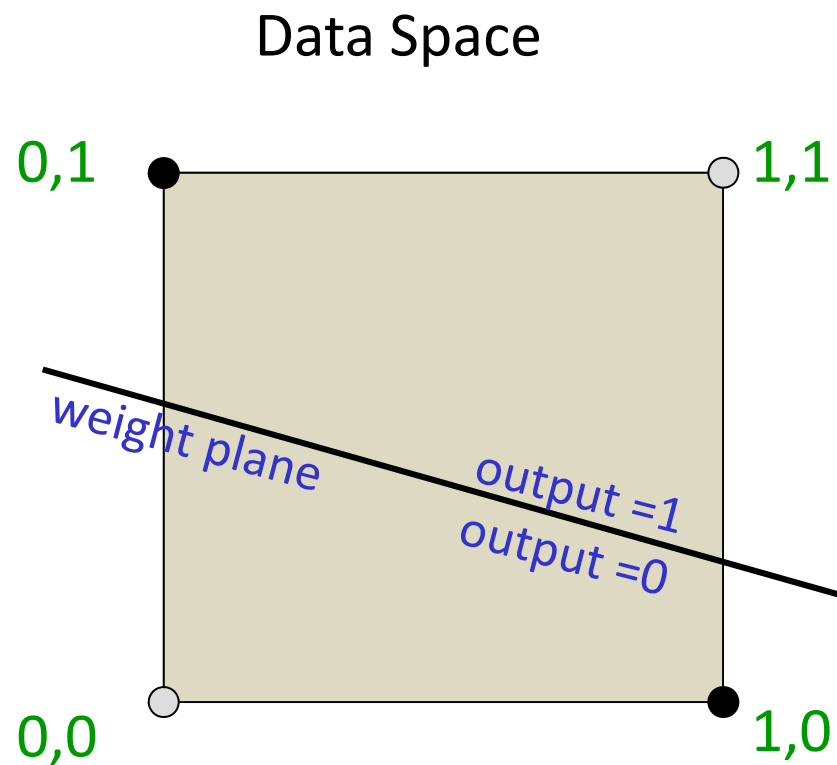
- Does not provide probabilistic outputs
- Does not generalize readily to more than 2 classes
- Is based on linear combinations of fixed basis functions

What Perceptrons Cannot Learn

- The adaptive part of a perceptron cannot even tell if two single bit features have the same value!
Same: $(1,1) \rightarrow 1$; $(0,0) \rightarrow 1$
Different: $(1,0) \rightarrow 0$; $(0,1) \rightarrow 0$
- The four feature-output pairs give four inequalities that are impossible to satisfy:

$$w_1 + w_2 \geq \theta, \quad 0 \geq \theta$$

$$w_1 < \theta, \quad w_2 < \theta$$



The positive and negative cases cannot be separated by a plane

The Logistic Sigmoid (due to S-shape)

- This is also called a squashing function because it maps the entire real axis into a finite interval

- For classification, the output a is a smooth function of the inputs and the weights $\mathbf{w} \longrightarrow a = \mathbf{w}^T \mathbf{x} + w_0$

- Properties

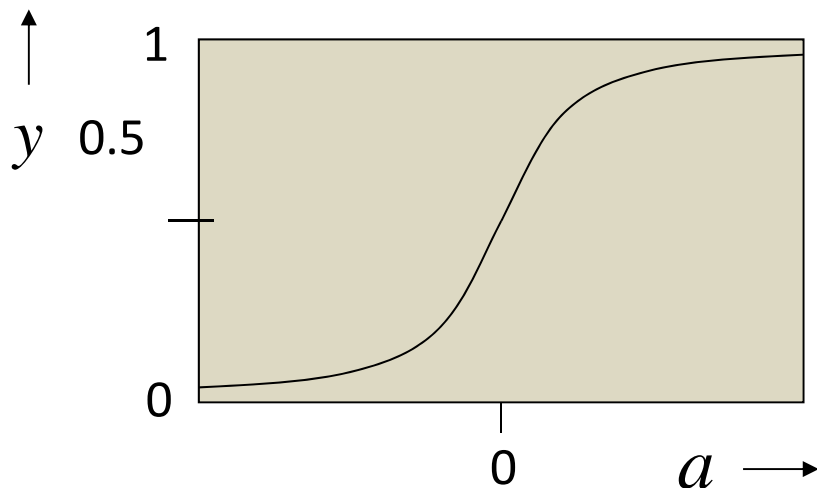
$$\sigma(-a) = 1 - \sigma(a), \quad a = \ln\left(\frac{\sigma}{1 - \sigma}\right)$$

logit function

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

$$\frac{\partial a}{\partial w_i} = x_i \quad \frac{\partial a}{\partial x_i} = w_i$$

$$\frac{dy}{da} = y(1 - y)$$



Probabilistic Generative Models

- Use a class prior and a separate generative model of the input vectors for each class, and compute which model makes a test input vector most probable
- The posterior probability of class 1 is given by:

$$p(C_1 | \mathbf{x}) = \frac{p(C_1)p(\mathbf{x} | C_1)}{p(C_1)p(\mathbf{x} | C_1) + p(C_2)p(\mathbf{x} | C_2)} = \frac{1}{1 + e^{-a}}$$

Logistic
sigmoid

$$\text{where } a = \ln \frac{p(C_1)p(\mathbf{x} | C_1)}{p(C_2)p(\mathbf{x} | C_2)} = \ln \frac{p(C_1 | \mathbf{x})}{1 - p(C_1 | \mathbf{x})}$$

↑
z is called the logit and is given
by the log odds

Multiclass Model (Softmax)

$$\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

where $a_k = \ln p(\mathbf{x}|C_k)p(C_k)$

- This is known as the *normalized exponential*
- Can be viewed as a multiclass generalization of the logistic sigmoid
- It is also called a *softmax function* (it is a smoothed version of 'max')

if $a_k \gg a_j \forall j \neq k$, then $p(C_k|x) \simeq 1$ and $p(C_j|x) \simeq 0$

Gaussian Class-Conditionals

- Assume that the input vectors for each class are from a Gaussian distribution, and all classes have the same covariance matrix. The class conditionals are

$$p(\mathbf{x} | C_k) = \underset{\text{normalizer}}{1/Z} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \underset{\text{inverse covariance matrix}}{\Sigma^{-1}} (\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

- For two classes, C_1 and C_2 , the posterior turns out to be a logistic

$$p(C_1 | \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$$

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

Quadratic terms canceled
due to common covariance

$$w_0 = -\frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)}$$

Interpretation of Decision Boundaries

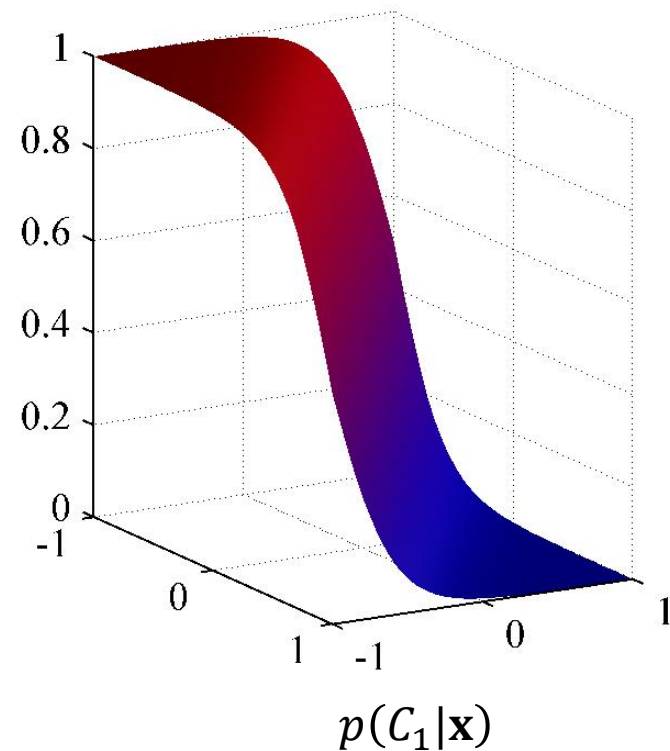
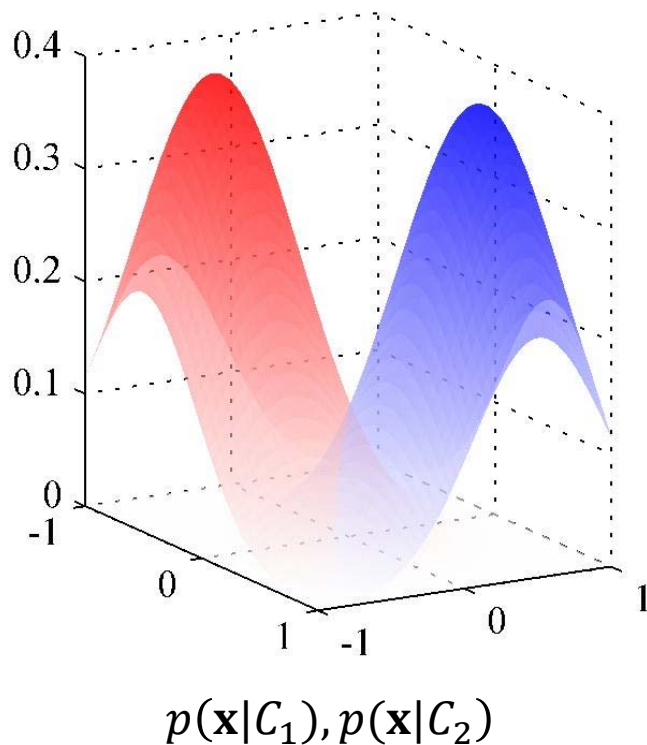
$$p(C_1 | \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$$

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)}$$

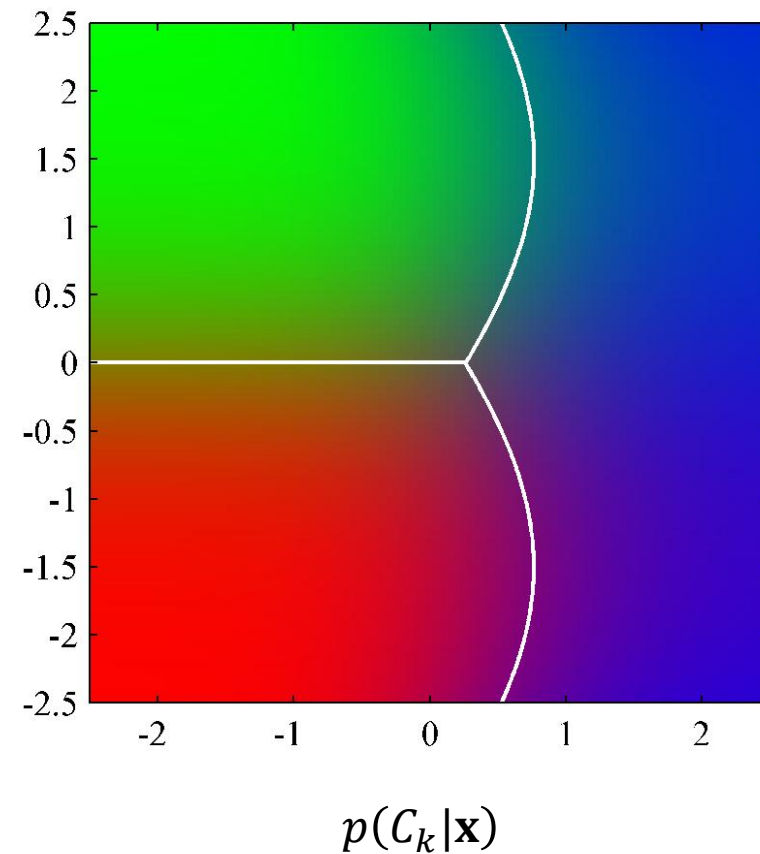
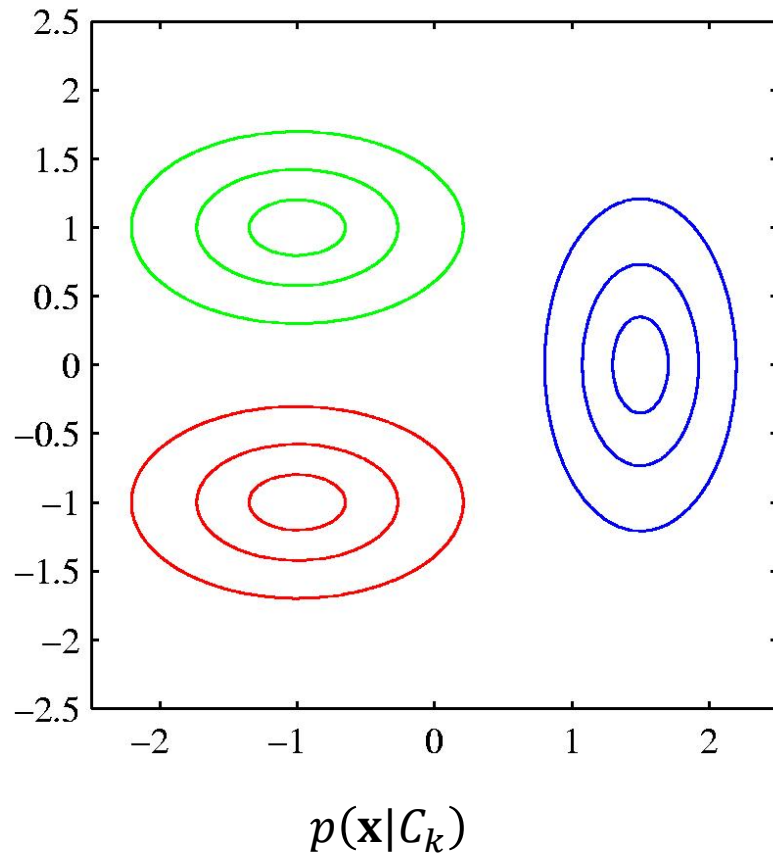
- Quadratic terms canceled due to common covariance
- The sigmoid takes a linear function of \mathbf{x} as argument
- The decision boundaries correspond to surfaces along which the posteriors $p(C_k | \mathbf{x})$ are constant, so they will be given by linear functions of \mathbf{x} . Thus, decision boundaries are linear functions in input space
- The prior probabilities $p(C_k)$ enter only through the bias parameter w_0 , so changes in priors have the effect of making parallel shifts of the decision boundary (more generally of the parallel contours of constant posterior probability)

A picture of the two Gaussian models and the resulting posterior for the red class



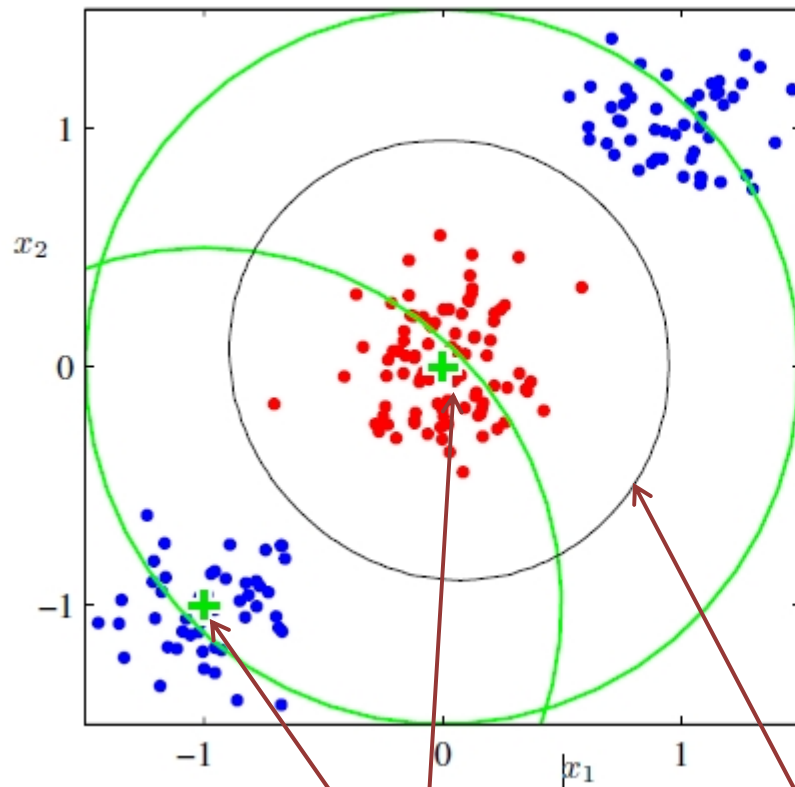
The logistic sigmoid in the right-hand plot is coloured using a proportion of red tone given by $p(C_1|\mathbf{x})$ and a proportion of blue tone given by $p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$.

Class posteriors when covariance matrices are different for different classes

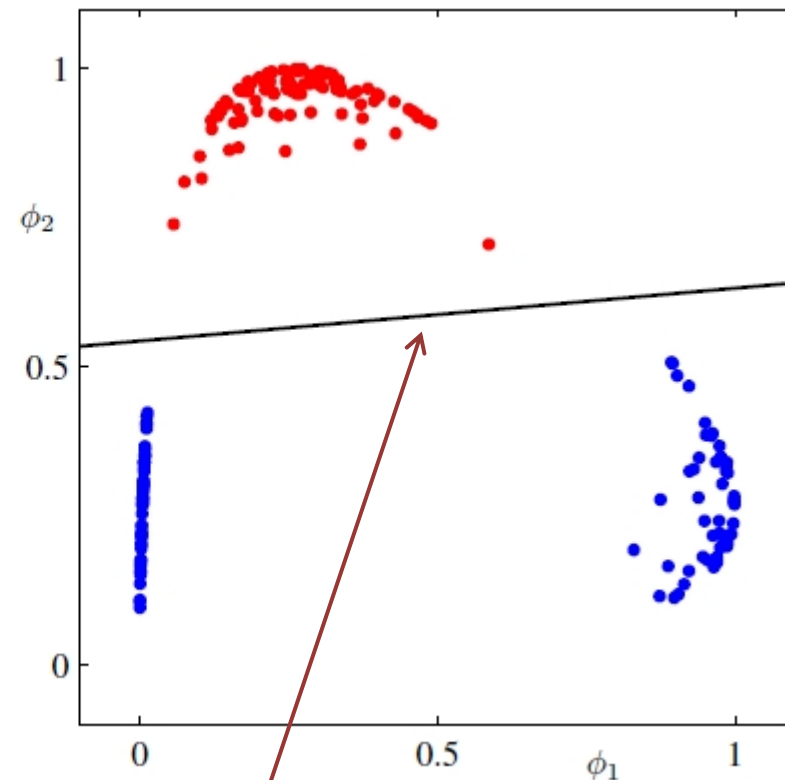


The decision surface is planar when the covariance matrices are the same and quadratic when they are not

Effect of using Basis Functions



Centers of Gaussian basis functions $\phi_1(x)$ and $\phi_2(x)$ with green iso-contours



Linear decision boundary (logistic regression) in feature space
Decision boundary induced in input space

Probabilistic Discriminative Models

Logistic Regression

- In our discussion of generative approaches, we saw that under general assumptions, the class posterior for C_1 can be written as a logistic sigmoid acting on a linear function of the feature vector ϕ
- In logistic regression, we use the functional form of the generalized linear model explicitly

$$p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi), \text{ where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$

- Fewer adaptive parameters compared to the generative model
- For M -dimensional feature space
 - Discriminative: M parameters
 - Generative: $2M$ parameters for the means + $\frac{M(M+1)}{2}$ parameters for (shared!) covariance $\Rightarrow \frac{M(M+5)}{2} + 1$ total parameters
 - Quadratic versus linear number of parameters!

Logistic Regression

Chain Rule for Error Derivatives

$$E = -\sum_{n=1}^N \ln p(t_n | y_n) = -\sum_{n=1}^N t_n \ln y_n + (1-t_n) \ln (1-y_n)$$

$$\frac{\partial E_n}{\partial y_n} = -\frac{t_n}{y_n} + \frac{1-t_n}{1-y_n} = \frac{y_n - t_n}{y_n (1-y_n)}$$

$$a_n = \mathbf{w}^T \phi_n + w_0, \quad \frac{\partial a_n}{\partial \mathbf{w}} = \phi_n$$

$$\frac{\partial E_n}{\partial y_n} = \frac{y_n - t_n}{y_n (1-y_n)}, \quad \frac{dy_n}{da_n} = y_n (1-y_n)$$

$$\frac{\partial E_n}{\partial \mathbf{w}} = \frac{\partial E_n}{\partial y_n} \frac{dy_n}{da_n} \frac{\partial a_n}{\partial \mathbf{w}} = (y_n - t_n) \phi_n$$

Maximum Likelihood for *Logistic Regression*

$$p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi); \sigma(a) = \frac{1}{1 + \exp(-a)}, \frac{d\sigma}{da} = \sigma(1 - \sigma)$$

For dataset $\{\phi_n \equiv \phi(\mathbf{x}_n), t_n\}$, with $t_n \in \{0,1\}, n = 1, \dots, N$,

$$\mathbf{t} = (t_1, t_2, \dots, t_N)^T, \quad y_n = p(C_1|\phi_n) = \sigma(a_n), a_n = \mathbf{w}^T \phi_n$$

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

Cross-entropy error

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n$$

Similar form as the gradient
of the sum-of-squares
regression model

Iterative Reweighted Least Squares

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n$$

- The Newton-Raphson update

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \mathbf{H}^{-1} \nabla E(\mathbf{w})$$

- Logistic model (Φ is the $N \times M$ design matrix with n -th row ϕ_n^T)

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n = \Phi^T (\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n \phi_n^T = \Phi^T \mathbf{R} \Phi$$

where $R_{nn} = y_n(1 - y_n)$, $0 < y_n < 1$; Then $\mathbf{u}^T \mathbf{H} \mathbf{u} > 0, \forall \mathbf{u}$

- It follows that

Normal equations with non-constant weighting matrix \mathbf{R}

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) = (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z}$$

$$\text{where } \mathbf{z} = \Phi \mathbf{w}^{(i)} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})$$

Facts on IRLS

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - (\boldsymbol{\Phi}^T \mathbf{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T (\mathbf{y} - \mathbf{t}) = (\boldsymbol{\Phi}^T \mathbf{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{R} \mathbf{z}, \quad \mathbf{z} = \boldsymbol{\Phi} \mathbf{w}^{(i)} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})$$

- The weighting matrix \mathbf{R} is not constant, but the Hessian is positive definite
- This means that we have to iterate to find the solution, but the likelihood function is concave in \mathbf{w} . We have a unique optimum
- The n -th component of \mathbf{z} can be interpreted as an effective target value obtained by making a local linear approximation to the logistic sigmoid around the current operating point $\mathbf{w}^{(i)}$
- The elements of the diagonal weighting matrix \mathbf{R} , $y_n(1 - y_n)$ can be interpreted as variances
- We can interpret IRLS as the solution to a linearized problem in the space of the variable $a = \mathbf{w}^T \boldsymbol{\phi}$ (the sigmoid argument)

Readings

Bishop

Ch. 4, up to 4.3.4