# Linear Models for Classification

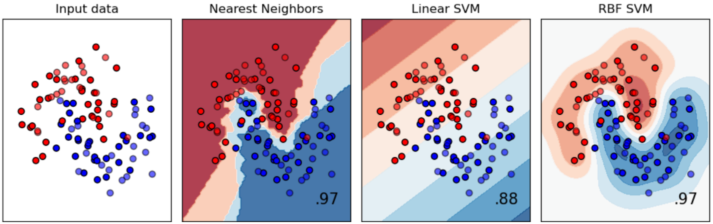
## Chapter 4

Prof. Reza Azadeh

University of Massachusetts Lowell

## Introduction

- Goal of classification: take an input vector $\mathbf{x}$ and assign it to one of $K$ discrete classes $\mathcal{C}_k$ where $k = 1, \ldots, K$.
- *decision regions* vs. *decision boundaries* (or *decision surfaces*).



## Assumptions

- Classes are disjoint so each input is assigned to one and only one class
- We consider linear models for classification, by which we mean that the decision surfaces are linear functions of the input vector $\mathbf{x}$ and hence are defined by $(D-1)$-dimensional hyperplanes within the $D$-dimensional input space.

## Linear Models for Classification

- Target values for probabilistic classification models can be represented as binary values.
- In the case of a two-class problem, we use a single target value $t \in 0, 1$ where $t = 1$ represents $\mathcal{C}_1$ and $t = 0$ represents $\mathcal{C}_2$.
- for $K > 2$ classes, use a 1-of-$K$ coding scheme in which $\mathbf{t}$ is a vector of length $K$. For instance fo $\mathcal{C}_j$ we have $\mathbf{t} = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$ where the $j^{\text{th}}$ element is 1.

## Approaches to the Classification Problem

a. Construct a *discriminant function* that directly assigns each vector $\mathbf{x}$ to a specific class.

b. Model the conditional probability $p(\mathcal{C}_k|\mathbf{x})$ and use it to find an optimal decision, separating inference and decision steps. This can be done in two ways:

   1. model the conditional probability directly (e.g., using a parametric model) and then optimize the parameters using a training set.
   2. Use a generative approach by applying Bayes' theorem

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

## Activation Function

- In regression, the model outputs real numbers, for example

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

- In classification, we want values in range $(0, 1)$. We achieve this by using a nonlinear function known as *activation function*

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

## Linear Discriminant Functions

- assign inputs directly to classes
- decision surfaces are hyperplanes, hence linear

## Linear Discriminant Functions - Two classes

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector so that

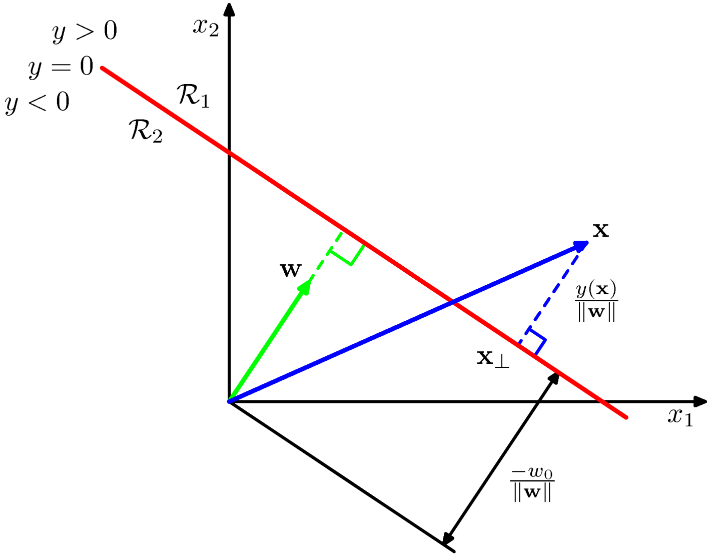
$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

where $\mathbf{w}$ and $w_0$ are the *weight vector* and the *bias*. For an input $\mathbf{x}$,

- If $y(\mathbf{x}) \geq 0$, then $\mathbf{x}$ belongs to $\mathcal{C}_1$,
- otherwise $\mathbf{x}$ belongs to $\mathcal{C}_2$

The decision boundary is defined by $y(\mathbf{x}) = 0$.

## Linear Discriminant Functions - Two classes

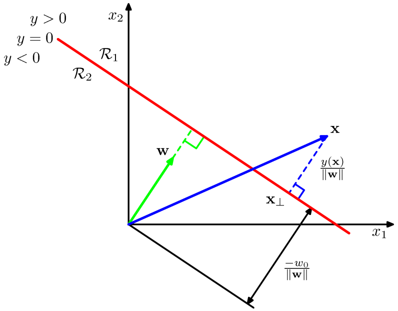


## Linear Discriminant Functions - Two classes

- $\mathbf{w}$ is orthogonal to the decision surface and determines its orientation.
- $w_0$ determines the location of the decision surface.

$$\frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

- we make use parameters so $\mathbf{w} = (w_0, \mathbf{w})$ and $\mathbf{x} = (x_0, \mathbf{x})$, so $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$.



## Linear Discriminant Functions - Multiple classes

For $K > 2$, consider a single $K$-class discriminant comprising $K$ linear function of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

And then assign point $\mathbf{x}$ to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. The decision boundary between classes $\mathcal{C}_k$ and $\mathcal{C}_j$ is given by $y_k(\mathbf{X}) = y_j(\mathbf{X})$ corresponding to a $(D-1)$-dimensional hyperplane defined by

$$(\mathbf{w}_k - \mathbf{w}_j)^\top \mathbf{x} + (w_{k0} - w_{j0}) = 0.$$

## Least Squares for Classification (1)

Consider $K$ classes with a 1-of-$K$ binary coding for the target values. Each class $\mathcal{C}_k$ is described by its own linear model so that

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

where $k = 1, \ldots, K$. Group these together as

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^\top \tilde{\mathbf{x}}$$

where the $k^{\text{th}}$ column of $\tilde{\mathbf{W}}$ is the $D + 1$-dimensional vector $\tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^\top)^\top$, and $\tilde{\mathbf{x}}$ is the augmented input vector $(1, \mathbf{x}^\top)^\top$.

## Least Squares for Classification (2)

A new value $\mathbf{x}$ is assigned to the class for which the output $y_k = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}$ is largest.
Consider a training data set $\{\mathbf{x}_n, \mathbf{t}_n\}$ where $n = 1, \ldots, N$ and define a matrix $\mathbf{T}$ whose $n^{\text{th}}$ row is the vector $\mathbf{t}_n^\top$ together with a matrix $\tilde{\mathbf{X}}$ whose $n^{\text{th}}$ row is $\tilde{\mathbf{x}}_n^\top$. Minimize a sum-of-squares error function to find $\tilde{\mathbf{W}}$,

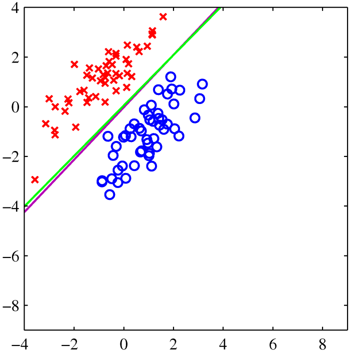
$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\{(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^\top (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})\}.$$

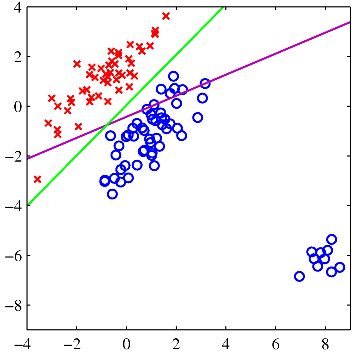
Setting the derivatives with respect to $\tilde{\mathbf{W}}$ to zero, we get

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \mathbf{T} = \tilde{\mathbf{X}}^\dagger \mathbf{T}$$

## Least Squares for Classification (3)

- Least-squares approach gives an exact closed-form solution.
- Similar to least-squares in regression, it lacks robustness to outliers.





# Least Squares for Classification (4)

A more severe issue is that by definition it corresponds to maximum likelihood under the assumption of a Gaussian conditional distribution, whereas binary target vectors clearly have a distribution that is far from Gaussian.

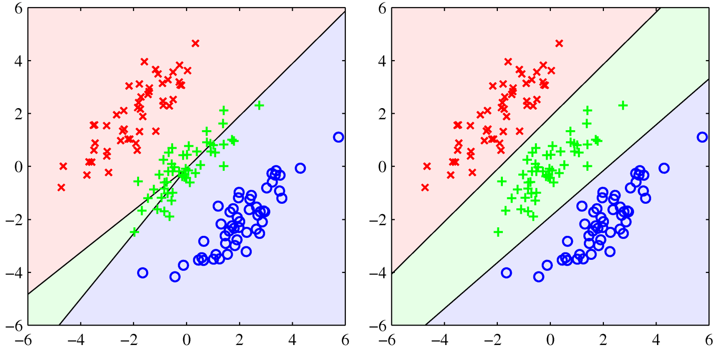


Figure: (left) least-squares, (right) logistic regression

## Fisher's Linear Discriminant (1)

Idea: To improve class separation, project the data set into a lower dimension before classification

- select a projection that maximizes class separation to avoid overlapping
- minimize variance within each class

# Fisher's Linear Discriminant (2)

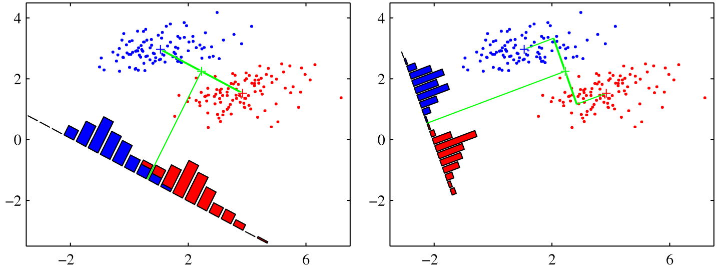


Figure: (left) projection by considering only the class means, (right) using Fisher's discriminant

## Fisher's Linear Discriminant (3)

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1),$$

where $\mathbf{m}_1$ and $\mathbf{m}_2$ are mean vectors of classes

$$\mathbf{m_i} = \frac{1}{N_i} \sum_{n \in \mathcal{C}_i} \mathbf{x}_n \quad i = 1, 2$$

and $\mathbf{S}_W$ is within-class covariance defined as

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^\top + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^\top$$

## The Perceptron Algorithm (1)

- Another example of a linear discriminant by Rosenblatt (1962)
- use a fixed nonlinear transformation to give a feature vector $\boldsymbol{\phi}(\mathbf{x})$ and build a generalized linear model

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

- the nonlinear activation function is given by

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- instead of using a one-hot coding for perceptron we use $t = +1$ for $\mathcal{C}_1$ and $t = -1$ for $\mathcal{C}_2$.

## The Perceptron Algorithm (2)

- To find $\mathbf{w}$, the perceptron algorithm considers $\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) > 0$ if $\mathbf{x}_n \in \mathcal{C}_1$ and $\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) < 0$ if $\mathbf{x}_n \in \mathcal{C}_2$.
- Since we selected $t = \{-1, 1\}$ then we want, $\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) t_n > 0$
- *Perceptron criterion*: minimizes the quantity $-\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) t_n$ for misclassified data points:

$$E_p(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) t_n$$

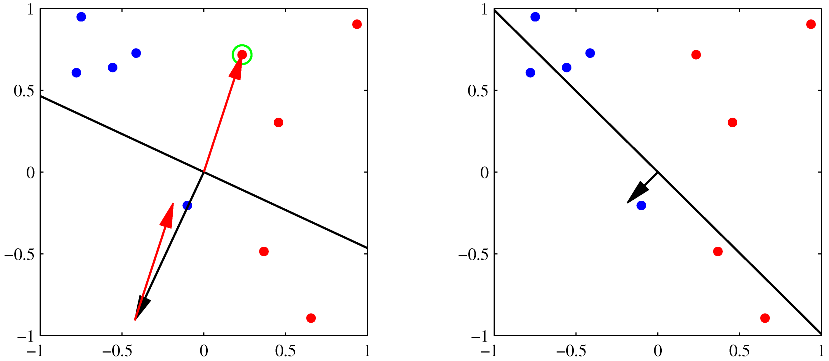
## The Perceptron Algorithm (3)

- optimize using stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_p(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \boldsymbol{\phi}_n t_n$$

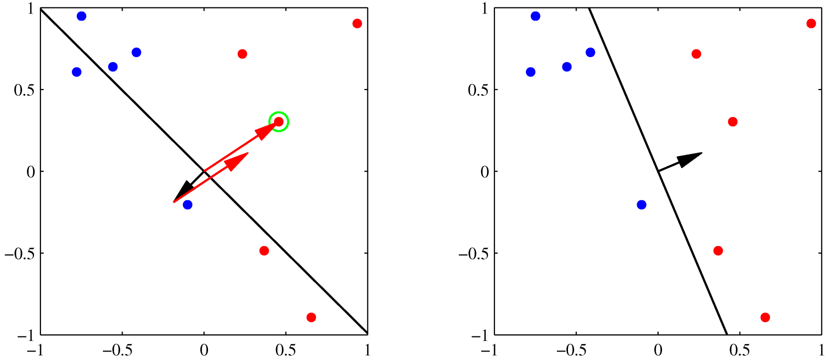
  where $\eta$ is the learning rate and $\tau$ indexes time steps.
- It has been shown that if there exists an exact solution (i.e., the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.

## The Perceptron Algorithm (4)



(left) initial **w** as a black arrow towards the red class. missed classified data point shown with a green circle and its feature vector is added to the current **w** resulting in the new boundary (right)

## The Perceptron Algorithm (5)



(left) next misclassified point shown with a green circle and its feature vector is added to the current **w** resulting in the new boundary (right)

## The Perceptron Algorithm (6)

Limitations of the perceptron algorithm:

- difficulties with the learning algorithm
- does not provide probabilistic output
- does not generalize readily to $K > 2$ classes
- relies on linear combination of fixed basis functions.

Can we do better? Yes, probabilistic method!

## Probabilistic Generative Models

Consider a two class problem. The posterior probability for class $\mathcal{C}_1$ can be written as

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

If we define

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

Then the posterior can be written as

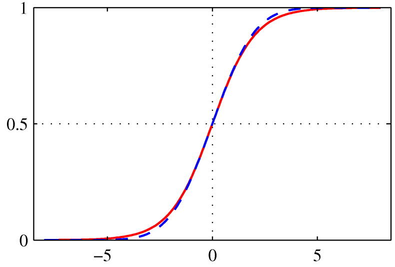
$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

## Logistic Sigmoid Function

$\sigma(a)$ is the *logistic sigmoid function* with important properties

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

The second property is the inverse of the sigmoid function a.k.a the *logit* function.



## Extension to multiple classes

For $K > 2$ classes, we have

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

which is known as the normalized exponential, a.k.a the *softmax* function, and can be regarded as a multiclass generalization of the logistic sigmoid with

$$a_k = \ln(p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k))$$

## Different forms of class-conditional distribution

We now investigate the consequences of choosing specific forms for the class-conditional densities, $p(\mathbf{x}|\mathcal{C}_k)$, and see the form of the posterior for them.

- continuous (Gaussian)
- discrete

## Continuous inputs (1)

Let us assume Gaussian form

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\}$$

Considering the case of two classes and using the posterior in the form of a sigmoid function, we have

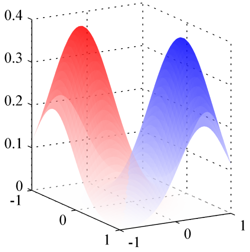
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$

where we have defined

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

## Continuous inputs (2)

Note that the above results have been obtained by assuming all classes share the same covariance matrix. This assumption leads to a **linear function** of $\mathbf{x}$ in the argument of the logistic sigmoid. In other words, the decision boundaries are linear in input space.



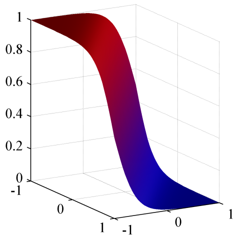


Figure: (left) $p(\mathbf{x}|\mathcal{C}_k)$ for two classes, (right) posterior given by a logistic sigmoid of a linear function of $\mathbf{x}$.

## Continuous inputs (3)

For the case of $K$ classes we have

$$a_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_0$$

where

$$\begin{aligned} \mathbf{w}_k &= \Sigma^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k) \end{aligned}$$

We see that $a_k(\mathbf{x})$ are again linear functions of $\mathbf{x}$ due to the shared covariances. The resulting decision boundaries will occur when two of the posterior probabilities are equal, and so we have a **generalized linear model**.

## Continuous inputs (4)

Now let us relax the the assumption of a shared covariance and allow each class conditional density to have its own covariance matrix $\Sigma_k$. Then we will obtain quadratic functions of $\mathbf{x}$ given rise to *quadratic discriminant*.

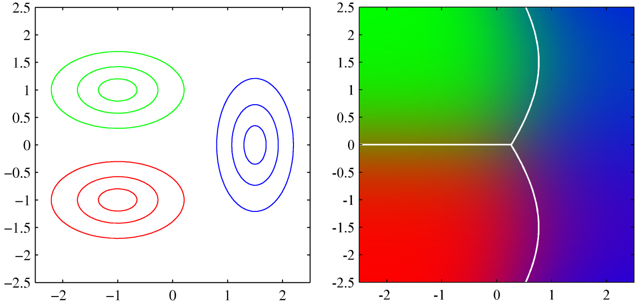


Figure: (left) three class-conditional densities two of which have the same covariance and one has a different one (right) posterior probabilities separated with a linear and two quadratic decision boundaries

## Maximum likelihood solution (1)

We can find the values of the parameters of $p(\mathbf{x}|\mathcal{C}_k)$ together with the prior class probabilities $p(\mathcal{C}_k)$ using maximum likelihood.

Consider first the case of two classes, each having a Gaussian class-conditional density with a shared covariance matrix.

Suppose we have a data set $\{\mathbf{x}_n, \mathbf{t}_n\}$ where $n = 1, \ldots, N$.

Here $t_n = 1$ denotes class $\mathcal{C}_1$ and $t_n = 0$ denotes $\mathcal{C}_2$. We denote the prior class probability $p(\mathcal{C}_1) = \pi$ so that $p(\mathcal{C}_2) = 1 - \pi$.

## Maximum likelihood solution (2)

For a data point $\mathbf{x}_n$ from class $\mathcal{C}_1$, we have $t_n = 1$ and so

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)$$

Similarly for class $\mathcal{C}_2$, we have $t_n = 0$ so

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)$$

Thus the likelihood is given by

$$p(\mathbf{t}, \mathbf{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) = \prod_{n=1}^{N} [\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)]^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^\top$.

## Maximum likelihood solution (3)

Now we should maximize the log likelihood function. First consider maximization w.r.t $\pi$. we get

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N_1 + N_2}$$

where $N_1$ and $N_2$ denote the total number of data points in classes $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

## Maximum likelihood solution (4)

Now let us maximize w.r.t $\boldsymbol{\mu}_1$. For this we get

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n$$

which is simply the mean of all input vectors $\mathbf{x}_n$ assigned to class $\mathcal{C}_1$. Similarly, we get

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

## Maximum likelihood solution (5)

Finally, let us maximize w.r.t shared covariance. Picking the terms in the log likelihood that depend on $\Sigma$, we have

$$-\frac{N}{2}\ln|\Sigma| - \frac{N}{2}\mathrm{Tr}\{\Sigma^{-1}\mathbf{S}\}$$

where we have defined

$$\begin{aligned}
\mathbf{S} &= \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \\
\mathbf{S}_1 &= \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\top \\
\mathbf{S}_2 &= \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\top
\end{aligned}$$

We see that $\Sigma = \mathbf{S}$ which represents a weighted average of the covariance associated with each of the two classes.

## Discrete features (1)

Now consider the case of discrete feature vector $x_i$. For simplicity consider the binary feature values $x_i \in \{0, 1\}$. Assume the feature values are independent, conditioned on class $\mathcal{C}_k$. So, the class-conditional density is of form

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}$$

which contains $D$ independent parameters for each class. Using the generalization of sigmoid functions, we get

$$a_k(\mathbf{x}) = \sum_{i=1}^{D} x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki}) + \ln p(\mathcal{C}_k)$$

which again are **linear functions** of the input values $x_i$.

## Exponential family

For both Gaussian distributed and discrete inputs, the posterior class probabilities are given by **generalized linear models** with logistic sigmoid ($K = 2$ classes) or softmax ($K \geq 2$ classes) activation functions.
This result can be generalized when the class-conditional densities are members of the exponential family of distributions.

## Probabilistic Discriminative Models

- For the two-class classification problem, we have seen that the posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of $\mathbf{x}$.
- For a multi-class classification case, the posterior probability of class $\mathcal{C}_k$ is given by a softmax transformation of a linear function of $\mathbf{x}$.
- We have used maximum likelihood to determine the parameters of the densities as well as the class priors and then used Bayes' theorem to find the posterior class probabilities.
- Alternatively, we can use the functional form of the generalized linear model explicitly and to find its parameters directly using maximum likelihood.

## Fixed basis functions

- So far we have considered classification models that work directly with the original input vector $\mathbf{x}$.
- However, all of the algorithms are equally applicable if we first make a fixed nonlinear transformation of the inputs using a vector of basis functions $\boldsymbol{\phi}(\mathbf{x})$.
- the resulting decision boundaries will be linear in the feature space $\boldsymbol{\phi}$, and these correspond to nonlinear decision boundaries in the original $\mathbf{x}$ space.

# Nonlinear basis functions in linear classification models

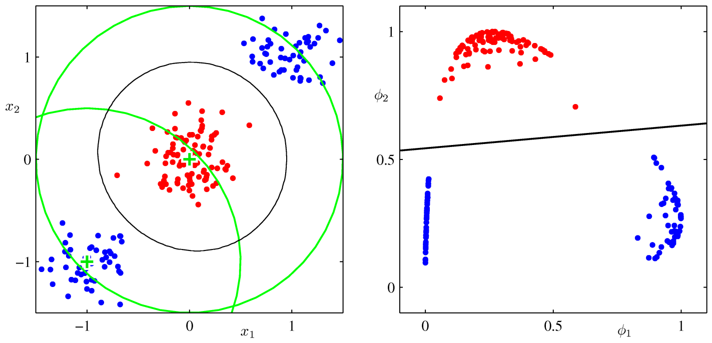


Figure: (left) original input space $(x_1, x_2)$ with data points for two classes. Two Gaussian basis functions are defined shown in green. (right) corresponding feature space $(\phi_1, \phi_2)$ together with the linear decision boundary obtained by a logistic regression model.

## Logistic Regression (1)

Begin with a two-class classification problem. We saw that under rather general assumptions, the posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of the feature vector $\boldsymbol{\phi}$ so that

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^\top \boldsymbol{\phi})$$

with $p(\mathcal{C}_2|\boldsymbol{\phi}) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi})$. This model is known as *logistic regression* and is a model for classification rather than regression.

## Logistic Regression (2)

We now use maximum likelihood to determine the parameters of the logistic regression model. We use

$$\frac{d\sigma}{da} = \sigma(1 - \sigma)$$

For a data set $\{\boldsymbol{\phi}_n, t_n\}$ where $t_n \in \{0, 1\}$ and $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$ with $n = 1, \ldots, N$ the likelihood function is

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

where $\mathbf{t} = (t_1, \ldots, t_n)^\top$ and $y_n = p(\mathcal{C}_1|\boldsymbol{\phi}_n)$.

## Logistic Regression (3)

We define an error function using the negative log likelihood, which gives the *cross-entropy* error function

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

where $y_n = \sigma(a_n)$ and $a_n = \mathbf{w}^\top \boldsymbol{\phi}_n$. The gradient w.r.t $\mathbf{w}$ is

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

- gradient is the error multiplied by the basis function
- the same form as the gradient of the sum-of-squares error function for the linear regression model.

## Iterative reweighted least squares

- In the case of the linear regression model, we used maximum likelihood, on the assumption of Gaussian noise model, to find a closed-form solution.
- For logistic regression, there is no longer a closed-form solution, due to the nonlinearity of the logistic sigmoid function.
- However, the error function is convex and hence has a unique minimum.
- furthermore, the error function can be minimized iteratively based on the Newton-Raphson iterative optimization scheme, which uses a quadratic approximation to the log likelihood function.

## Newton-Raphson update

For minimizing a function $E(\mathbf{w})$, the Newton-Raphson update takes the form

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

where $\mathbf{H}$ is the Hessian matrix.

## Newton-Raphson for linear regression

For the error function $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n)\}^2$, we have

$$\nabla E_D(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^\top \boldsymbol{\phi}_n - t_n)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}\mathbf{w} - \boldsymbol{\Phi}^\top \mathbf{t}$$

$$\mathbf{H} = \nabla\nabla E_D(\mathbf{w}) = \sum_{n=1}^{N} \boldsymbol{\phi}_n \boldsymbol{\phi}_n^\top = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$$

where $\boldsymbol{\Phi}$ is the $N \times M$ design matrix, whose $n^{\text{th}}$ row is given by $\boldsymbol{\phi}_n^\top$. The update is the standard least-squares solution as

$$\mathbf{w}^{(\text{new})} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{t}$$

## Newton-Raphson for logistic regression

For the cross-entropy error function, we have

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n = \mathbf{\Phi}^\top(\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^\top = \mathbf{\Phi}^\top\mathbf{R}\mathbf{\Phi}$$

where $\mathbf{R}$ is an $N \times N$ design matrix with elements $R_{nn} = y_n(1 - y_n)$. Since the Hessian is positive definite, the error function is convex. The Newton-Raphson update is

$$\mathbf{w}^{(\text{new})} = (\mathbf{\Phi}^\top\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{R}\mathbf{z}$$

where $\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$ is an $N$-dimensional vector.

## Multiclass logistic regression (1)

In the discussion about generative models we have seen that for multiclass classification for a large class of distributions, the posterior probabilities are given by a softmax transformation of linear functions of the feature variables so

$$p(\mathcal{C}_k|\boldsymbol{\phi}) = y_k(\boldsymbol{\phi}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where the activation $a_k = \mathbf{w}_k^\top \boldsymbol{\phi}$. We use maximum likelihood to determine to parameters $\{\mathbf{w}_k\}$ of the model directly. We need the derivatives of $y_k$ w.r.t all of the activations $a_j$, as

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j)$$

where $I_{kj}$ are the elements of the identity matrix.

## Multiclass logistic regression (2)

We write he likelihood using the 1-of-$K$ coding scheme as

$$p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{\phi})^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

where $y_{nk} = y_k(\boldsymbol{\phi}_n)$ and $\mathbf{T}$ is an $N \times K$ matrix of target variables with elements $t_{nk}$. The negative log results in

$$E(\mathbf{w}_1,\ldots,\mathbf{w}_k) = -\ln p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$$

which is known as the *cross-entropy* error function for the multiclass classification problem.

## Multiclass logistic regression (3)

Now calculate the gradient of the error function

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_k) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \boldsymbol{\phi}_n$$

And the Hessian matrix is

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_k) = \sum_{n=1}^{N} y_{nk} (I_{kj} - y_{nj}) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^\top$$