Introduction to Linear Classification

by Grady Jensen

Reading Material:
"Pattern Recognition and Machine Learning" by Bishop [ch. 4, 4.1.1, 4.1.2, 4.1.7, 4.2, 4.3.2, 4.3.4]

Further extra reading:
"Machine Learning: A Probabilistic Perspective" by Murphy

Note: these slides are adapted from slides originally by Justin Bayer
Most figures are from C. Bishop: "Pattern Recognition and Machine Learning"

Notation

Symbol	Meaning
s	a scalar number is lowercase and not bold
S	A vector is uppercase
$\mathbf{S}$	a matrix is uppercase and bold
$y(X)$	distance from decision surface
y	predicted class label
z	actual class label
$\mathbb{I}(a = b)$	Indicator function; $\mathbb{I}(a) = 1$ if a else 0
z_n	The actual class label of the n'th example
$X^\dagger$	The Moore-Penrose Pseudoinverse of X

There is not a special symbol for vectors or matrices augmented by the bias term, w_0. Assume it is always included as was done with linear regression.

What is the difference between classification and regression?

Regression:

Calculating an output that consists of one or more continuous variables.

For example, prediction of the yield in a chemical manufacturing process in which inputs consist of the concentrations of reactants, the temperature, and the pressure.

Classification:

Calculating an output that consists of one or more classes or groups.

For example, separating flowers into types given the length and width of the sepals and the pedals.

Classification Problems

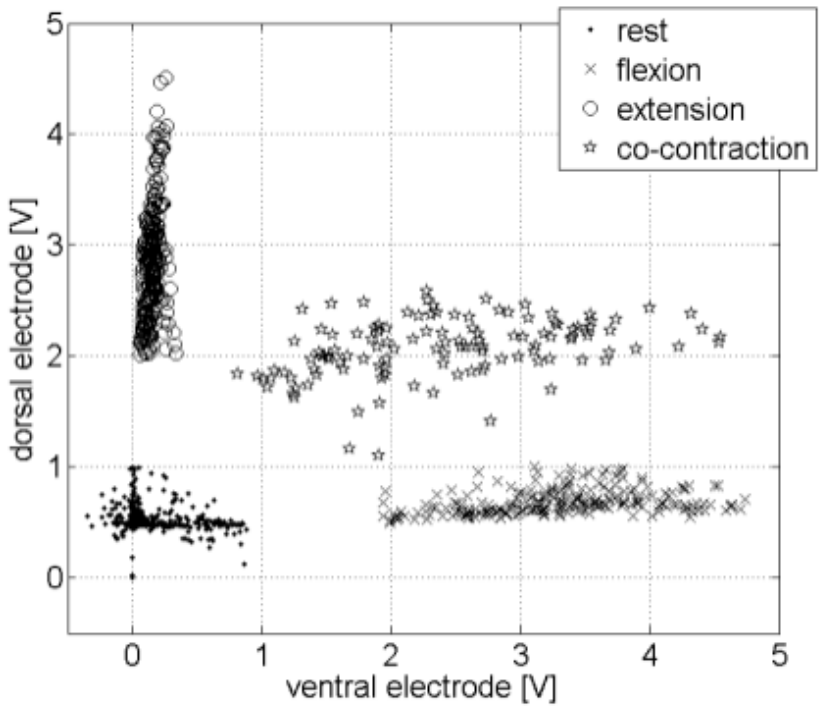
In classification problems we are given a set of training data

$$\mathcal{D} = \{(X^n, z^n), n = 1 \ldots, N\},$$

where $z \in \{0, 1, \ldots K-1\}$.

Goal: Assign unknown input vector X to one of K classes.

Example: electromyography (EMG) data recorded at the skin surface.



Basic Classification: Zero-one loss

We are interested in the amount of samples we get right:

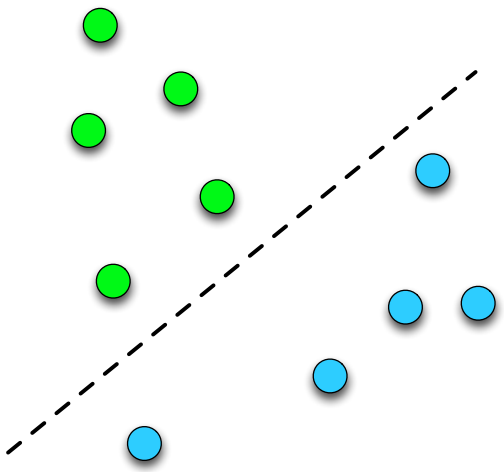
$$\sum_{i=1}^{N} \mathbb{I}(y_i = z_i).$$

Ideally, we are interested in the amount of *future* samples we get right.

How should this be done?

Hyperplanes as a Decision Boundary

How to decide whether a point $x \in \mathbb{R}^n$ belongs to class 0 or 1? $\rightarrow$ Check on which side of a hyperplane the point is.



Let a plane be defined by its normal vector W and an offset b.

$$W^T X + b \begin{cases} = 0 & \text{if } X \text{ on the plane} \\ > 0 & \text{if } X \text{ on normal's side of plane} \\ < 0 & \text{else} \end{cases}$$

Hyperplanes are computationally very convenient: easy to evaluate.

A data set $\mathcal{D} = \{(X_n, z_n)\}$ is *linearly separable* if there exists a hyperplane for which all X_n with $z_n = 0$ are on one and all X_n with $z_n = 1$ on the other side.

The Perceptron

A historical algorithm for binary classification.

Decision rule

$$y = f(W^T X + b)$$

where f is the step function defined as:

$$f(\xi) = \begin{cases} 1 \text{ if } \xi > 0, \\ 0 \text{ else.} \end{cases}$$

Learning Rule for the Perceptron

Initialise parameters to any value, e.g., a zero vector: $W \leftarrow 0$.

While there is at least one misclassified X_i in the training set:

$$W \leftarrow \begin{cases} W + X_i \text{ if } z_i = 1, \\ W - X_i \text{ if } z_i = 0. \end{cases}$$

What is the learning rule for the bias?

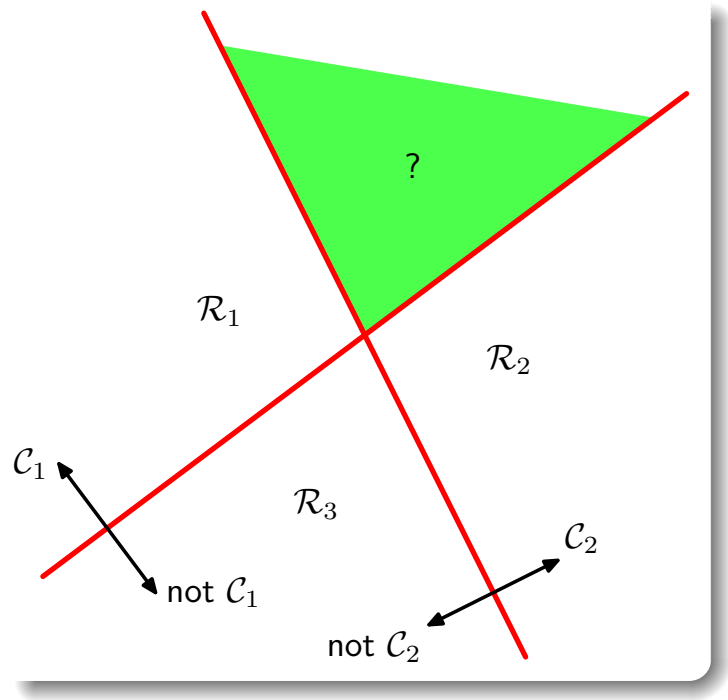
$$B \leftarrow \begin{cases} B + 1 \text{ if } z_i = 1, \\ B - 1 \text{ if } z_i = 0. \end{cases}$$

This method converges to a W discriminating between two classes *if it exists*.

Does the simple binary approach scale up to multiple classes?

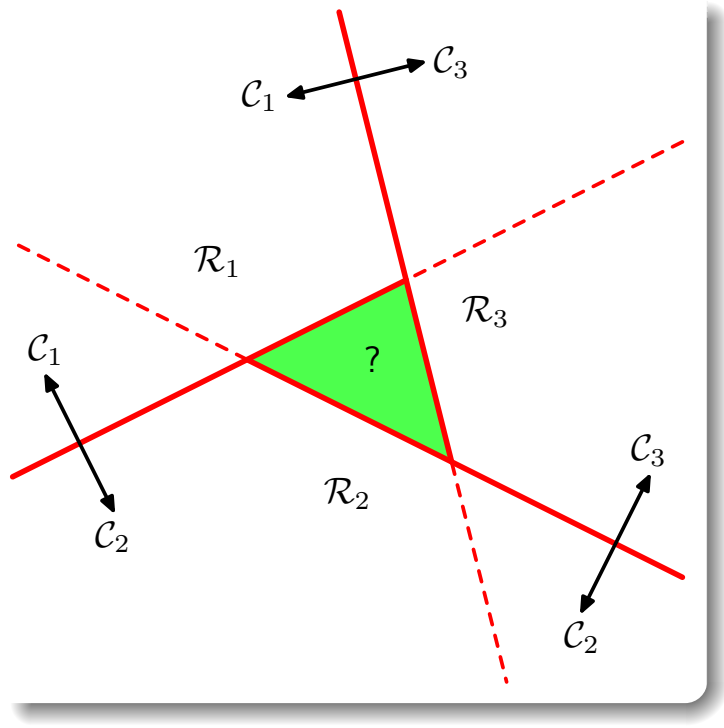
One-Versus-The-Rest Classifier

Separate points in one class from all other classes.



One-Versus-One Classifier

Have a discriminant function for each pair of classes and use majority vote to classify.



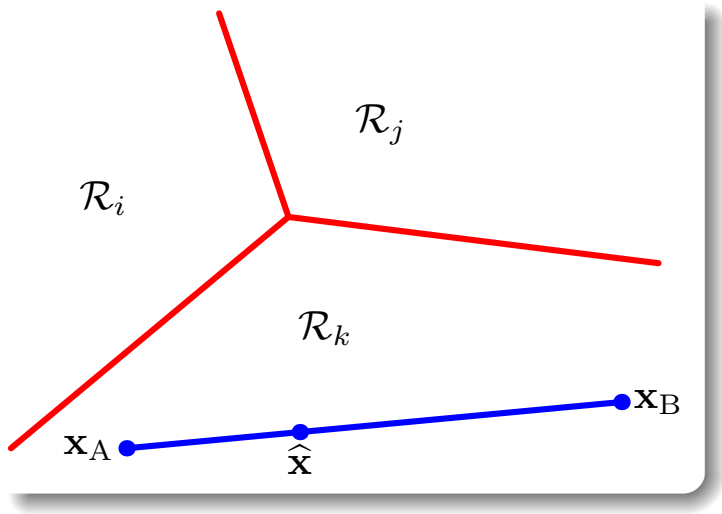
K-Class Discriminant

K linear functions of the form

$$y_k = W_k^T X + w_{k0}$$

where

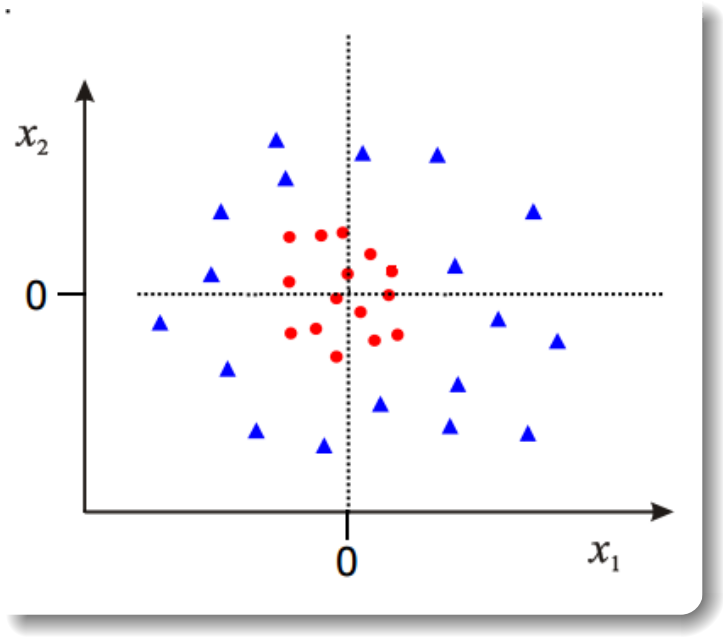
$$X \in \mathcal{C}_k \text{ if } y_k(X) > y_j(X) \quad \forall \quad j \neq k$$



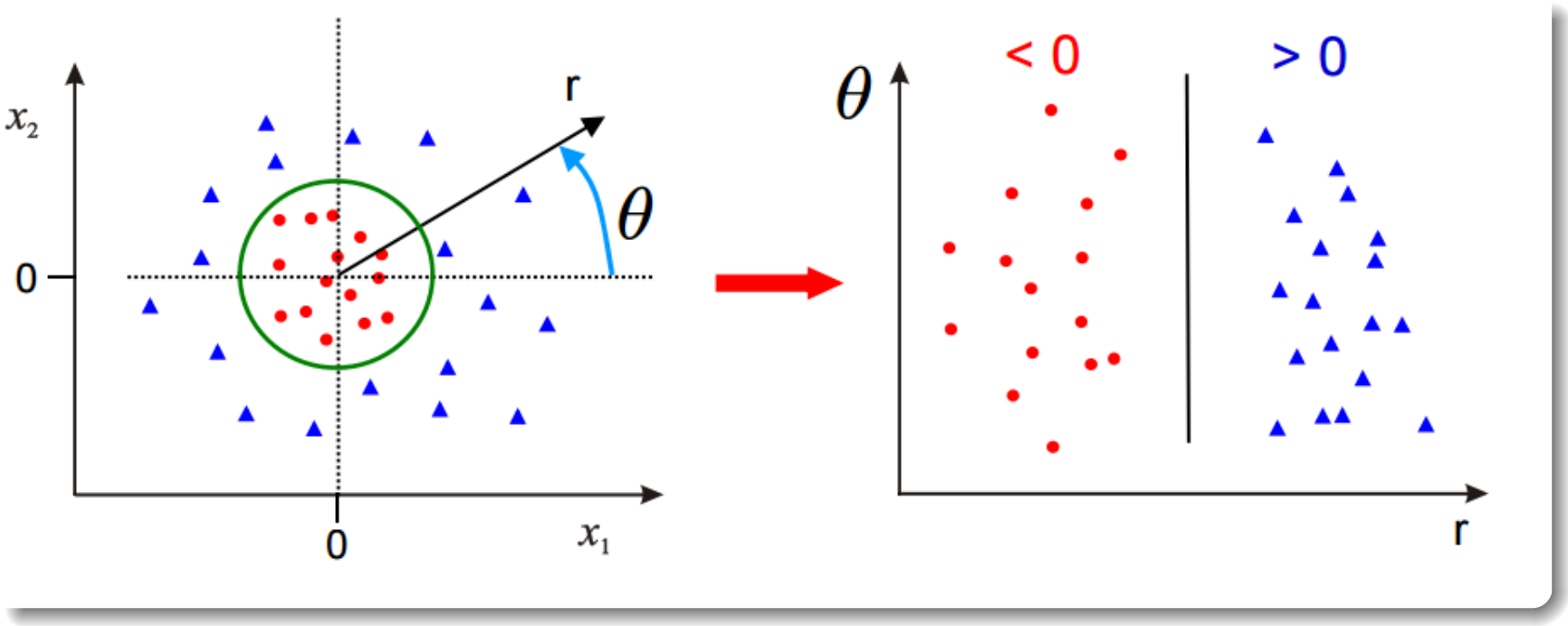
Non-Linearly Separable

Sometimes, data is not $\rightarrow$ try to find basis functions $\phi(X)$ in which the data is linearly separable (as in linear regression classes.)

This can be done manually. Or it can be *learned*, which will be taught to you in later classes.

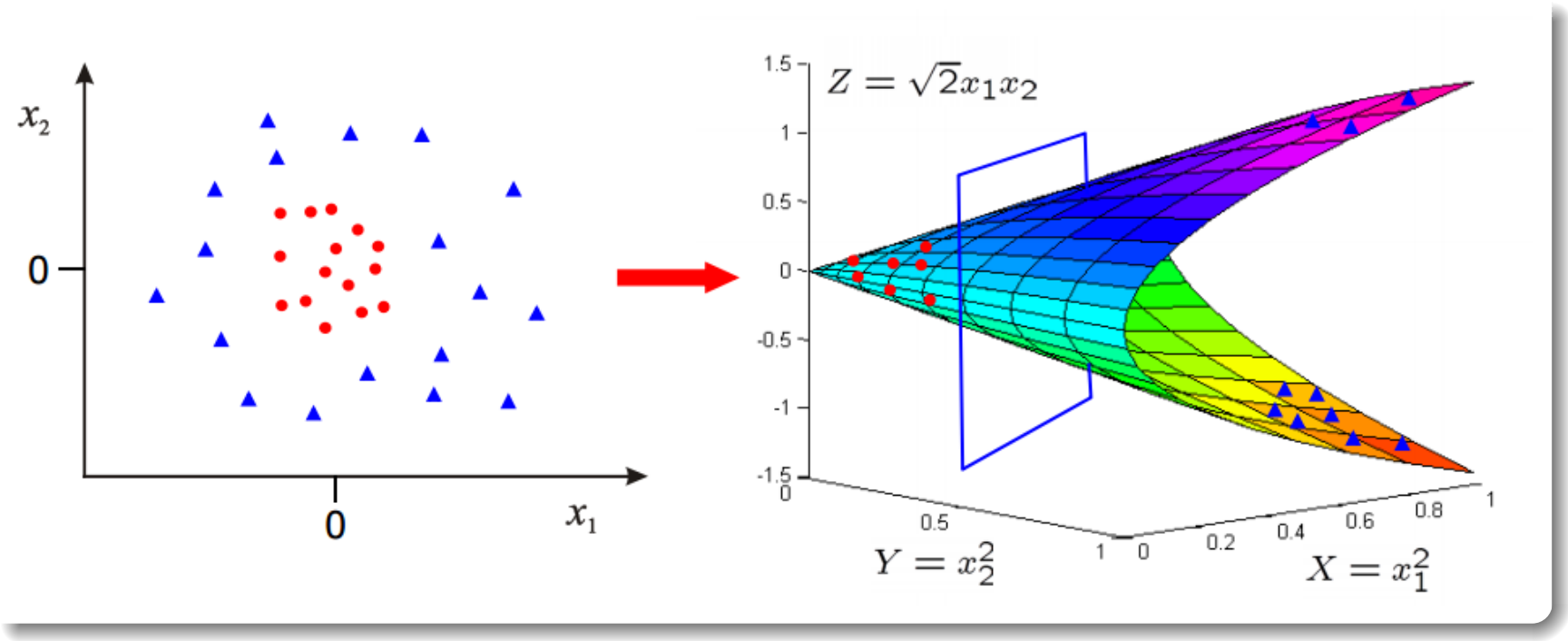


Non-Linearly Separable



$$\phi(X) : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix}$$

Non-Linearly Separable



$$\phi(X) : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Least Squares for Classification

$$y_k(X) = W_k^T X + w_{k0}$$
$$Y(X) = \mathbf{W}^T X$$

$$E_{\mathcal{D}}(\mathbf{W}) = \frac{1}{2} \mathrm{Tr}\left\{(\mathbf{X}\mathbf{W} - \mathbf{Z})^T (\mathbf{X}\mathbf{W} - \mathbf{Z})\right\} \tag{1}$$

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Z} = \mathbf{X}^\dagger\mathbf{Z} \tag{2}$$

$$Y(X) = \mathbf{W}^T X = \mathbf{Z}^T(\mathbf{X}^\dagger)^T X \tag{3}$$

This should all look familiar, but with more matrices instead of vectors.

(If not, review the lecture on basic linear regression.)

When the target vectors, z_n, in the training set, $\mathcal{D}$, all satisfy linear constraints, then

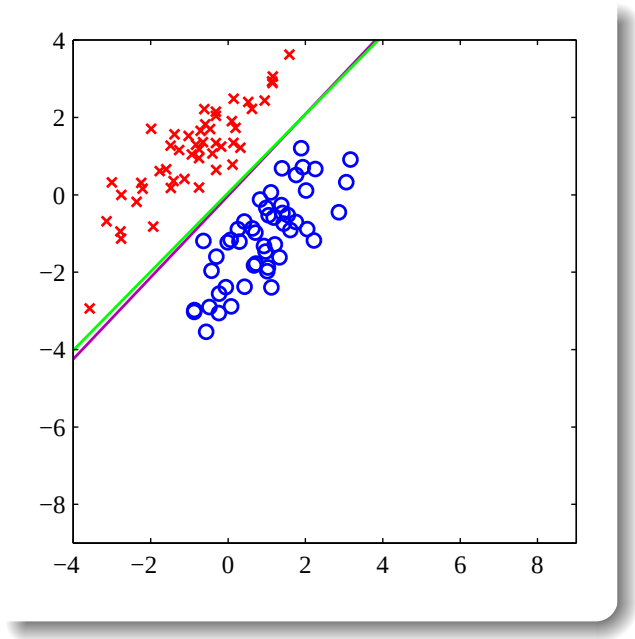
$$A^T Z_n + b = 0 \Leftrightarrow A^T y(X) + b = 0$$

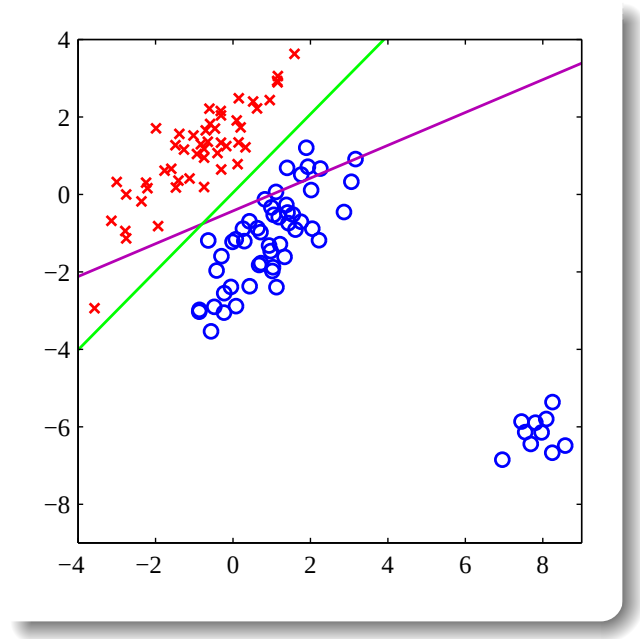
If we use 1-of-K coding, the elements of $y(X)$ will sum to 1.

Careful: This does not mean that we can interpret them as probabilities. The value is still not contrained to (0,1).

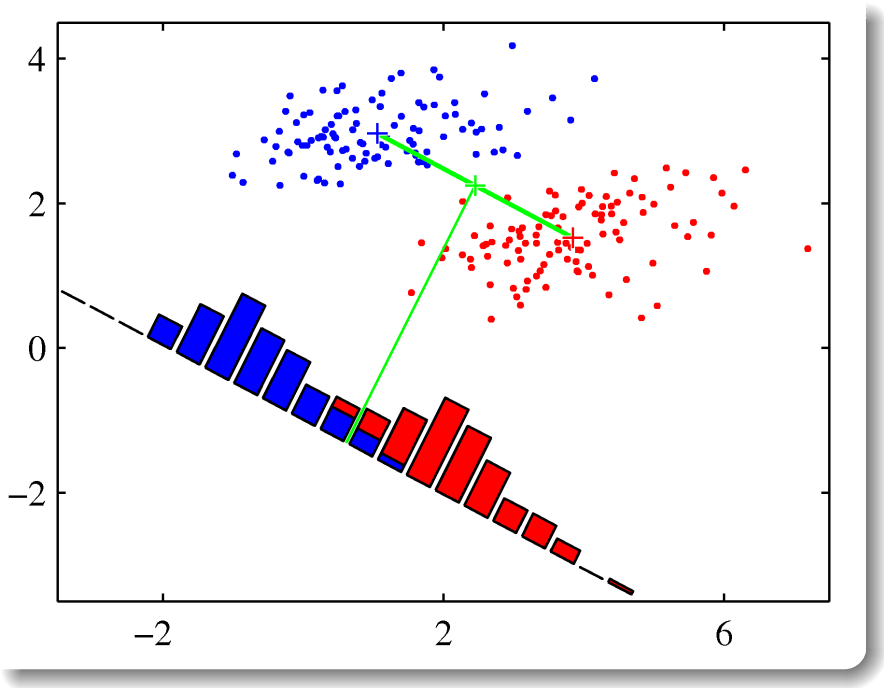
Problems with outliers: penalize predictions that are too correct

The reason why should be intuitive...

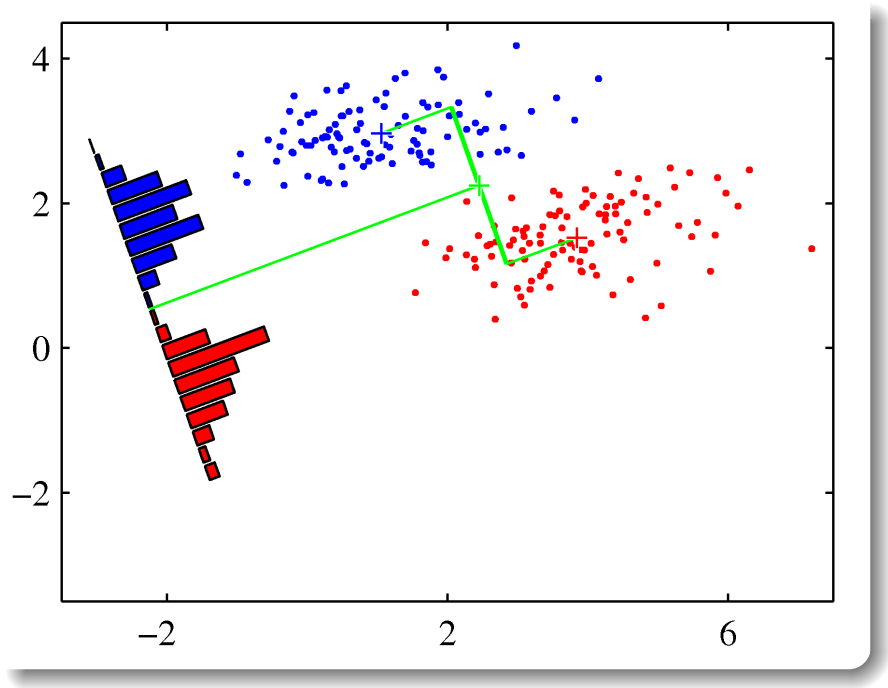




A Dimensionality Reduction View



Simplest approach: Choose the dimension that connects class means and project onto it.



Smarter approach: Instead, choose a dimension in which classes have maximum separation.

Fisher Criterion

Ratio of between-class variance to the within-class variance.

$$J(W) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{4}$$

where

$$\begin{aligned} m_1 &= \frac{1}{N_1} \sum_{n \in C_1} X_n, \\ m_2 &= \frac{1}{N_2} \sum_{n \in C_2} X_n, \\ m_k &= W^T M_k, \\ s_k^2 &= \sum_{n \in C_k} (y_n - m_k)^2, \\ y_n &= W^T X_n \end{aligned}$$

Which we can rewrite as

$$J(W) = \frac{W^T \mathbf{S_B} W}{W^T \mathbf{S_W} W} \tag{5}$$

where

$$\begin{aligned} \text{Between class} \quad \mathbf{S_B} &= (M_2 - M_1)^2 \\ \text{Within class} \quad \mathbf{S_W} &= \sum_{n \in C_1} (X_n - M_1)^2 + \sum_{n \in C_2} (X_n - M_2)^2 \end{aligned}$$

Approaches to Classification

$\Leftarrow$ Increasing Complexity $\Leftarrow$

Linear Discriminant Functions:

These approaches map each input directly onto a class label and probabilities play no role. If we increase the complexity of our algorithms we have two more choices.

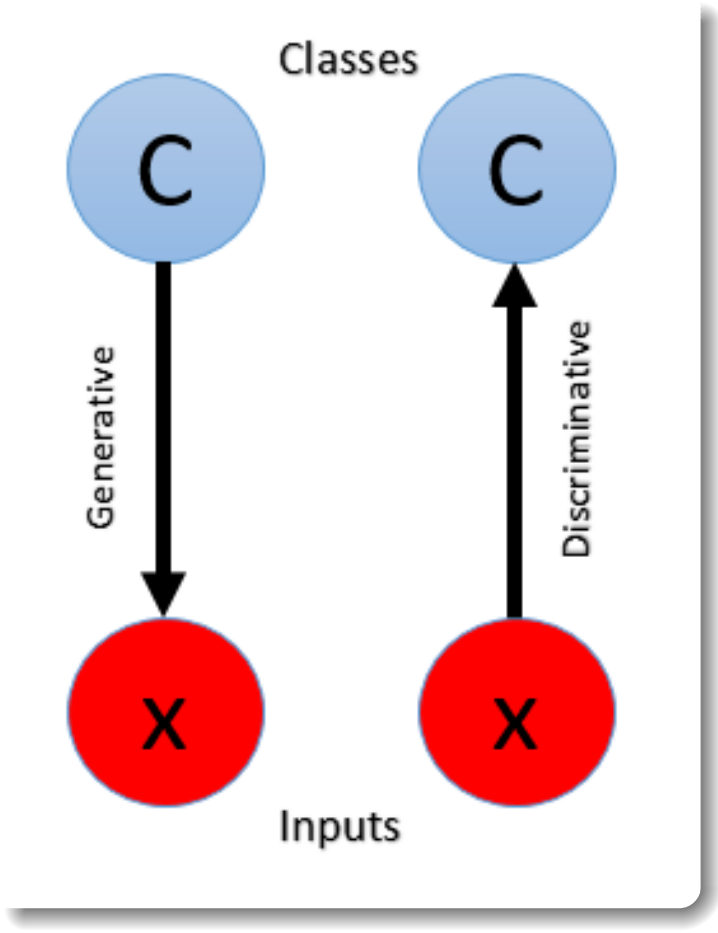
Discriminative Models:

These models solve the posterior class probabilities, $p(C_k \mid x)$, directly, and then assign each new x to a class using a suitable loss function or other decision function.

Generative Models:

Like discriminative models, these also solve for $p(C_k \mid x)$, but first have to determine $p(x \mid C_k)$ and $p(C_k)$ for each class individually. Afterwards, a decision function determines class membership.

Generative vs Discriminative



Linear Discriminative Models

by Grady Jensen

Reading Material:
"Pattern Recognition and Machine Learning" by Bishop [ch. 4, 4.1.1, 4.1.2, 4.1.7, 4.2, 4.3.2, 4.3.4]

Further extra reading:
"Machine Learning: A Probabilistic Perspective" by Murphy

Note: these slides are adapted from slides originally by Justin Bayer
Most figures are from C. Bishop: "Pattern Recognition and Machine Learning"

In the previous lecture we talked about discriminative functions where the function directly mapped each input into a class.

Now we will increase the complexity of our classification model and try to determine the posterior class probabilities for our problem.

In discriminative models, we model the conditional of the output given the input, but not the input. That is, we model $p(z \mid X)$, but not $p(X)$.

Notation

Symbol	Meaning
s	a scalar number is lowercase and not bold
S	A vector is uppercase
$\mathbf{S}$	a matrix is uppercase and bold
$y(X)$	distance from decision surface
y	predicted class label
z	actual class label
$\mathbb{I}(a = b)$	Indicator function; $\mathbb{I}(a) = 1$ if a else 0
z_n	The actual class label of the n'th example
$\mathcal{C}_k$	equivalent to $z =$ class k
$\mathcal{D}$	the training data $\mathcal{D} = \{(X_n, z_n), n = 1 \ldots, N\}$
$\phi(x)$	a basis function

There is not a special symbol for vectors or matrices augmented by the bias term, w_0. Assume it is always included as was done with linear regression.

Logistic Regression

Logistic regression is a discriminative model for classification: the output of our model is the parameter of a Bernoulli variable.

Let us start with binary classification.

$$p(\mathcal{C}_1 \mid X) = \sigma(b + X^T W),$$
$$p(\mathcal{C}_0 \mid X) = 1 - \sigma(b + X^T W),$$

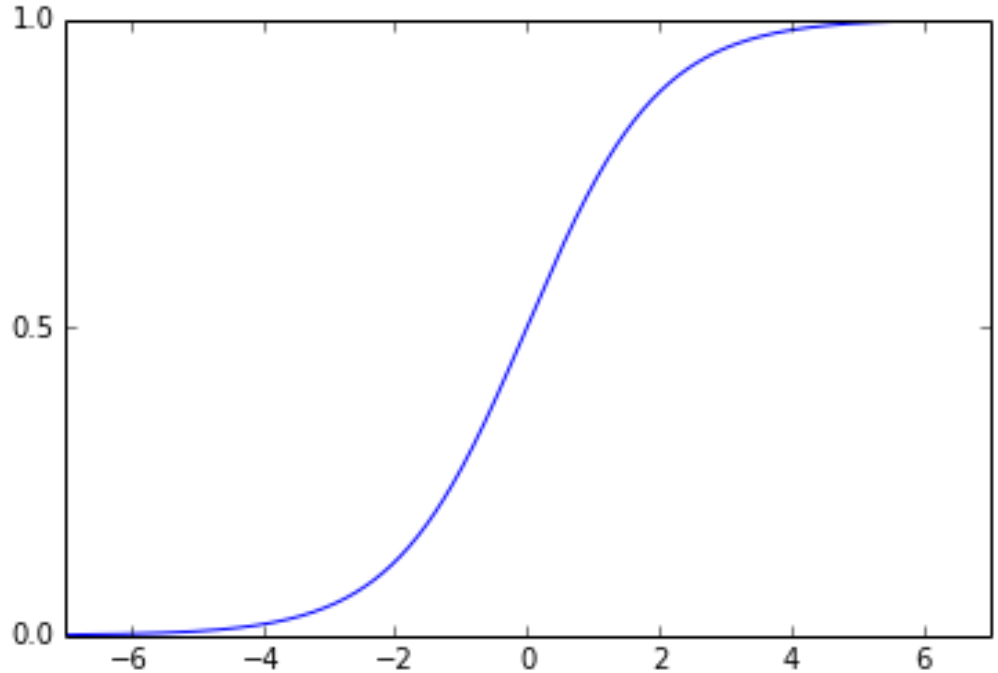
where σ is the *sigmoid* or *logistic* function that "squashes" the real numbers to the interval of $(0, 1)$.

The logistic (sigmoid) function

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

$$\sigma^{'}(a) = \sigma(a)(1 - \sigma(a)) \quad (2)$$

"...natural representation for the posterior probability in a binary classification problem." [1]



[1] "Why the logistic function? A tutorial discussion on probabilities and neural networks" by Michael Jordan

Maximum Likelihood for Logistic Regression

Learning logistic regression comes down to finding a "good" set of parameters $\{W, b\}$.

A standard approach is to write down the *likelihood* and try to optimise it with respect to those parameters.

Assuming that all the (X_i, z_i) are drawn independently, we can do so as follows.

$$
\begin{aligned}
p(Z \mid \mathbf{W}, \mathbf{X}) &= \prod_{n=1}^{N} p(z_n \mid X_n, W_n) \\
&= \prod_{n=1}^{N} p(z = 1 \mid X_n, W_n)^{z_n} \big(1 - p(z = 1 \mid X_n, W_n)\big)^{1 - z_n}.
\end{aligned}
$$

Let's inspect the two factors of each sample more closely.

$$\underbrace{p(z=1 \mid X_n, b, W)^{z_n}}_{=1 \text{ if } z_n=0} \underbrace{\big(1-p(z=1 \mid X_n, b, W)\big)^{1-z_n}}_{=1 \text{ if } z_n=1}$$

- Only one of the first two factors is "important" to the result, since the other one is the neutral element of multiplication.

Log-likelihood

Optimising over a product is difficult, since the terms interact. If we take the logarithm, the positions of the optima do not change, only their values. But since products change into sums, optimisation is much easier: each of the sum's terms is independent.

$$
\begin{aligned}
E(W) &= -\ln p\big(Z \mid W, X\big) \\
&= -\sum_{n=1}^{N} \left[z_n \ln y_n + (1 - z_n) \ln(1 - y_n) \right]
\end{aligned}
$$

where

$$y_n = \sigma(W^T \phi_n)$$

This loss function is harder to optimise than the one for linear regression—there is no closed form available.

Derivatives of the log-likelihood

With a little bit of math manipulation, we see

$$\nabla_W E_{\mathcal{D}}(W) = \sum_{n=1}^{N} \big(\sigma(W^T \Phi) - Z\big)\Phi_n \tag{3}$$

Look familiar?

Hint:

$$\ln(1 - \frac{a}{b}) = \ln(\frac{b-a}{b})$$
$$= \ln(b-a) - \ln b$$

and

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

where σ is as in eqn (1)

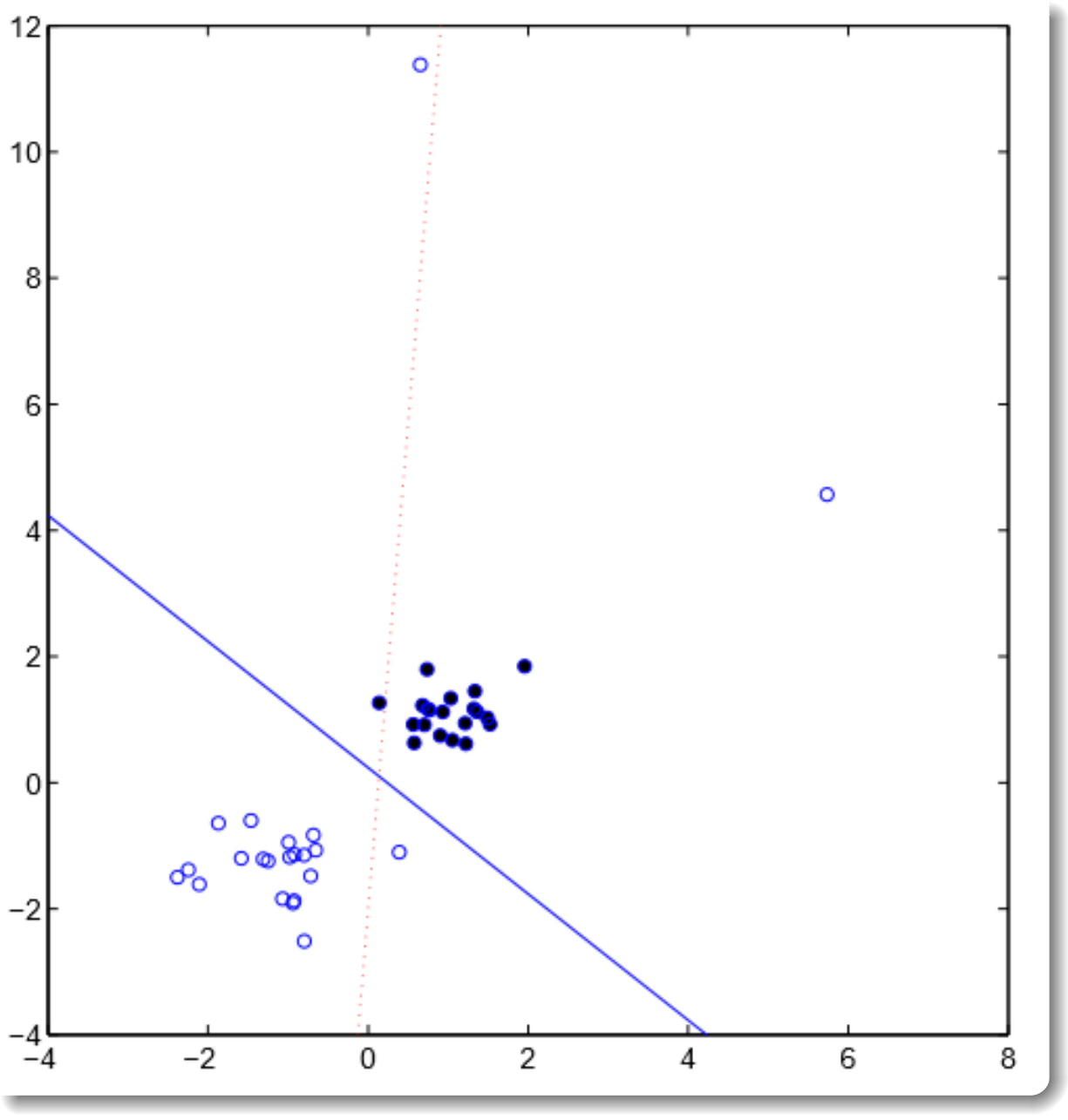
Soft Zero-One Loss

As we saw in linear regression, outliers (both correct and incorrect) can cause problems.

What if we minimise the following objective with respect to our parameter W:

$$\widetilde{E}(W) = \sum_{n=1}^{N} \Big[\sigma\big(\beta(W^T \Phi_n)\big) - z_n\Big]^2 + \lambda W^T W$$

- For $\beta \to \infty$ the above becomes the zero-one loss,
- λ is used to control the complexity of the model to prevent overfitting,
- The objective is no longer convex: that means that we are no longer guaranteed to find the optimum.



The red dotted line is Logistic Regression, while the solid blue line is soft loss.

Logistic regression misclassifies 3 points while soft loss messes up on 2.

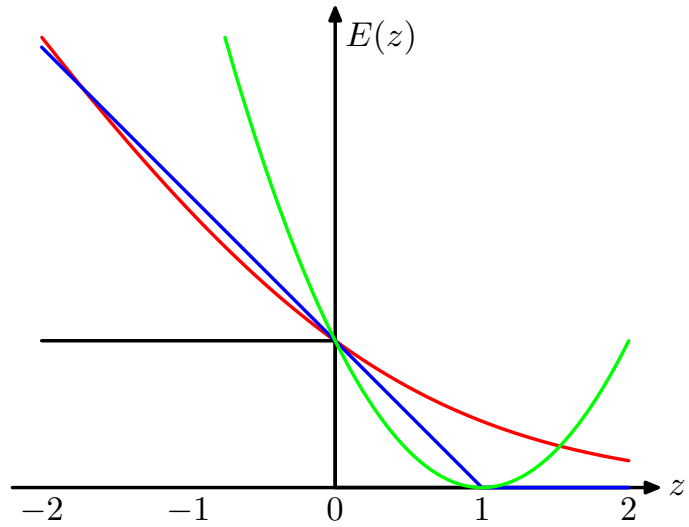
This image was taken from "Bayesian Reasoning and Machine Learning" by Barber

Hinge loss

One more alternative is to mimimise

$$\sum_{n=1}^{N} \max(0, 1 - \bar{z}_n(b + W^T X_n)),$$

where $\bar{z}_n = 2z_n - 1$.



- Although it is only locally differentiable, it works very good.
- A variant, *squared hinge loss* is a reasonable alternative.
- Objective is convex.

Multiclass problems

Most of the time, we do not have a binary classification problem, but one with dozens or thousands of labels. In that case, make z a vector Z that is zero everywhere and one (i.e., "hot") only at the index of the right class.

Example

$$Z = (0, 0, 1, 0, 0, 0)^T.$$

We will call these "hot k" or "one of k" vectors.

We then also have multiple W, stacked into the columns of a matrix $\mathbf{W}$.

Multiclass Logistic Regression

If we have multiple classes, we use multiple logistic regression models and normalize so the outputs sum up to 1. This is done via the *softmax* function. So we define our posterior function as follows:

$$p(\mathcal{C}_i \mid X) = \frac{e^{(W_i^T X)}}{\sum_j e^{(W_j^T X)}}$$

Interestingly, you can follow the same maximum likelihood steps we did earlier to derive the gradient of this function.

Hint: it has a form we've seen before :)

Non-closed form solutions

When we don't have a closed form solution to the problem, we must use an optimization algorithm to calculate it.

To do this we need to look at the gradient and the Hessian of the loss function. Using methods like steepest descent, Newton's method, BFGS, or iteratively reweighted least squares one can seek to find **a** minimum of the loss function.

The basics of optimization will be covered in a later lecture.

Linear Generative Models

by Grady Jensen

Reading Material:
"Pattern Recognition and Machine Learning" by Bishop [ch. 4, 4.1.1, 4.1.2, 4.1.7, 4.2, 4.3.2, 4.3.4]

Further extra reading:
"Machine Learning: A Probabilistic Perspective" by Murphy

Note: these slides are adapted from slides originally by Justin Bayer
Most figures are from C. Bishop: "Pattern Recognition and Machine Learning"

Notation

Symbol	Meaning
s	a scalar number is lowercase and not bold
S	A vector is uppercase
$\mathbf{S}$	a matrix is uppercase and bold
$y(X)$	distance from decision surface
y	predicted class label
z	actual class label
$\mathbb{I}(a = b)$	Indicator function; $\mathbb{I}(a) = 1$ if a else 0
z_n	The actual class label of the n'th example
$\mathcal{C}_k$	equivalent to $z =$ class k
$\mathcal{D}$	the training data $\mathcal{D} = \{(X_n, z_n), n = 1 \ldots, N\}$
$\phi(x)$	a basis function

There is not a special symbol for vectors or matrices augmented by the bias term, w_0. Assume it is always included as was done with linear regression.

In the previous lecture we talked about discriminative models where we modelled the conditional of the output given the input, but not the input. That is, we modelled $p(z \mid X)$, but not $p(X)$

Now we will increase the complexity of our classification model once again and try to determine the posterior class probabilities along with the likelihoods and the priors for our problem.

Like discriminative models, these also solve for $p(C_k \mid x)$, but first have to determine $p(x \mid C_k)$ and $p(C_k)$ for each class individually. Afterwards, a decision function determines class membership.

Generative Models

Intuitive idea: For each class, estimate a model. For a new input X, check to which model it fits best.

Formal idea: Use $p(\mathcal{C}_k)$ and $p(X \mid \mathcal{C}_k)$ and Bayes to get $p(\mathcal{C}_k \mid X)$.

What is $p(\mathcal{C}_k)$?

It gives us the probability that an unseen training point belongs to class k.

What is $p(X \mid \mathcal{C}_k)$?

It gives us the probability of a point given a class.

Finally, $p(\mathcal{C}_k \mid X)$?

It gives us the probability that a given point belongs to a class.

Model for the class priors

How do we model

$$p(\mathcal{C}_k)?$$

Use categorical distribution. For each of the classes $0, \ldots, k-1$ there is one parameter $0 \leq \theta_i \leq 1$. Furthermore $\sum_i \theta_i = 1$. So

$$p(\mathcal{C}_i \mid \theta) = \theta_i.$$

The maximum likelihood estimator for this is the fraction of samples that fall into the class—plain counting.

This method goes by many names; empirical Bayes and evidence approximation are more common in machine learning literature.

How can we justify doing this?

Like we discussed in previous lectures, we can also use distributions as priors, like gaussian or poisson.

Class conditionals (Continuous)

We have to make assumptions about the class conditional densities. Let's choose a multivariate Gaussian:

$$p(X \mid \mathcal{C}_k) = \frac{1}{|2\pi\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(X - M_k)^T \mathbf{\Sigma}^{-1}(X - M_k)\right\}$$

We can estimate all the parameters $\{M_0, \ldots, M_{k-1}, \mathbf{\Sigma}\}$ by maximum likelihood for the MVG on the subsets corresponding to the class of the data set. For the covariance, we take the full data set.

Likelihood

Once we have a parametric forms of $p(X \mid \mathcal{C}_k)$ and $p(\mathcal{C}_k)$ we can use maximum likelihood to determine the values of the parameters.

Let's suppose we have a dataset $\{X_n, z_n\}$ and $z_n = 1$ means class $\mathcal{C}_1$ and $z_n = 0$ means class $\mathcal{C}_2$. The priors, $p(\mathcal{C}_1)$ and $p(\mathcal{C}_2)$ are denoted by π and $1 - \pi$, respectively.

When $z_n = 1$,

$$p(X_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(X_n \mid \mathcal{C}_1) = \pi\mathcal{N}(X_n \mid M_1, \mathbf{\Sigma})$$

A when $z_n = 0$,

$$p(X_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(X_n \mid \mathcal{C}_2) = (1 - \pi)\mathcal{N}(X_n \mid M_2, \mathbf{\Sigma})$$

thus the likelihood is,

$$p(Z \mid \pi, M_1, M_2, \mathbf{\Sigma}) = \prod_{n=1}^{N} [\pi\mathcal{N}(X_n \mid M_1, \mathbf{\Sigma})]^{z_n} [(1 - \pi)\mathcal{N}(X_n \mid M_2, \mathbf{\Sigma})]^{1 - z_n} \tag{1}$$

Maximum Likelihood

If we maximize according to π, then

$$\pi = \frac{1}{N}\sum_{n=1}^{N} z_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2} \tag{2}$$

maximizing according to M_1 and M_2 gives us

$$M_1 = \frac{1}{N_1}\sum_{n=1}^{N} z_n X_n \tag{3}$$

$$M_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - z_n) X_n \tag{4}$$

and finally according to $\boldsymbol{\Sigma}$,

$$-\frac{1}{2}\sum_{n=1}^{N} z_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} z_n (X_n - M_1)^T \boldsymbol{\Sigma}^{-1}(X_n - M_1) -$$

$$\frac{1}{2}\sum_{n=1}^{N}(1-z_n)\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}(1-z_n)(X_n - M_2)^T \boldsymbol{\Sigma}^{-1}(X_n - M_2)$$

$$= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2} Tr\{\boldsymbol{\Sigma}^{-1}\boldsymbol{S}\}$$

where

$$\boldsymbol{S} = \frac{N_1}{N}\boldsymbol{S}_1 + \frac{N_2}{N}\boldsymbol{S}_2$$

$$\boldsymbol{S}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(X_n - M_1)(X_n - M_1)^T$$

$$\boldsymbol{S}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(X_n - M_2)(X_n - M_2)^T$$

Gaussian Discriminant Analysis ($K = 2$)

$$p(\mathcal{C}_1 \mid X) = \frac{p(X \mid \mathcal{C}_1)\, p(\mathcal{C}_1)}{p(X \mid \mathcal{C}_0)p(\mathcal{C}_0) + p(X \mid \mathcal{C}_1)\, p(\mathcal{C}_1)}.$$
$$= \frac{1}{1 + e^{(-a)}} = \sigma(a)$$

Where we have defined

$$a = \ln \frac{p(X \mid \mathcal{C}_1)p(\mathcal{C}_1)}{p(X \mid \mathcal{C}_2)p(\mathcal{C}_2)}$$

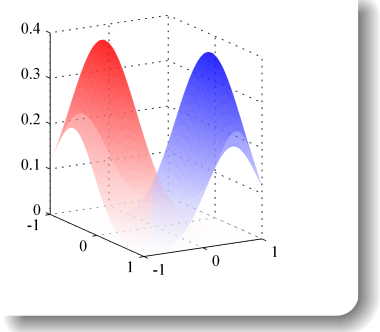


Figure : Class conditional densities

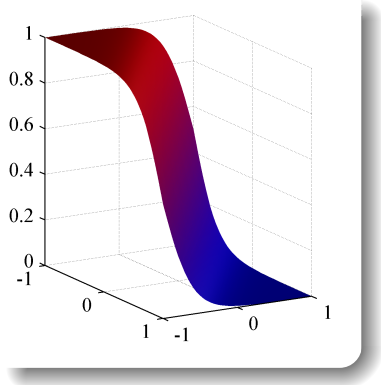


Figure : $p(\mathcal{C}_1 \mid X)$

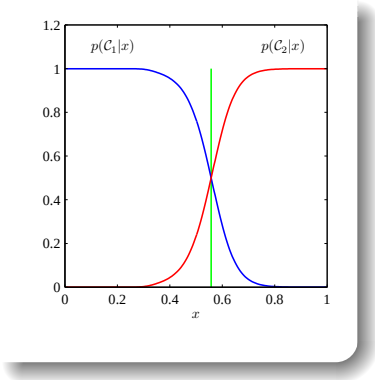


Figure : How decision functions can be determined

Form of the Posterior (Continuous)

For the case with $K = 2$ we have

$$\begin{aligned} p(\mathcal{C}_1 \mid X) =& \frac{1}{1 + \exp\big(-(W^T X + w_0)\big)} \\ =& \sigma(W^T X + w_0) \end{aligned}$$

with

$$\begin{aligned} W &= \mathbf{\Sigma}^{-1}(\mu_1 - \mu_2) \\ w_0 &= -\frac{1}{2}\mu_1^T \mathbf{\Sigma}^{-1}\mu_1 + \frac{1}{2}\mu_2^T \mathbf{\Sigma}^{-1}\mu_2 + \ln \frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)} \end{aligned}$$

It's the same model as with logistic regression. But the parameters are estimated differently. Note the priors! Note that we don't have quadratic terms in X because of cancellation!

If they have similar forms, how do we pick which one to use?

Gaussian Discriminant Analysis ($K > 2$)

$$p(\mathcal{C}_k \mid X) = \frac{p(X \mid \mathcal{C}_k)\, p(\mathcal{C}_k)}{\sum_j p(X \mid \mathcal{C}_j)\, p(\mathcal{C}_j)}.$$

Does this look familiar?

What if I write it as

$$p(\mathcal{C}_k \mid X) = \frac{e^{\ln p(X|\mathcal{C}_k)\, p(\mathcal{C}_k)}}{\sum_j e^{\ln p(X|\mathcal{C}_j)\, p(\mathcal{C}_j)}} = \frac{e^{a_k}}{\sum_j e^{a_j}} \tag{5}$$

$$\text{Where } a_k = \ln p(X \mid \mathcal{C}_k)\, p(\mathcal{C}_k)$$

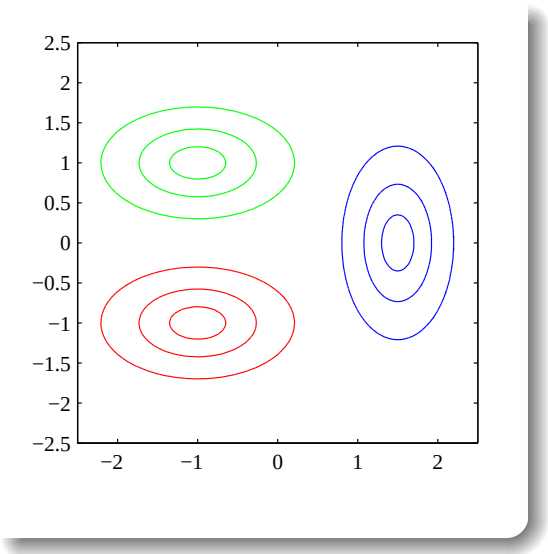
Form of the Posterior (Continuous)

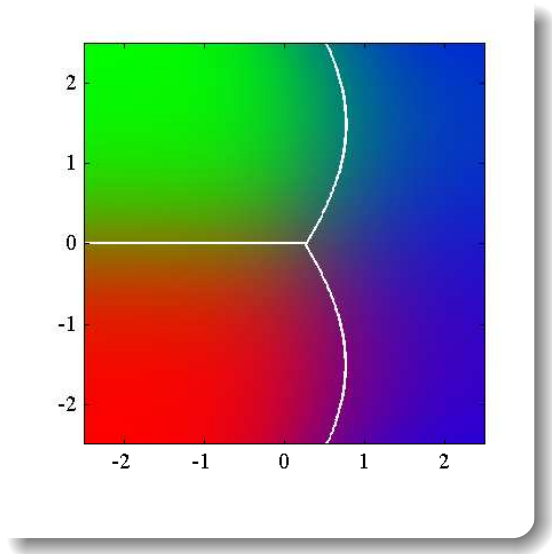
K > 2

Using equation (5)

$$a_k(X) = W_k^T X + w_{k0}$$
$$W_k = \mathbf{\Sigma}^{-1} M_k$$
$$w_{k0} = -\frac{1}{2} M_k^T \mathbf{\Sigma}^{-1} M_k + \ln p(\mathcal{C}_k)$$

We don't get the nice cancelation of quadratic in x if we don't have the same covariance matrix.





Outliers

Logistic regression and the generative model we showed are not robust towards outliers. The chance of a single point to be far from the right side of the decision boundary is exponentially small.

You should remember this from linear regression!

Discrete Data

So far we have only been discussing data that is continuous, what about discrete data?

Imagine we are trying to create a spam filter for email.

$$X_n^T = \begin{bmatrix} 1 & \#1 \\ 0 & 4U \\ 1 & credit \\ \vdots & \vdots \\ 0 & Sales \\ 1 & Success \\ 0 & Urgent \\ 0 & Winner \end{bmatrix}$$

Lets try to model the likelihood $p(\boldsymbol{X} \mid Z)$,

$$\text{vocabulary} = 10000 \text{ words}$$
$$z_{ni} \in \{0, 1\}^{10000}$$

If we tried to model $\boldsymbol{Z}$ with a multinomial we would need a $2^{10000} - 1$ dimensional parameter vector!

Naive Bayes

What if we assume that the possible outcomes are conditionally independent?

$$
\begin{aligned}
& p(x_1, \ldots, x_{10000} \mid z) \\
& \quad = p(x_1 \mid z)\, p(x_2 \mid z, x_1)\, p(x_2 \mid z, x_1, x_2) \ldots p(x_{10000} \mid z, x_1, \ldots, x_{9999}) \\
& \quad = p(x_1 \mid z)\, p(x_2 \mid z) \ldots p(x_{10000} \mid z) \\
& \quad = \prod_{i=1}^{N} p(x_i \mid z)
\end{aligned}
$$

In other words, if I tell you that an email is spam, the fact it contains the word "credit" has no effect on the probability of "Winner" being in the same email.

This is a **BIG** assumption, but interestingly, the algorithm still works well with many problems.

Pros/Cons (Murphy 8.6.1)

1. Easy to fit?
   - Generative is easy (Naive Bayes is just counting)
   - Logistic regression requires solution to complex optimization problem
2. Fit classes Separately?
   - We estimate the parameters of each class of a generative model independently.
   - All parameters interact in a discriminative model, so we must retrain with new class
3. Handle missing features?
   - Generative models can handle data Missing at Random(MAR) well
   - Descriminative assumes data always available to be conditioned on
4. Work with unlabeled training data?
   - Semi-supervised learning easier with generative than descriminative models
5. Can handle feature preprocessing?
   - Hard to define a generative model that can handle data put through pre-processing (basis functions,etc.)
6. Well-calibrated probabilities?
   - Some generative models can have extreme posterior values b/c of assumptions they make (ex. Naive Bayes)
   - Discriminative models usually don't have this problem (ex. logistic regression)