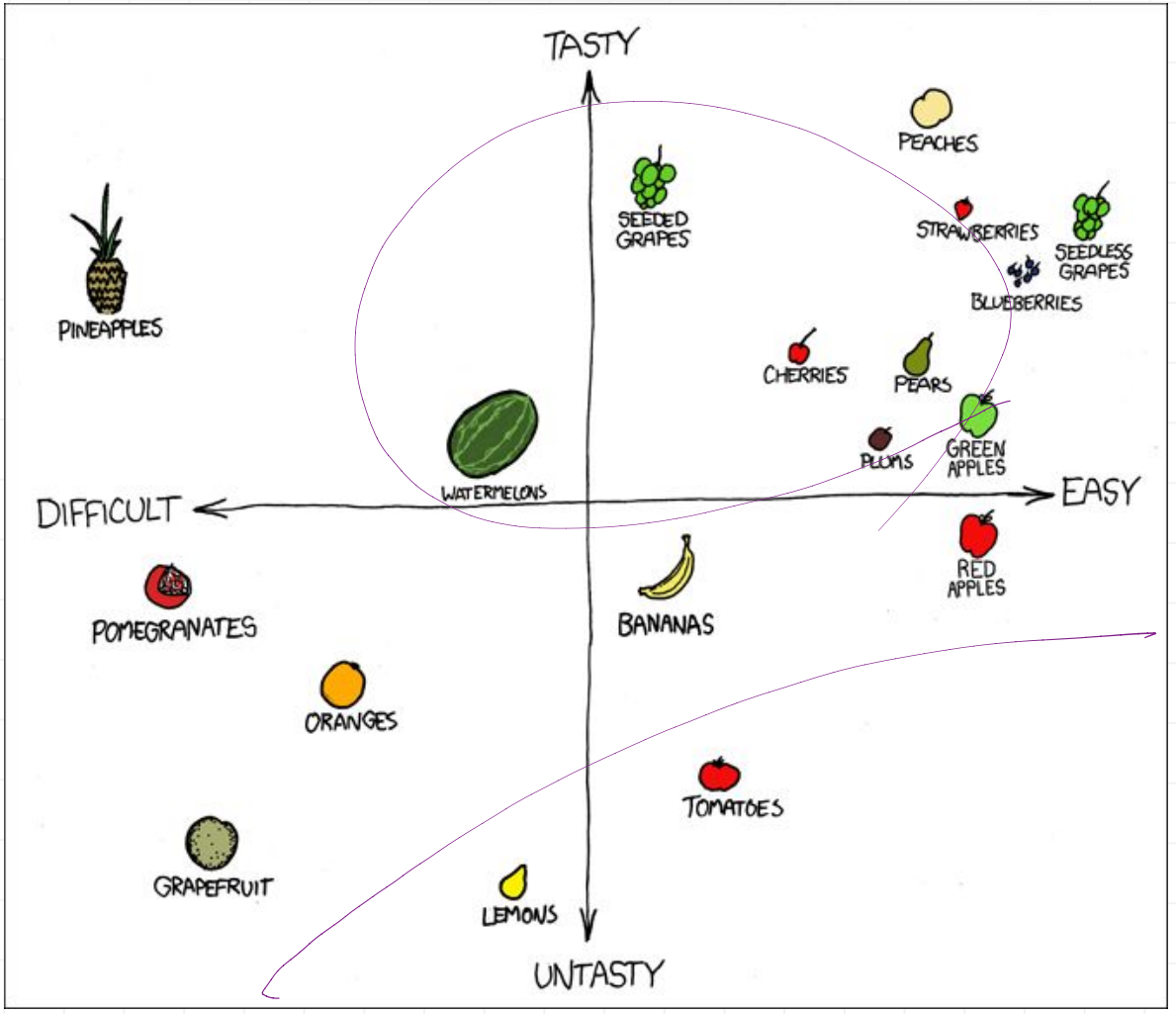
https://xkcd.com/388/



Announcements

In person lecture: Wed 16 March, PHYS T (will try *simulcast* in Teams, with us luck!)
https://studentvip.com.au/anu/main/maps/142757

Quiz 1 next week, due Thu (releasing by Mon)

Linear models for classification

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here

The perceptron algorithm

Probabilistic generative models - origin of the logistic function

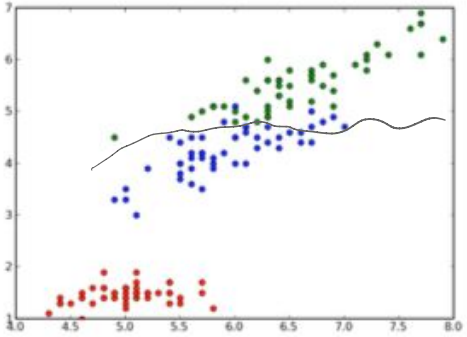
Probabilistic discriminative models
Logistic regression

Laplace approximation (Bayesian logistic regression) — later.

Origin of the logistic function, logistic regression and how it connects to perceptron

Classification

- Goal : Given input data $\mathbf{x}$, assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, \ldots, K$.
- Divide the input space into different regions.
- Equivalently: map each point to a categorical label.



Length of petal [in cm] vs sepal [cm] for three types of flowers (Iris Setosa, Iris Versicolor, Iris Virginica).

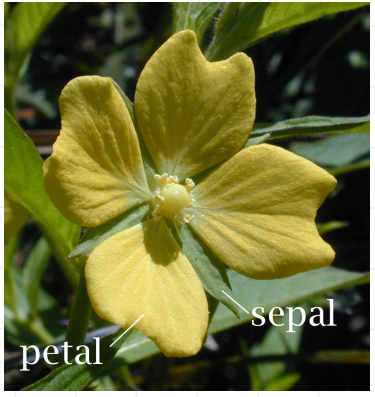


Image from wikipedia

Representing classes

binary $t \in \{0, 1\}$

Meeting in person/remote, pass/fail, benign/malignant, should be given credit (Y/N), ...

one-hot $\mathbf{t} = (0, 1, 0, 0, 0)^{\mathrm{T}}.$ (4.1)

$\sum_K t_K = 1$

Also denote $t_k \longrightarrow \mathcal{C}_k$

Iris flowers, object classes in photos, natural language, ...

Other representations abound, e.g. structured (time series, sequences – PRML Chapter 13, graphs), Sec 4.1.5 (an output representation that connects Fisher Discriminant to least squares), Sec 4.1.7 perceptrons, SVMs $\{-1, +1\}$

A primer on decision theory (Sec 1.5)

Inference: compute $p(\mathbf{x}, \mathbf{t})$, e.g. from a set of training data.

Decision theory: take a specific action based on our understanding of the values t is likely to take.

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}. \qquad (1.77)$$

(end of page 38) "We shall see that the decision stage is generally very simple, even trivial, once we have solved the inference problem. "

-- many counter-examples! e.g. multiple medical tests, test + quarantine for covid

-- frontiers of theoretical and practical ML, e.g. active/online learning, ML and economics

Minimise error

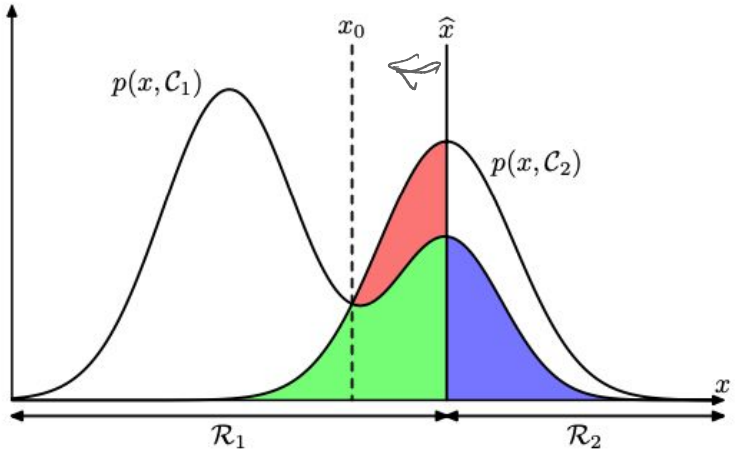


Figure 1.24 Schematic illustration of the joint probabilities $p(x, \mathcal{C}_k)$ for each of two classes plotted against x, together with the decision boundary $x = \widehat{x}$. Values of $x \geqslant \widehat{x}$ are classified as class $\mathcal{C}_2$ and hence belong to decision region $\mathcal{R}_2$, whereas points $x < \widehat{x}$ are classified as $\mathcal{C}_1$ and belong to $\mathcal{R}_1$. Errors arise from the blue, green, and red regions, so that for $x < \widehat{x}$ the errors are due to points from class $\mathcal{C}_2$ being misclassified as $\mathcal{C}_1$ (represented by the sum of the red and green regions), and conversely for points in the region $x \geqslant \widehat{x}$ the errors are due to points from class $\mathcal{C}_1$ being misclassified as $\mathcal{C}_2$ (represented by the blue region). As we vary the location $\widehat{x}$ of the decision boundary, the combined areas of the blue and green regions remains constant, whereas the size of the red region varies. The optimal choice for $\widehat{x}$ is where the curves for $p(x, \mathcal{C}_1)$ and $p(x, \mathcal{C}_2)$ cross, corresponding to $\widehat{x} = x_0$, because in this case the red region disappears. This is equivalent to the minimum misclassification rate decision rule, which assigns each value of x to the class having the higher posterior probability $p(\mathcal{C}_k|x)$.

$$
\begin{aligned}
p(\text{correct}) &= \sum_{k=1}^{K} p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\
&= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k)\, \mathrm{d}\mathbf{x}
\end{aligned}
$$

red+green

$$
\begin{aligned}
p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \quad \text{→ blue} \\
&= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\, \mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\, \mathrm{d}\mathbf{x}. \qquad (1.78)
\end{aligned}
$$

What are the potential problems for minimising classification error?

$$
\begin{aligned}
p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
&= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\, \mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\, \mathrm{d}\mathbf{x}. \qquad (1.78)
\end{aligned}
$$

Example scenario: medical diagnosis

Minimise loss, balanced metric, having a reject option

Figure 1.25 An example of a loss matrix with elements L_{kj} for the cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the assignment of class made by our decision criterion.

True

$$\begin{array}{c} \\ \text{cancer} \\ \text{normal} \end{array} \begin{array}{c} \begin{array}{cc} \text{cancer} & \text{normal} \end{array} \\ \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \end{array}$$

	Predicted 1 –Cancer	Predicted 0 – No Cancer	
Actual 1 -Cancer	90 [TP]	4 [FN]	Cancer Records [90+4=94]
Actual 0 – No Cancer	1 [FP]	5 [TN]	Non Cancer Records [5+1 =6]

$\frac{TP}{P} = \frac{90}{94}$

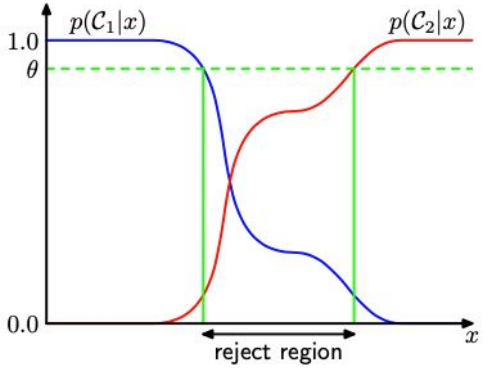
https://towardsdatascience.com/confusion-matrix-clearly-explained-fee63614dc7

$\frac{TN}{N} = \frac{5}{6}$

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) \, \mathrm{d}\mathbf{x}. \qquad (1.80)$$

Decision: $\text{argmin}_j \sum_k L_{kj} p(\mathcal{C}_k|\mathbf{x})$ (1.81)

Figure 1.26 Illustration of the reject option. Inputs x such that the larger of the two posterior probabilities is less than or equal to some threshold θ will be rejected.



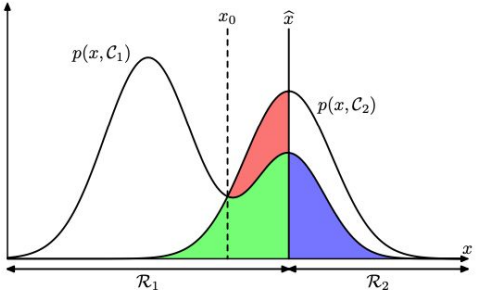
Brodersen, K. H.; Ong, C. S.; Stephan, K. E.; Buhmann, J. M., 2010. **The balanced accuracy and its posterior distribution**. International Conference on Pattern Recognition.

½ (TP/P + TN/N)

(many other metrics also address this problem)

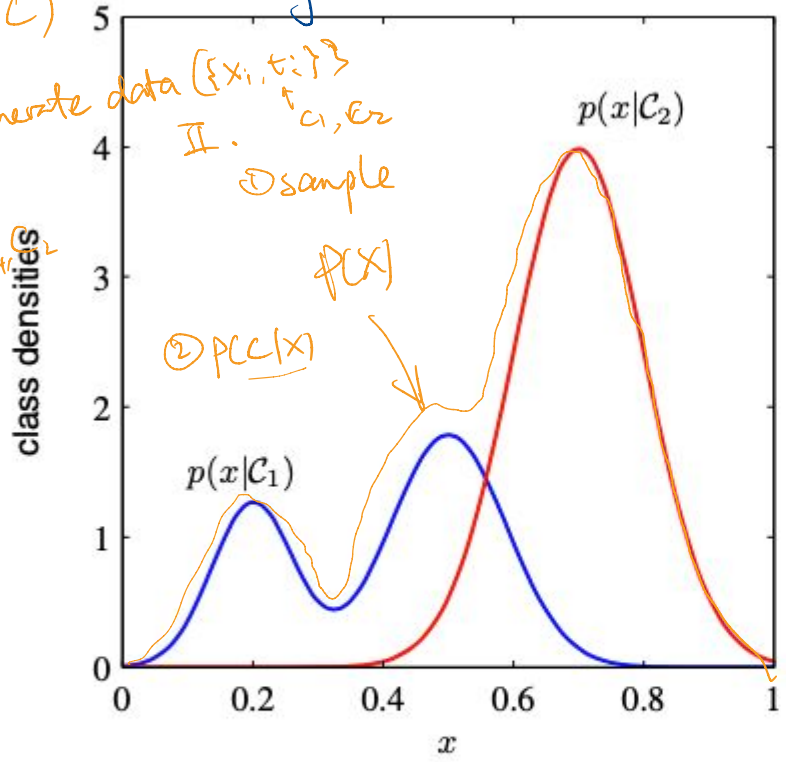
Three types of models for decision problems

- Find a discriminant function $f(\mathbf{x})$ which maps each input directly onto a class label.
- Discriminative Models
  1. Solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k \mid \mathbf{x})$.
  2. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.
- Generative Models
  1. Solve the inference problem of determining the class-conditional probabilities $p(\mathbf{x} \mid \mathcal{C}_k)$.
  2. Also, infer the prior class probabilities $p(\mathcal{C}_k)$.
  3. Use Bayes' theorem to find the posterior $p(\mathcal{C}_k \mid \mathbf{x})$.
  4. Alternatively, model the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ directly.
  5. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.



"generative"

"discriminative"



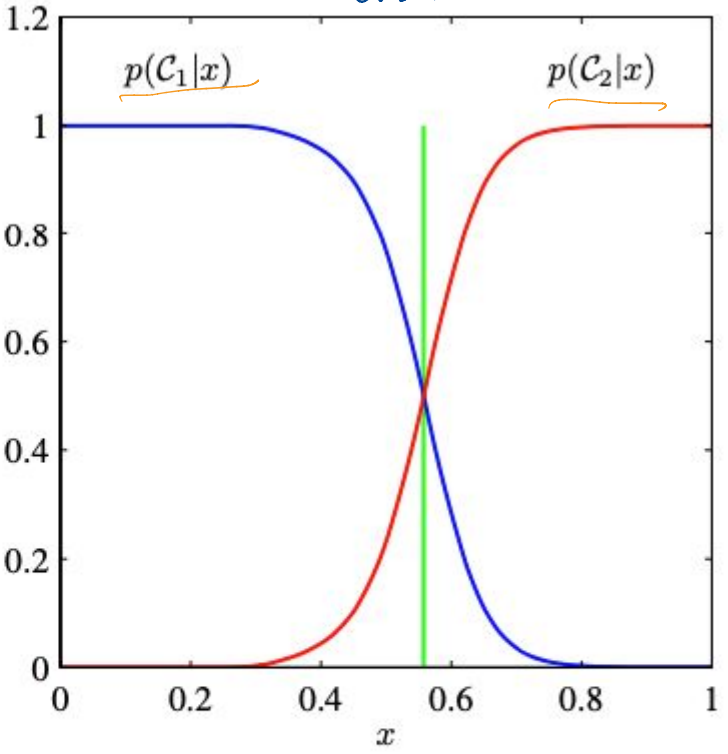
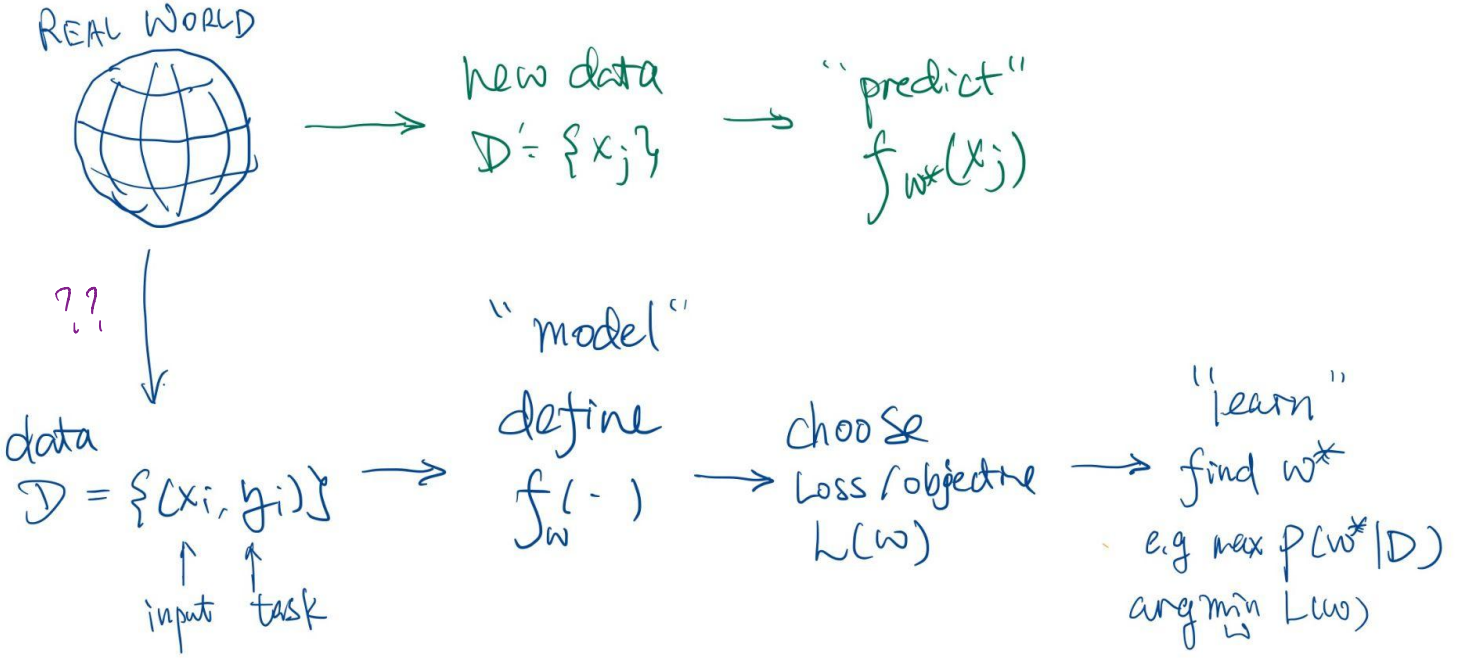


Figure 1.27 Example of the class-conditional densities for two classes having a single input variable x (left plot) together with the corresponding posterior probabilities (right plot). Note that the left-hand mode of the class-conditional density $p(\mathbf{x}|\mathcal{C}_1)$, shown in blue on the left plot, has no effect on the posterior probabilities. The vertical green line in the right plot shows the decision boundary in x that gives the minimum misclassification rate.

high level goal of "learning": the non-trivial dependency between X, C

A modular view of (supervised) ML



What we covered so far

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here
The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models
Logistic regression

(Laplace approximation, Bayesian logistic regression)

Discriminant function

Definition

A discriminant is a function that maps from an input vector $\mathbf{x}$ to one of K classes, denoted by $\mathcal{C}_k$.

- Consider first two classes ($K = 2$).
- Construct a linear function of the inputs $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \tag{4.4}$$

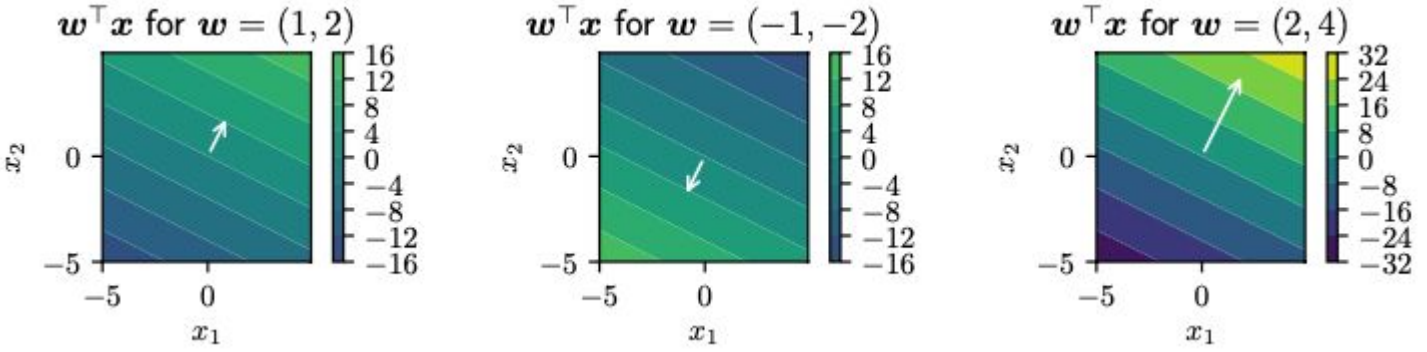
  such that $\mathbf{x}$ being assigned to class $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, and to class $\mathcal{C}_2$ otherwise.
- weight vector $\mathbf{w}$
- bias w_0 (sometimes $-w_0$ called threshold)

Linear discriminant functions

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \qquad (4.4)$$

- Decision boundary y(x)=0 -- is a hyperplane of D-1 dimensions (in a D-dimensional input space).
- **w** is orthogonal to any vector lying in the decision surface.

- Contour lines of y(x) are hyperplanes too
- y(**x**) is the signed distance of point x from the hyper-plane (see next page)



Discriminant for two classes

$y(\mathbf{x})$ gives a **signed** measure of the perpendicular distance r **from** the decision surface **to** $\mathbf{x}$, that is $r = y(\mathbf{x})/\|w\|$.

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \tag{4.4}$$

$$y(\mathbf{x}) = \mathbf{w}^T \overbrace{\left(\mathbf{x}_\perp + r\frac{\mathbf{w}}{\|w\|}\right)}^{\mathbf{x}} + w_0 = r\frac{\mathbf{w}^\top\mathbf{w}}{\|w\|} + \overbrace{\mathbf{w}^\top\mathbf{x}_\perp + w_0}^{0} = r\|w\|$$

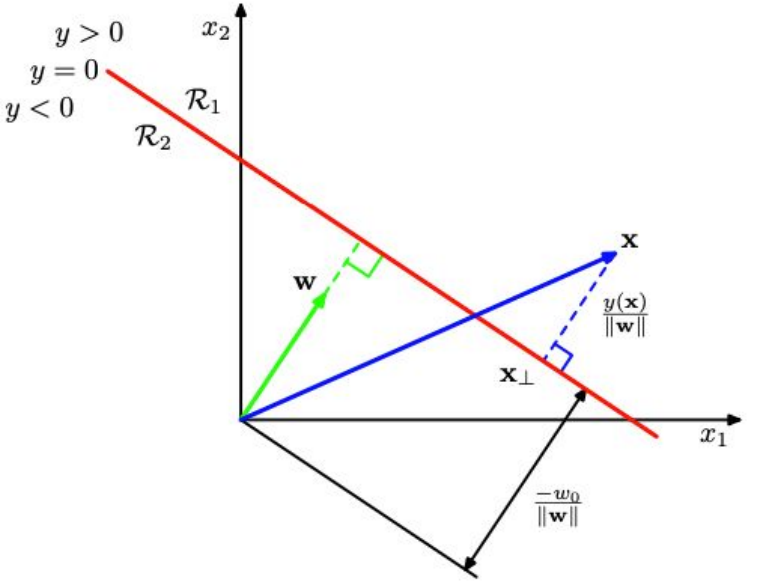


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

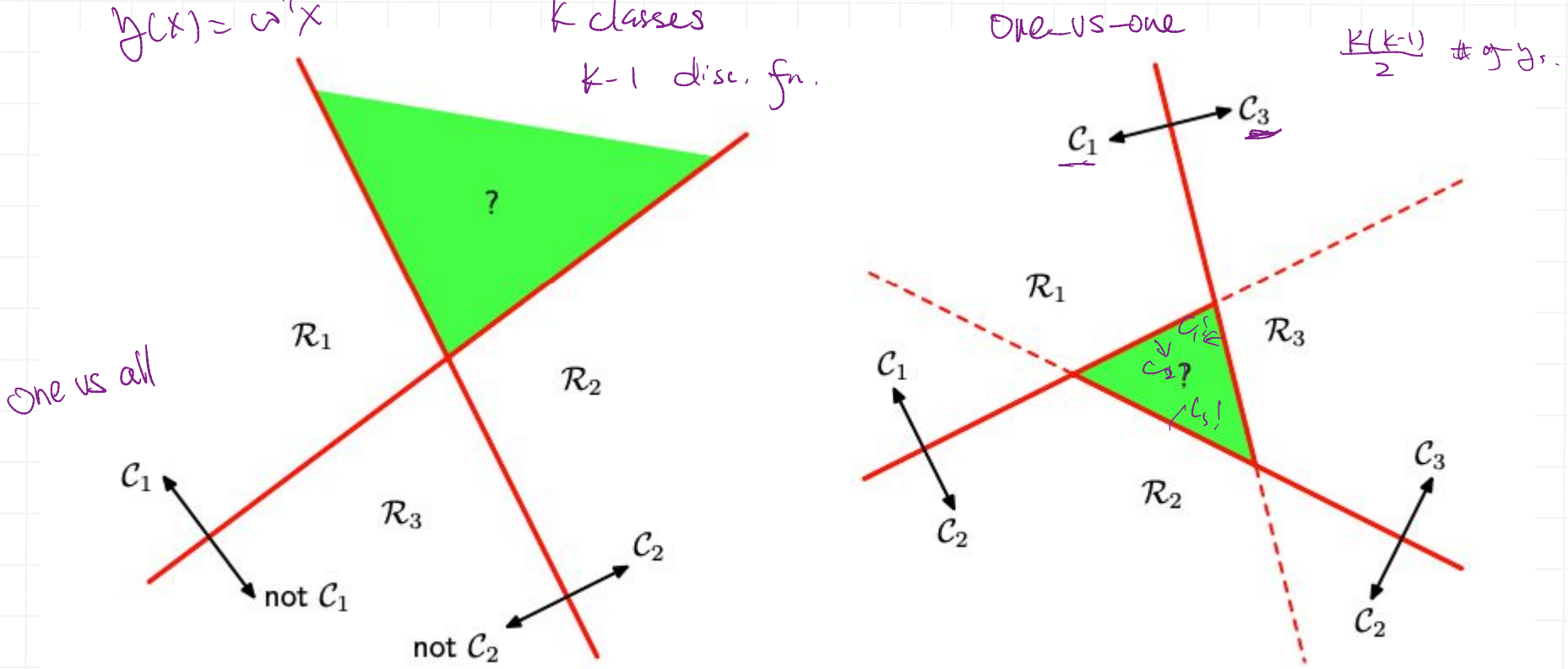


Figure 4.2 Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. On the right is an example involving three discriminant functions each of which is used to separate a pair of classes $\mathcal{C}_k$ and $\mathcal{C}_j$.

Discriminant function for K>=2 classes

Learn a series of functions k=1,.., K, assigning a point $\mathbf{x}$ to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$.

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \qquad (4.9)$$

$k^* \leftarrow \underset{k}{\operatorname{argmax}}\ y_k(x)$

Claim: The decision regions of such a discriminant are always singly connected and convex.

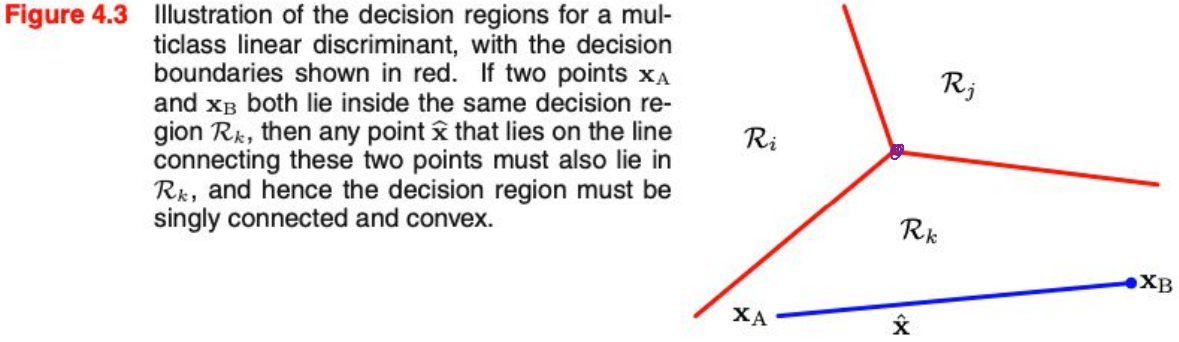
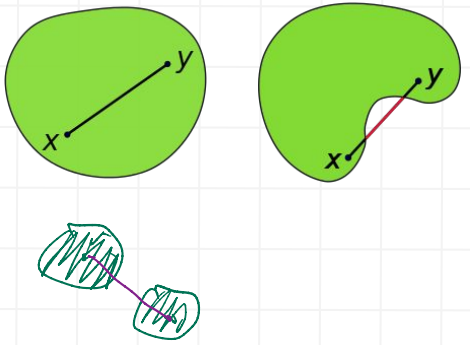


Figure 4.3 Illustration of the decision regions for a multiclass linear discriminant, with the decision boundaries shown in red. If two points $\mathbf{x}_\mathrm{A}$ and $\mathbf{x}_\mathrm{B}$ both lie inside the same decision region $\mathcal{R}_k$, then any point $\hat{\mathbf{x}}$ that lies on the line connecting these two points must also lie in $\mathcal{R}_k$, and hence the decision region must be singly connected and convex.

Non-Convex



$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_\mathrm{A}) + (1-\lambda) y_k(\mathbf{x}_\mathrm{B}).$$

Can we use least squares?

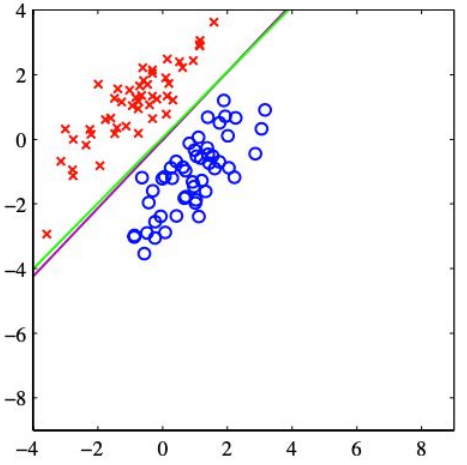
$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \tag{4.13}$$

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} \tag{4.14}$$

$t_n \in \{0, 1\}$

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}. \tag{4.15}$$

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T} \tag{4.16}$$



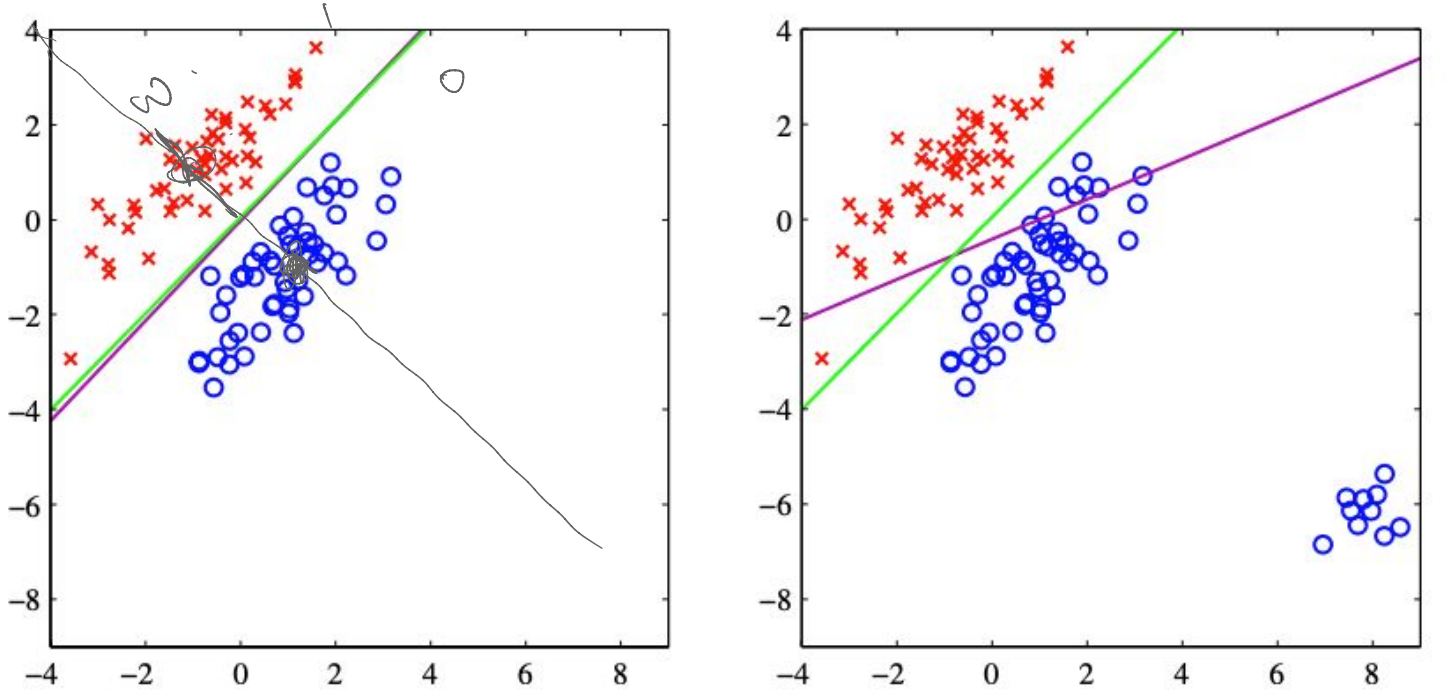


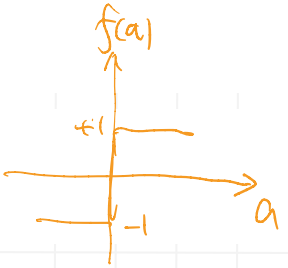
Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

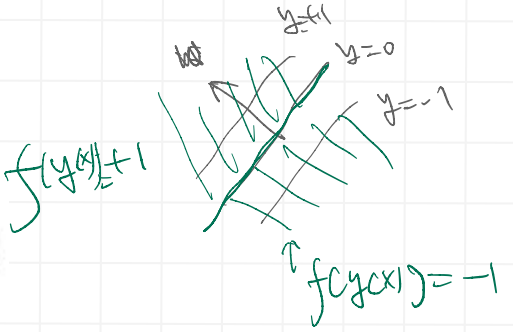
Mismatch between model requirement and model specification.

The (historically significant) perceptron [Rosenblatt 1962]

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) \tag{4.52}$$

$$f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0. \end{cases} \tag{4.53}$$





$$t \in \{-1, +1\}$$

$$\min \quad E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n t_n \tag{4.54}$$

learning rate

Stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\boldsymbol{\phi}_n t_n \tag{4.55}$$

"The *perceptron convergence theorem* states that if there exists an exact solution (in other words, if 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 … however, until convergence is achieved, we will not be able to distinguish between a nonseparable problem and one that is simply slow to converge."

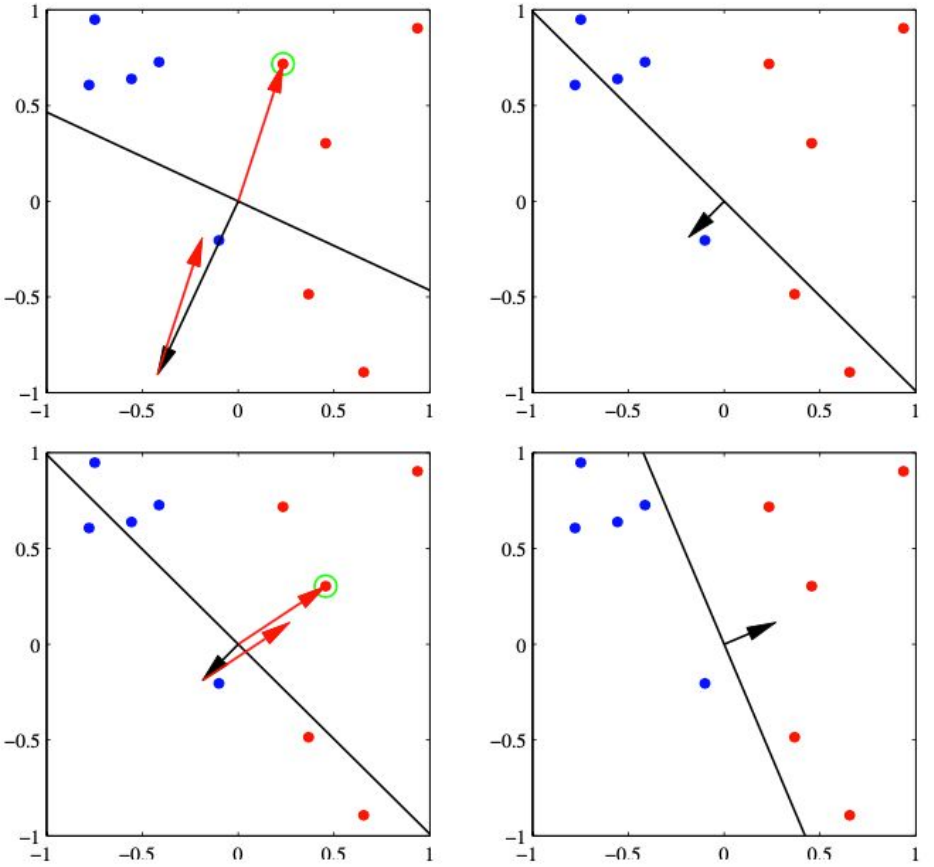


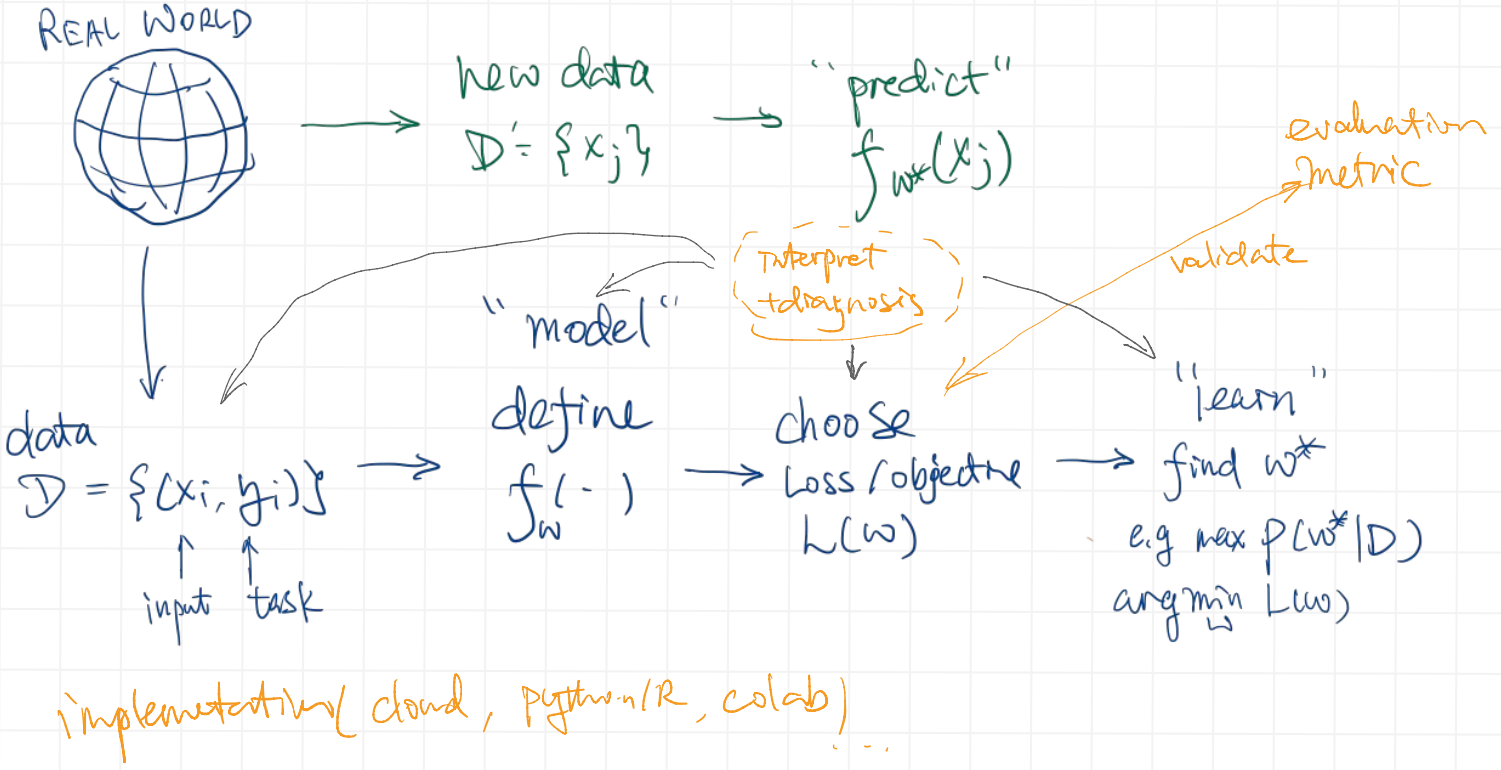
Figure 4.7 Illustration of the convergence of the perceptron learning algorithm, showing data points from two classes (red and blue) in a two-dimensional feature space (ϕ_1, ϕ_2). The top left plot shows the initial parameter vector $\mathbf{w}$ shown as a black arrow together with the corresponding decision boundary (black line), in which the arrow points towards the decision region which classified as belonging to the red class. The data point circled in green is misclassified and so its feature vector is added to the current weight vector, giving the new decision boundary shown in the top right plot. The bottom left plot shows the next misclassified point to be considered, indicated by the green circle, and its feature vector is again added to the weight vector giving the decision boundary shown in the bottom right plot for which all data points are correctly classified.

Figure 4.8 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a 20×20 array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

Another look at the modular view of supervised ML



What we covered so far

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here
The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models
Logistic regression

(Laplace approximation, Bayesian logistic regression)

Bayes theorem, again

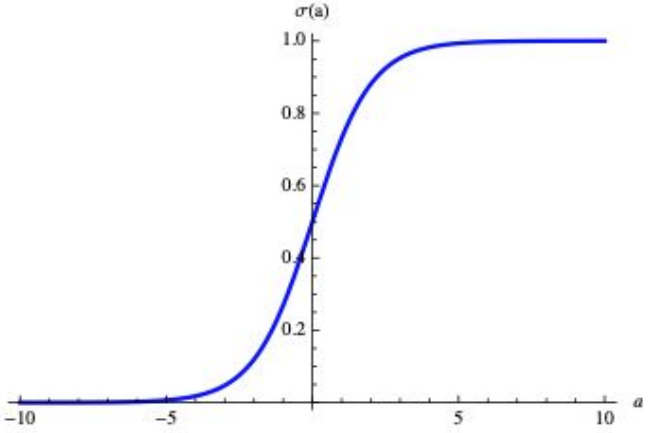
$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$ ← $p(\mathbf{x})$

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

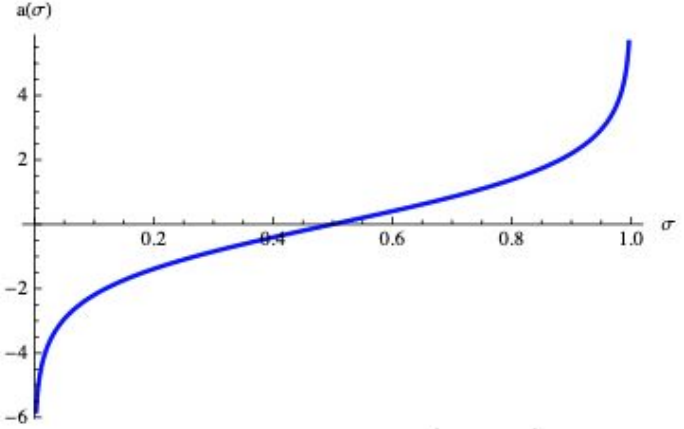
$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

logistic sigmoid function

$$= \frac{1}{1 + \exp(-a)} = \sigma(a) \qquad (4.57)$$



Sigmoid $\sigma(a) = \frac{1}{1+\exp(-a)}$



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

$\ln\left[p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})\right]$

a.k.a log odds

Symmetry: $\sigma(-a) = 1 - \sigma(a)$

Derivative: $\frac{\mathrm{d}}{\mathrm{d}a}\sigma(a) = \sigma(a)\,\sigma(-a) = \sigma(a)\,(1-\sigma(a))$

Probabilistic Generative Models - Multiclass

- The normalised exponential is given by

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

  where

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

- Usually called the softmax function as it is a smoothed version of the $\arg\max$ function, in particular:

$$a_k \gg a_j \;\forall j \neq k \Rightarrow \left(p(\mathcal{C}_k \,|\, \mathbf{x}) \approx 1 \wedge p(\mathcal{C}_j \,|\, \mathbf{x}) \approx 0\right)$$

- So, softargmax is a more descriptive though less common name.

Class-conditional distribution being Gaussians

cancel cancel

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}. \tag{4.64}$$

two classes C_1, C_2

class-means μ_1, μ_2

share cov. Σ

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

$-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)$

$x^T\Sigma^{-1}x$ → cancel

$\sigma(a)$

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0) \tag{4.65}$$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{4.66}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \tag{4.67}$$

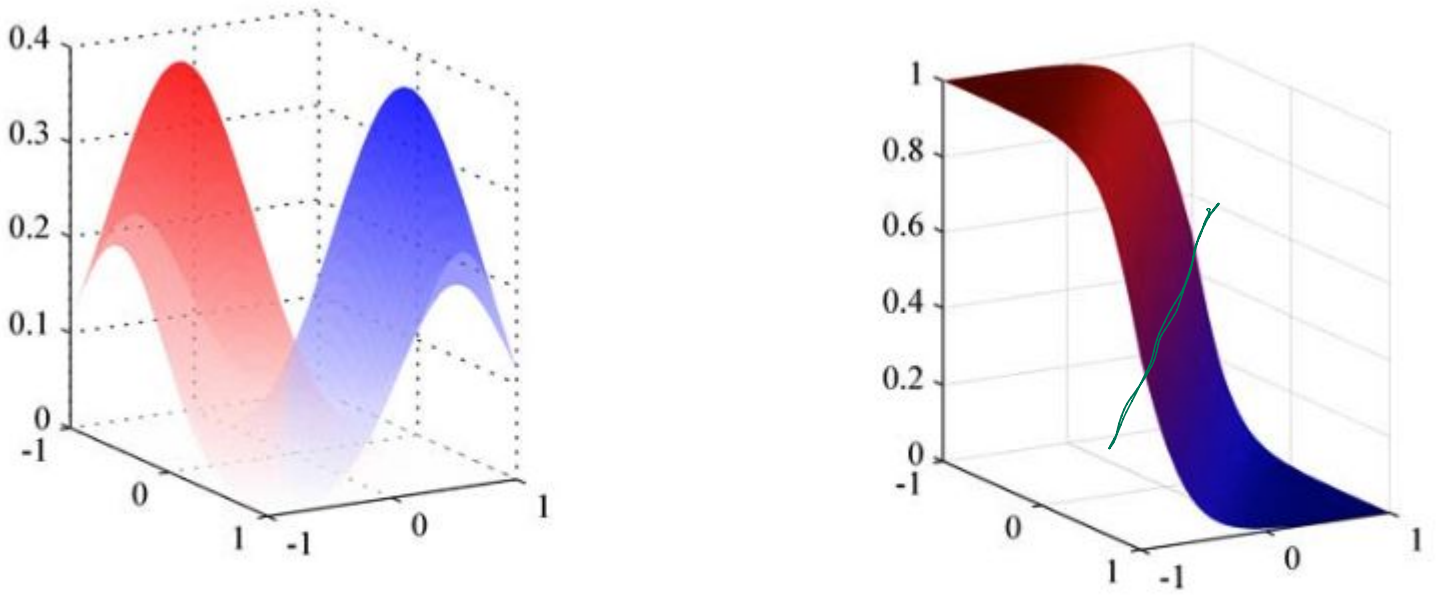


Figure 4.10 The left-hand plot shows the class-conditional densities for two classes, denoted red and blue. On the right is the corresponding posterior probability $p(\mathcal{C}_1|\mathbf{x})$, which is given by a logistic sigmoid of a linear function of $\mathbf{x}$. The surface in the right-hand plot is coloured using a proportion of red ink given by $p(\mathcal{C}_1|\mathbf{x})$ and a proportion of blue ink given by $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$.

- If the class-conditional distributions have different covariances, the quadratic terms $-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ do not cancel out.
- We get a quadratic discriminant.

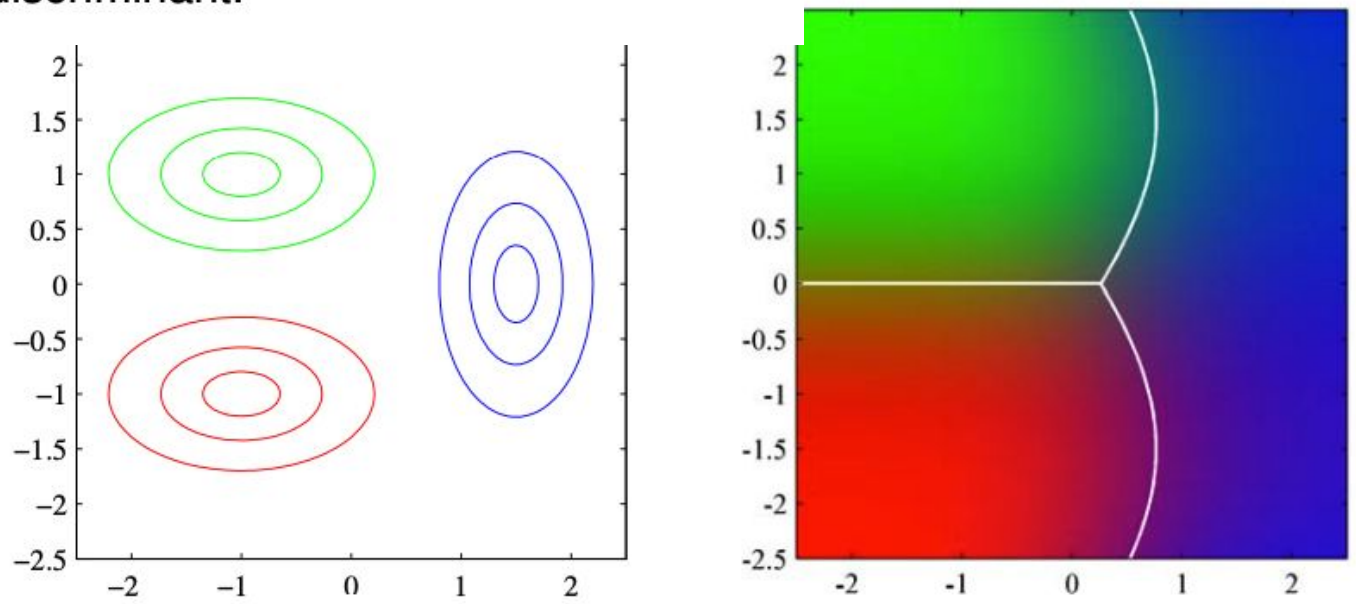


Figure 4.11 The left-hand plot shows the class-conditional densities for three classes each having a Gaussian distribution, coloured red, green, and blue, in which the red and green classes have the same covariance matrix. The right-hand plot shows the corresponding posterior probabilities, in which the RGB colour vector represents the posterior probabilities for the respective three classes. The decision boundaries are also shown. Notice that the boundary between the red and green classes, which have the same covariance matrix, is linear, whereas those between the other pairs of classes are quadratic.

Maximum likelihood estimates

Given the functional form of the class-conditional densities $p(\mathbf{x} \mid \mathcal{C}_k)$, how can we determine the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and the class prior ?

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n \mid \mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$
$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n \mid \mathcal{C}_2) = (1-\pi) \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

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

$$L = \log\left\{\prod_{n=1}^{N}\left[\pi\,\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \times \left[(1-\pi)\,\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n}\right\}$$

$$\sum_{n=1}^{N} t_n\left[\log \pi + \log \mathcal{N}(\quad)\right] + (1-t_n)\left[\log(1-\pi) + \log \mathcal{N}(\quad)\right]$$

The term depending on π is

$$\frac{\partial L}{\partial \pi} = \frac{\partial}{\partial \pi}\sum_{n=1}^{N}\left(t_n \ln \pi + (1-t_n)\ln(1-\pi)\right)$$

$$\Rightarrow \sum_{n=1}^{N} t_n \frac{1}{\pi} + (1-t_n)\frac{-1}{1-\pi}$$

The maximum is at:

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

MLE solution for means and covariance

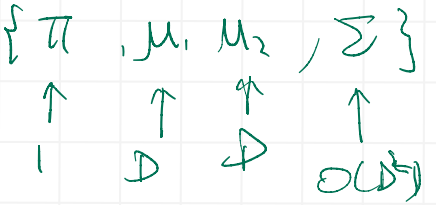
$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n \tag{4.75}$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\mathbf{x}_n \tag{4.76}$$

$$\mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \tag{4.78}$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \tag{4.79}$$

$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}. \tag{4.80}$$



$O(D^2)$ parameters!

Discrete features - Naive Bayes

Left as reading:
Sec 4.2.3 naive bayes
Sec 4.2.4 exponential family

- Assume the input space consists of discrete features, in the simplest case $x_i \in \{0, 1\}$.
- For a D-dimensional input space, a general distribution would be represented by a table with 2^D entries.
- Together with the normalisation constraint, this are $2^D - 1$ independent variables.
- Grows exponentially with the number of features.
- The Naïve Bayes assumption is that, given the class $\mathcal{C}_k$, the features are independent of each other:

$$p(\mathbf{x} \mid \mathcal{C}_k) = \prod_{i=1}^{D} p(x_i \mid \mathcal{C})$$

$$= \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{1 - x_i}$$

Naive Bayes classifier

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i}(1-\mu_{ki})^{1-x_i} \tag{4.81}$$

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

$$a_k(\mathbf{x}) = \sum_{i=1}^{D} \{x_i \ln \mu_{ki} + (1-x_i)\ln(1-\mu_{ki})\} + \ln p(\mathcal{C}_k) \tag{4.82}$$

Linear functions of input x

What we covered so far

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here
(Fisher discriminant)
The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models
Logistic regression

(Laplace approximation, Bayesian logistic regression)

Probabilistic Discriminative Models

- Discriminative training: learn only to discriminate between the classes.
- Maximise a likelihood function defined through the conditional distribution $p(\mathcal{C}_k \mid \mathbf{x})$ directly.
- Typically fewer parameters to be determined.
- As we learn the posteriror $p(\mathcal{C}_k \mid \mathbf{x})$ directly, prediction may be better than with a generative model where the class-conditional density assumptions $p(\mathbf{x} \mid \mathcal{C}_k)$ poorly approximate the true distributions.
- But: discriminative models can not create synthetic data, as $p(\mathbf{x})$ is not modelled.
- As an aside: *certain theoretical analyses show that generative models converge faster to their — albeit worse — asymptotic classification performance and are superior in some regimes.*

Fixed basis functions

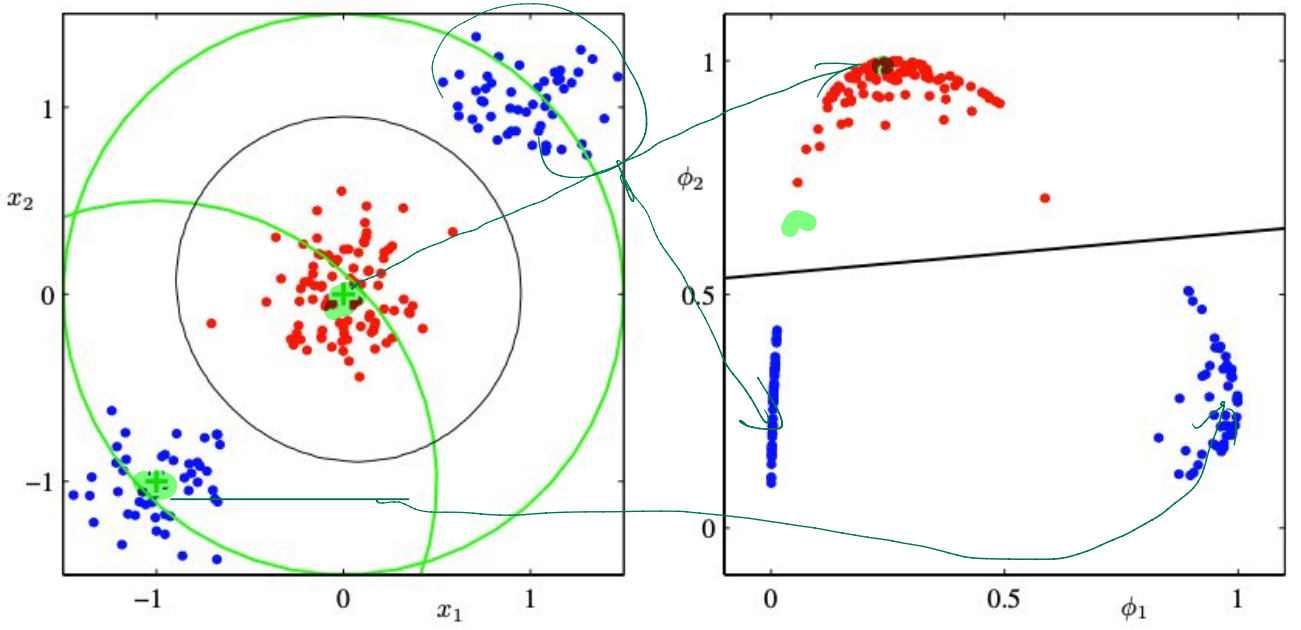
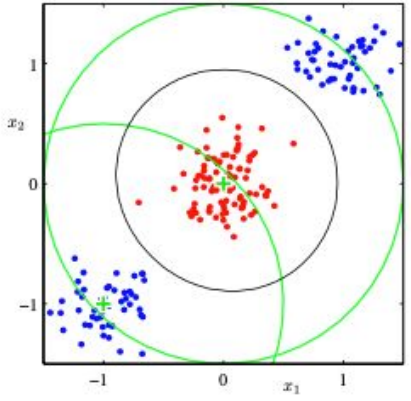
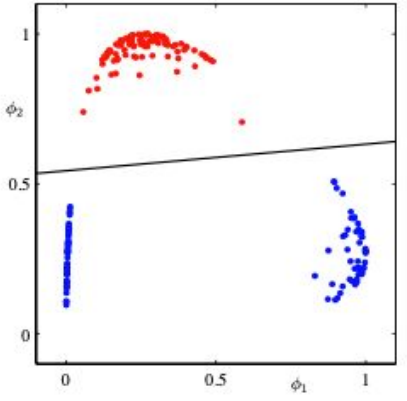


Figure 4.12 Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space (x_1, x_2) together with data points from two classes labelled red and blue. Two 'Gaussian' basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ are defined in this space with centres shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 4.3.2. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.

Original input vs feature space

- Linear decision boundaries in the feature space generally correspond to nonlinear boundaries in the input space.
- Classes which are NOT linearly separable in the input space may become linearly separable in the feature space:





- If classes overlap in input space, they will also overlap in feature space — nonlinear features $\phi(\mathbf{x})$ can not remove the overlap; but they may increase it.

Logistic regression

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right) \tag{4.87}$$

$\sigma(a)$

D

→ ignore $P(C_1)$ ← imbalanced

ignore slope of $P(x|C_1)$

The name came from statistics. It's a model for classification rather than regression!

Compare to generative model of one Gaussian per class with shared variance

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0) \tag{4.65}$$

model $P(x|C_1)$ ~ Gaussian

$P(C_1)$ ~ number

Same expression for posterior estimates, but linear number of parameters rather than quadratic.

↑

D in millions / billions. sparse.

Maximum likelihood for logistic regression

- Determine the parameter via maximum likelihood for data $(\boldsymbol{\phi}_n, t_n)$, $n = 1, \ldots, N$, where $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$. The class membership is coded as $t_n \in \{0, 1\}$.
- Likelihood function

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

$t_n = 1$ OR $t_n = 0$

where $y_n = p(\mathcal{C}_1 \mid \boldsymbol{\phi}_n)$.

- Error function : negative log likelihood resulting in the cross-entropy error function

$$y_n = p(\mathcal{C}_1 \mid \boldsymbol{\phi}_n) = \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)$$

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

← assumes every data point counts as the same.

Gradient of Logistic regression

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

$$= -\sum_{n=1}^{N} \{t_n \ln \sigma(\mathbf{w}^T\phi_n) + (1-t_n)\ln(1-\sigma(\mathbf{w}^T\phi_n))\}$$

$$\nabla \ln \sigma(\mathbf{w}^T\phi_n) =$$

$$\nabla \ln(1-\sigma(\mathbf{w}^T\phi_n)) =$$

$$\nabla E(\mathbf{w}) =$$

$$y_n = p(C_1 \mid \phi_n) = \sigma(\mathbf{w}^T\phi_n)$$

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

Gradient of Logistic regression

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

$$= -\sum_{n=1}^{N} \{ t_n \ln \sigma(w^T\phi_n) + (1-t_n)\ln(1-\sigma(w^T\phi_n)) \}$$

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

$$\nabla_w \ln \sigma(w^T\phi_n) = \frac{1}{\sigma(w^T\phi_n)} \sigma(w^T\phi_n)(1-\sigma(w^T\phi_n))\phi_n$$

$$\nabla \ln(1-\sigma(w^T\phi_n)) = \frac{1}{1-\sigma(w^T\phi_n)} [-\sigma(w^T\phi_n)(1-\sigma(w^T\phi_n))\phi_n]$$

$$= -\sigma(w^T\phi_n)\phi_n$$

$$\nabla E(w) = -\sum_{n=1}^{N} \{ t_n (1-\sigma(w^T\phi_n))\phi_n + (1-t_n)(-\sigma(w^T\phi_n)\phi_n) \}$$

(t_n: scalar)

$$= -\sum_{n=1}^{N} \{ t_n\phi_n - t_n\sigma(w^T\phi_n)\phi_n - \sigma(w^T\phi_n)\phi_n + t_n\sigma(w^T\phi_n)\phi_n \}$$

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

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

Stochastic gradient

$$\nabla E_n(w) = \underbrace{(y_n - t_n)}_{\text{error}}\phi_n$$

$\nabla_w E(w) = 0$

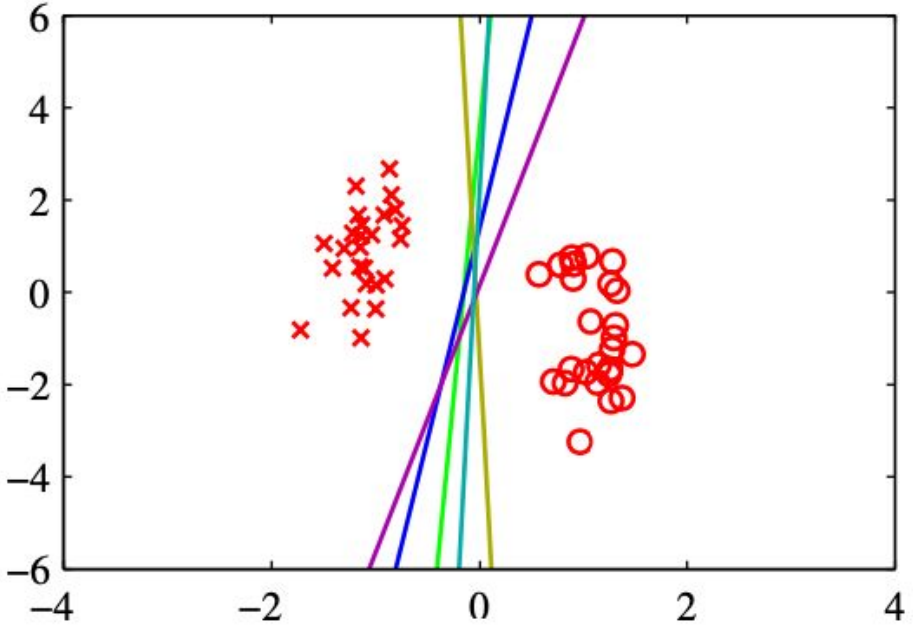
→ maybe a local maximum

$$w^{(\tau)} = w^{(\tau-1)} + \eta \nabla E(w)$$

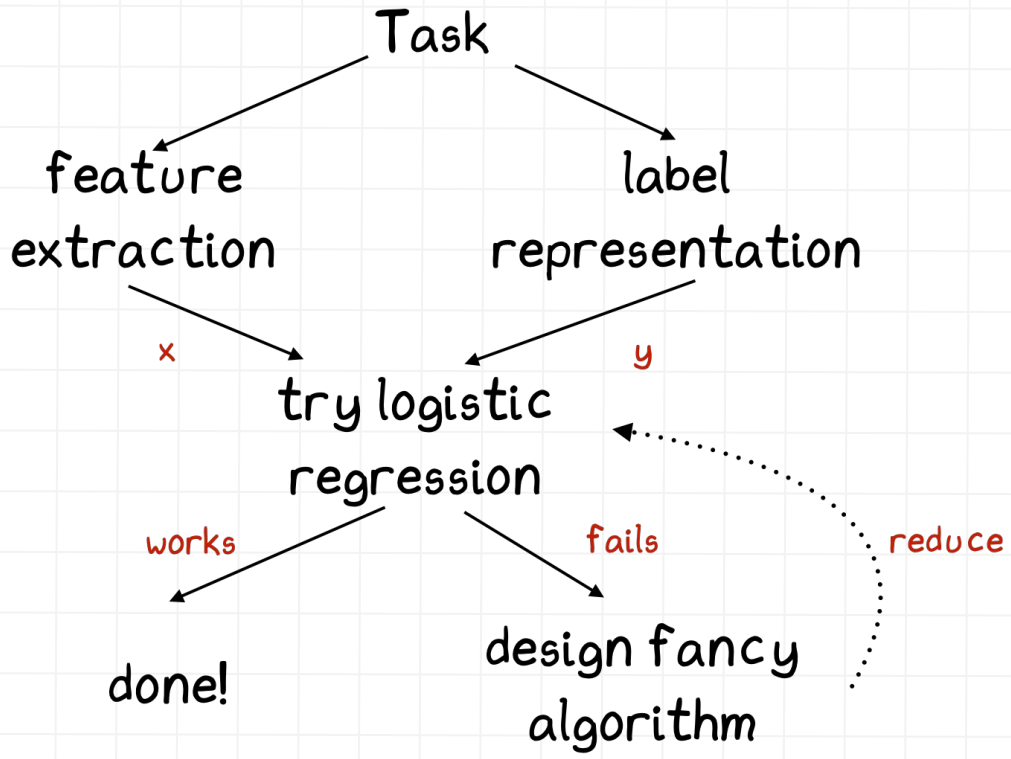
A critique of logistic regression

$p(C_1|x) = \sigma(w^T x)$

PRML Fig 10.13



the machine learning “master algorithm*”



* Not included: optimisation, inference, asymptotic bounds, evaluation setting

Courtesy of Aditya Menon (Google Research)

Summary of Linear Models for classification

Classification problems

A glimpse of decision theory (Sec 1.5)

Discriminant functions - why least squares doesn't work here
The perceptron algorithm

Probabilistic generative models - origin of the logistic function

Probabilistic discriminative models
Logistic regression

(Laplace approximation, Bayesian logistic regression) – later in the semester

Origin of the logistic function, logistic regression and how it connects to perceptron

Laplace approximation

Other linear discriminants (e.g. Fisher)

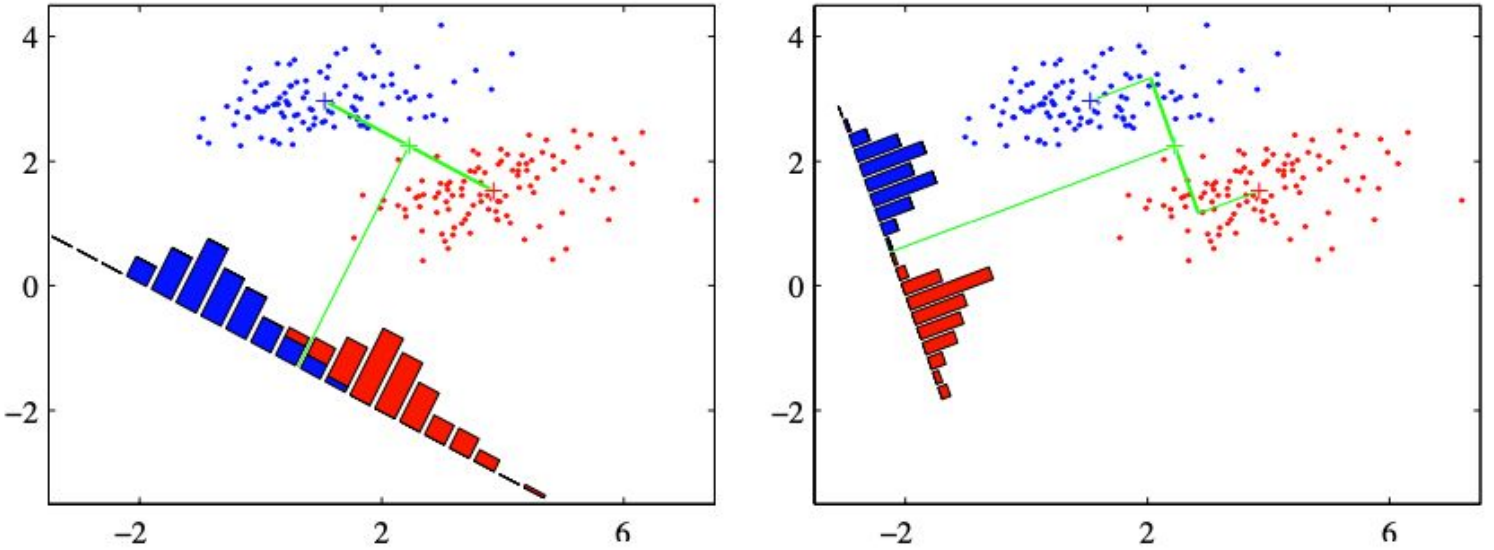


Figure 4.6 The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}} \tag{4.26}$$

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1). \tag{4.30}$$