Single-Layer Network & Probabilities

Consider a two-class classification problem in which the class-conditional densities are given by Gaussian distributions

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

with equal cov. matrices $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}$, where $\boldsymbol{\mu}_k$ is a d-dimensional mean vector and $\mathbf{\Sigma}$ is a $d \times d$ covariance matrix. $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$. Note, $\mathbf{\Sigma}$ must be invertible (non-singular). Location is determined by $\boldsymbol{\mu}$ whereas shape is determined by $\mathbf{\Sigma}$.

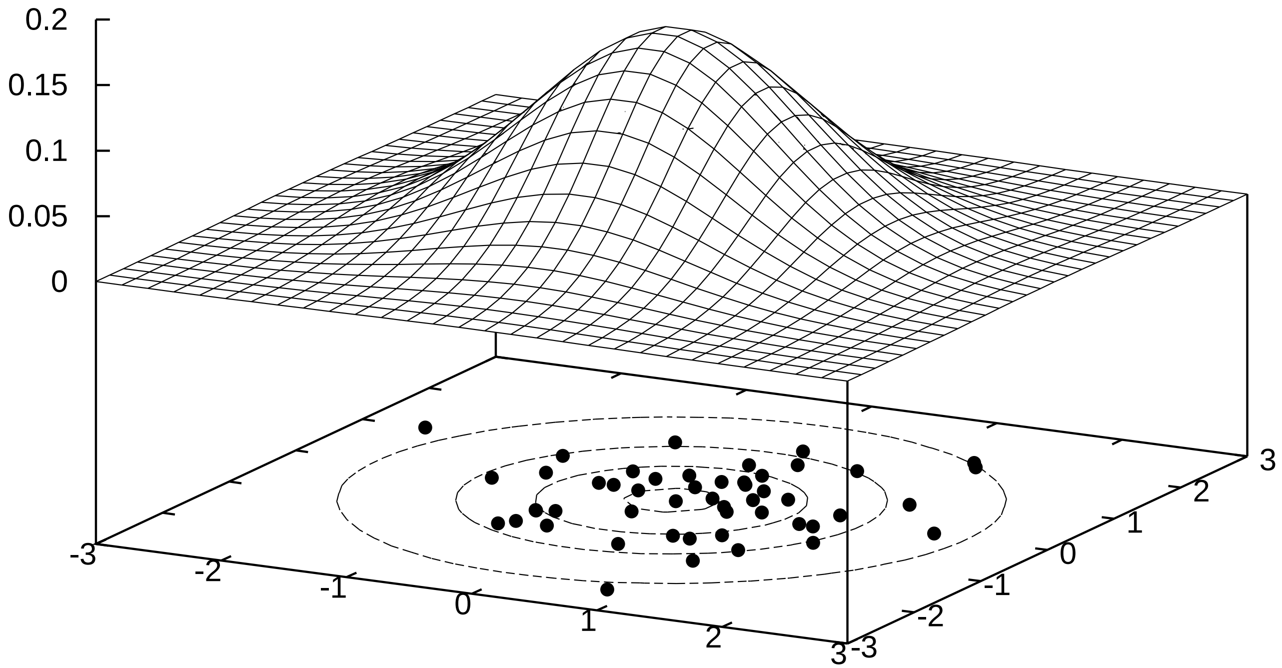
We will show that posterior probability $p(\mathcal{C}_1|\mathbf{x})$ can be expressed as single layer network output:

$$g(\mathbf{w}^T\mathbf{x} + w_0), \quad \text{where } g(a) = \frac{1}{1+\exp(-a)}$$

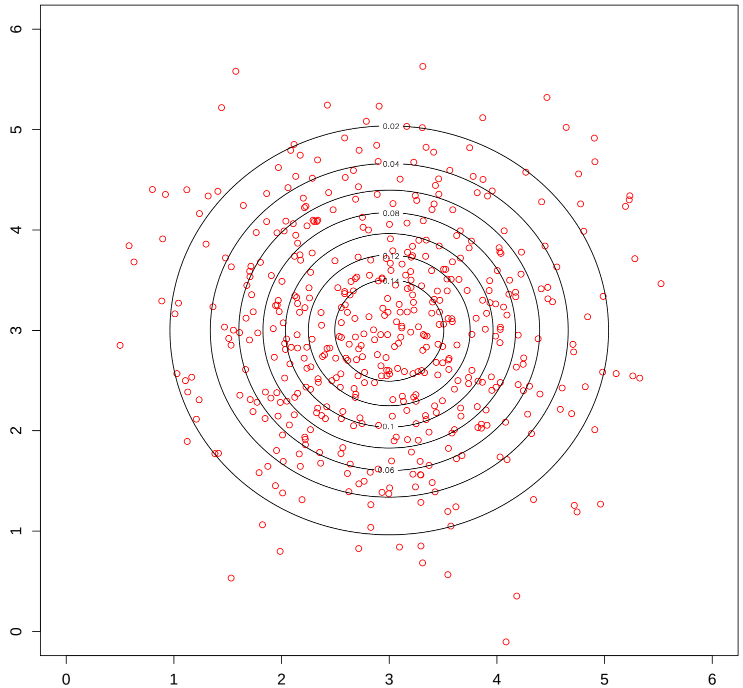
Two-dim. Gaussian Distribution

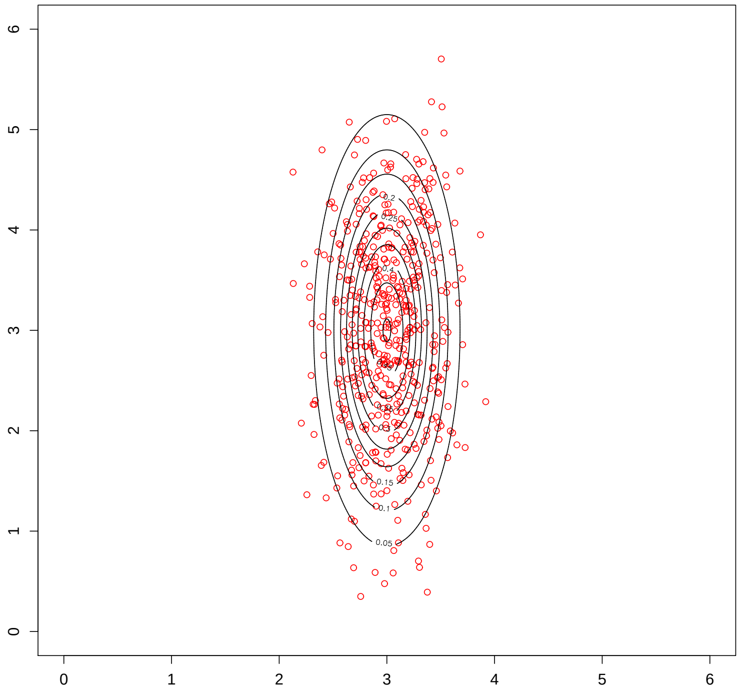
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Two-dim. Gaussian Distribution

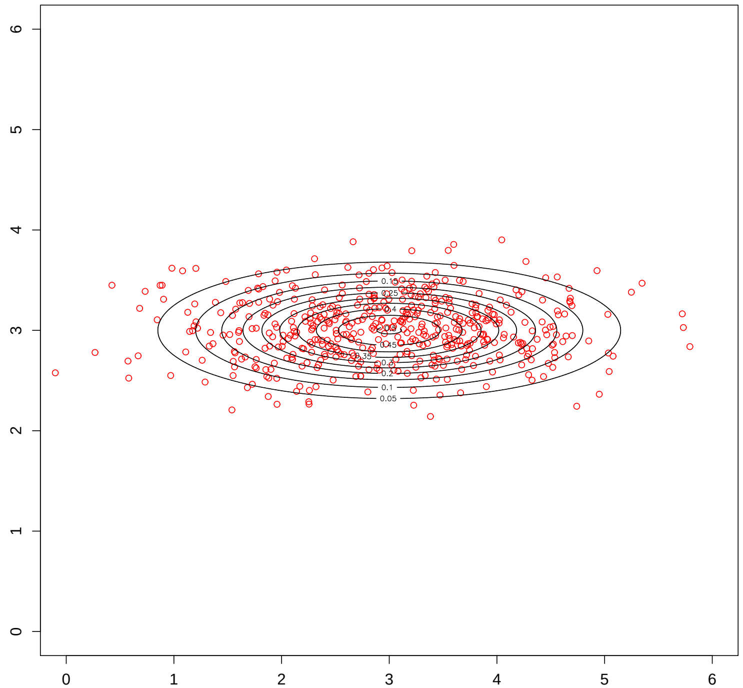


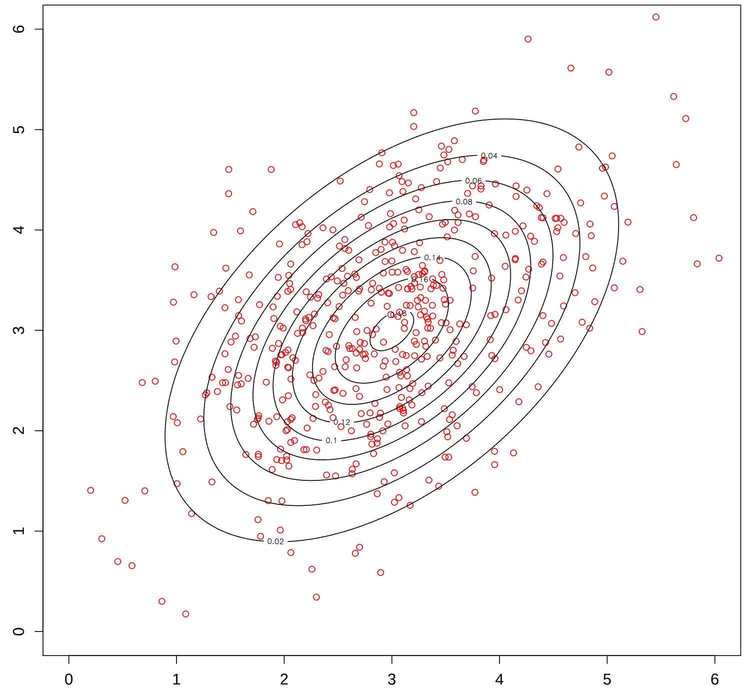


$$\boldsymbol{\mu} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}$$

Two-dim. Gaussian Distribution (cont.)





$$\boldsymbol{\mu} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

Bayes' Theorem and Posterior Probability

Joint probability density function of finding a pattern that has feature value $\mathbf{x}$ *and* is in class $\mathcal{C}_i$ can be written as:

$$p(\mathbf{x}, \mathcal{C}_i) = p(\mathcal{C}_i|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathcal{C}_i)p(\mathcal{C}_i) = p(\mathcal{C}_i, \mathbf{x})$$

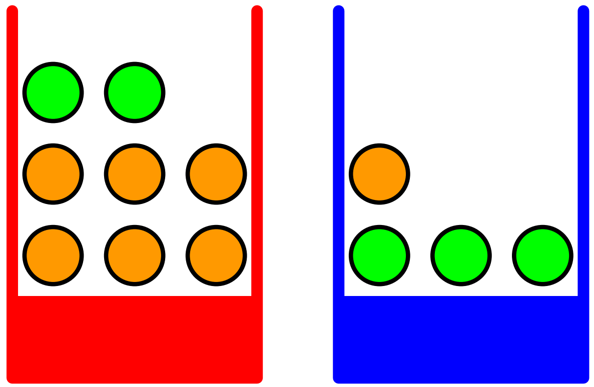
Bayes' theorem to compute the *posterior probability*:

$$p(\mathcal{C}_i|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_i)\, p(\mathcal{C}_i)}{p(\mathbf{x})}, \qquad p(\mathbf{x}) = \sum_i p(\mathbf{x}|\mathcal{C}_i)\, p(\mathcal{C}_i)$$

The posterior probability for class $\mathcal{C}_1$ can be written as:

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

Bishop's Bayes Example



The red box contains 6 oranges and 2 apples, the blue box contains 1 orange and 3 apples. Suppose we pick the red box $40\,\%$ of the time and the blue box $60\,\%$ of the time. Thanks to Bayes' theorem we can answer questions such as:

- What is the overall probability that we pick an apple?
- Given that we have chosen an orange, what is the probability that the box we chose was the blue one?

Bishop's Bayes Example (cont.)

For the sake of clarity let us introduce random variables B for box and F for fruit. B can take one of the two possibilities $B = r$ (for red) and $B = b$ (for blue); and $F = o$ (for orange) and $F = a$ (for apple).

The prior probability of selecting the red box is

$$p(B = r) = \frac{4}{10}$$

and of selecting the blue box

$$p(B = b) = \frac{6}{10}$$

Bishop's Bayes Example (cont.)

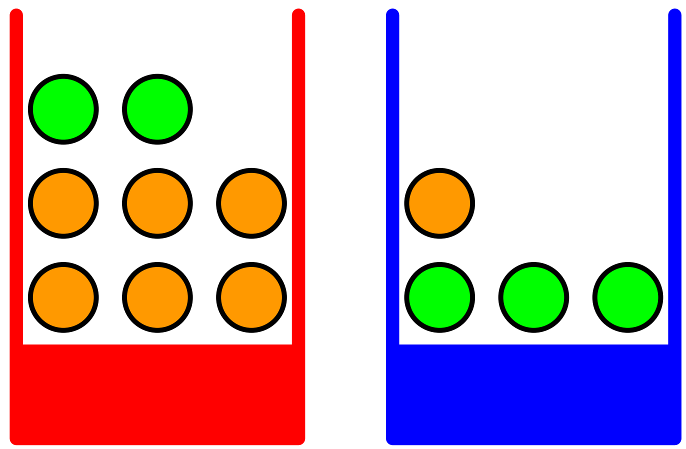
From given information we can write out all four conditional probabilities of given the selected box and picking the type of fruit.

$$p(F = a|B = r) = \frac{1}{4}$$

$$p(F = o|B = r) = \frac{3}{4}$$

$$p(F = a|B = b) = \frac{3}{4}$$

$$p(F = o|B = b) = \frac{1}{4}$$



Bishop's Bayes Example (cont.)

Back to our question: What is the overall probability that we pick an apple?

$$\begin{aligned} p(F = a) &= p(F = a|B = r)p(B = r) + p(F = a|B = b)p(B = b) \\ &= \frac{1}{4} \cdot \frac{4}{10} + \frac{3}{4} \cdot \frac{6}{10} = \frac{11}{20} \end{aligned}$$

from this it follows that $p(F = o) = 1 - \frac{11}{20} = \frac{9}{20}$. Although there are more oranges in total, picking an apple is more likely.

Back to our second question: Given that we have chosen an orange, what is the probability that the box we chose was the blue one, that is $p(B = b|F = o)$?

Bishop's Bayes Example (cont.)

$$p(B=b|F=o) = \frac{p(F=o|B=b)p(B=b)}{p(F=o)} = \frac{1}{4}\cdot\frac{6}{10}\cdot\frac{20}{9} = \frac{1}{3}$$

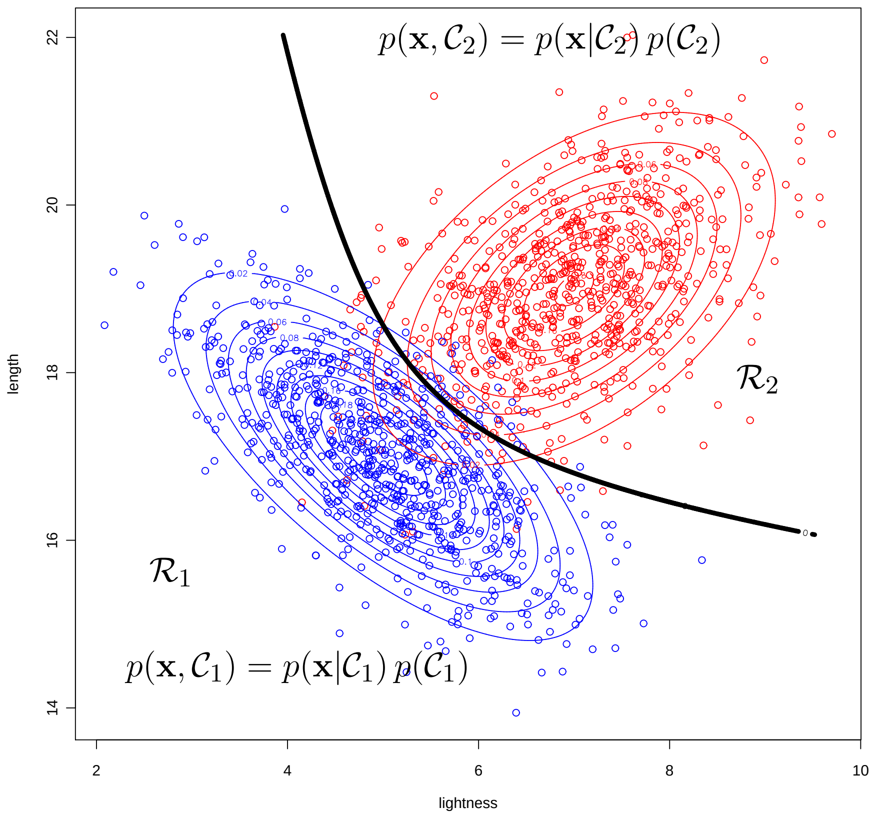
and that we chose the red box:

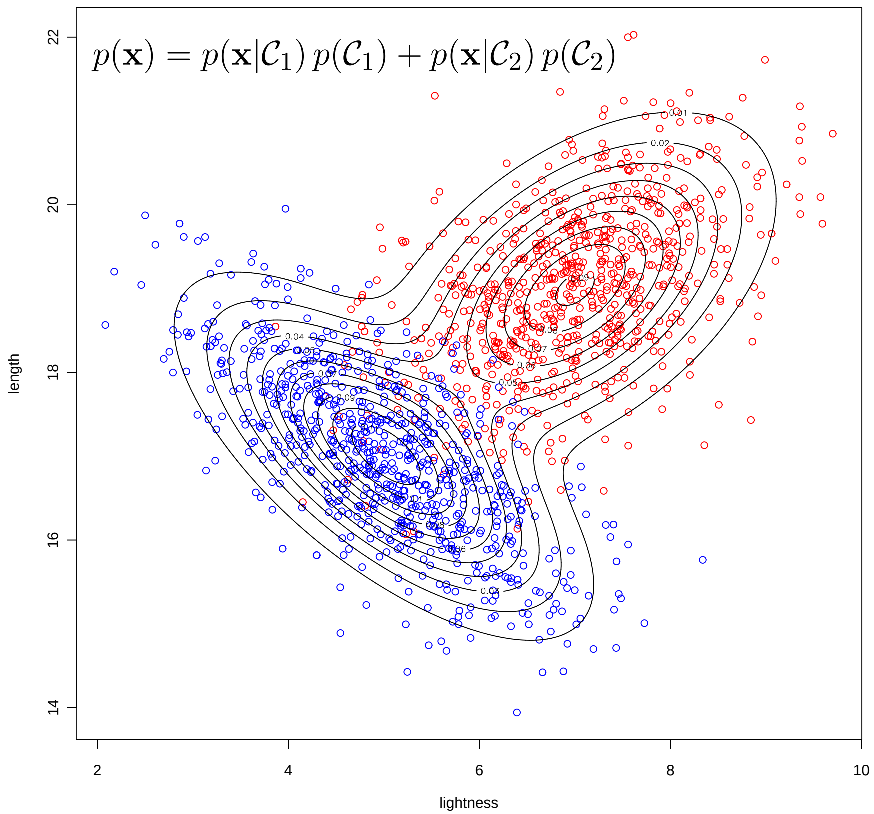
$$p(B=r|F=o) = \frac{p(F=o|B=r)p(B=r)}{p(F=o)} = \frac{3}{4}\cdot\frac{4}{10}\cdot\frac{20}{9} = \frac{2}{3}$$

The denominator in Bayes' theorem ensures that posterior probability summed over all classes $\mathcal{C}_i$ gives 1. In this examples, $1 = p(B=b|F=o) + p(B=r|F=o)$.

$$p(\mathcal{C}_i|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_i)\,p(\mathcal{C}_i)}{p(\mathbf{x})}, \qquad p(\mathbf{x}) = \sum_i p(\mathbf{x}|\mathcal{C}_i)\,p(\mathcal{C}_i)$$

Visualized Fish Posterior Probability





Decide $\mathcal{C}_1$ if $p(\mathbf{x}|\mathcal{C}_1)\, p(\mathcal{C}_1) > p(\mathbf{x}|\mathcal{C}_2)\, p(\mathcal{C}_2)$, otherwise decide $\mathcal{C}_2$. This decision rule will divide the input space in regions $\mathcal{R}_i$ such that all points in $\mathcal{R}_i$ are assigned to class $\mathcal{C}_i$.

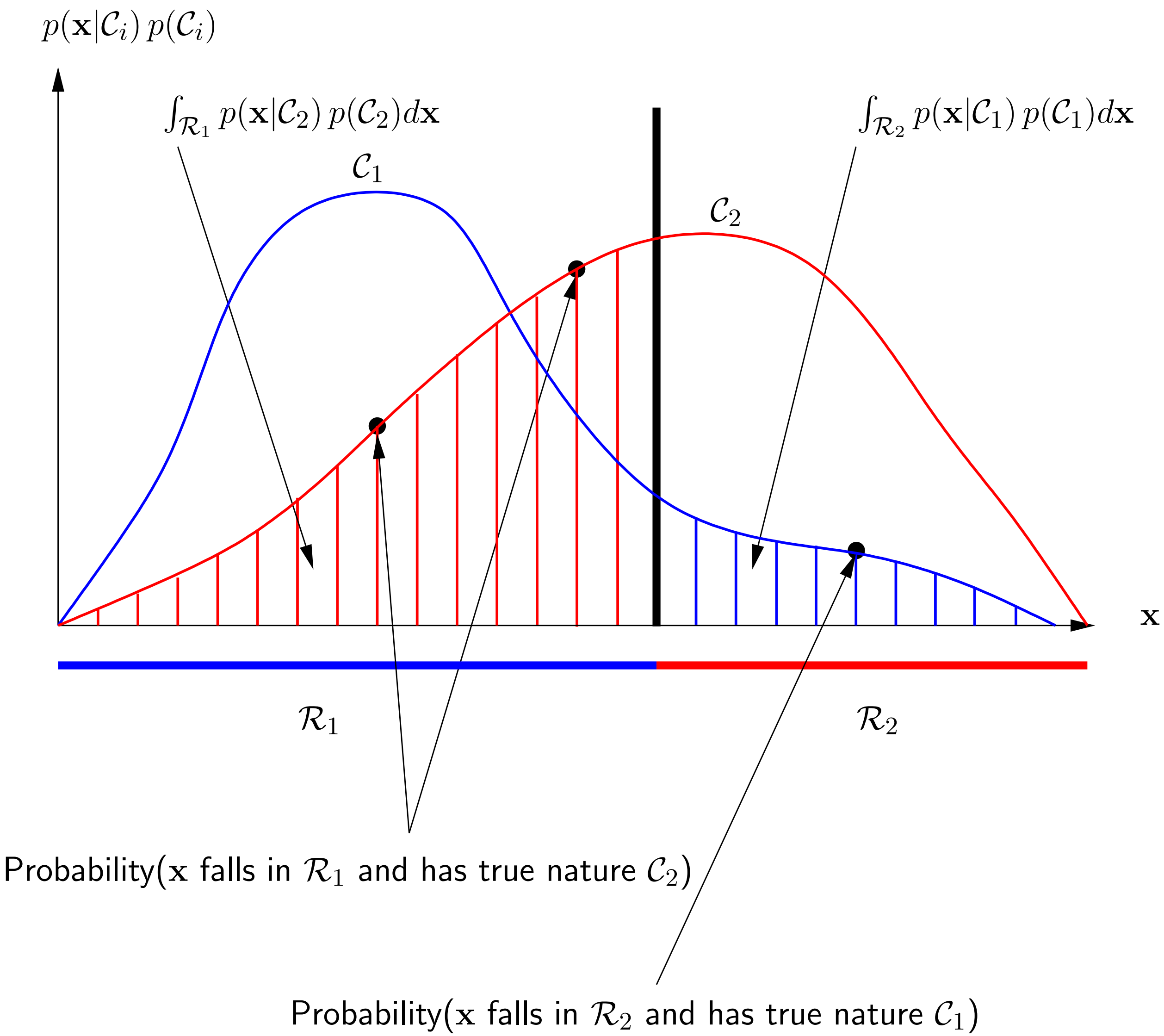
Bayesian Decision Theory

- How to make an optimal decision given the appropriate probabilities?
- Minimize the error of assigning $\mathbf{x}$ to the wrong class.
- Intuitively we would choose the class having the higher posterior probability.

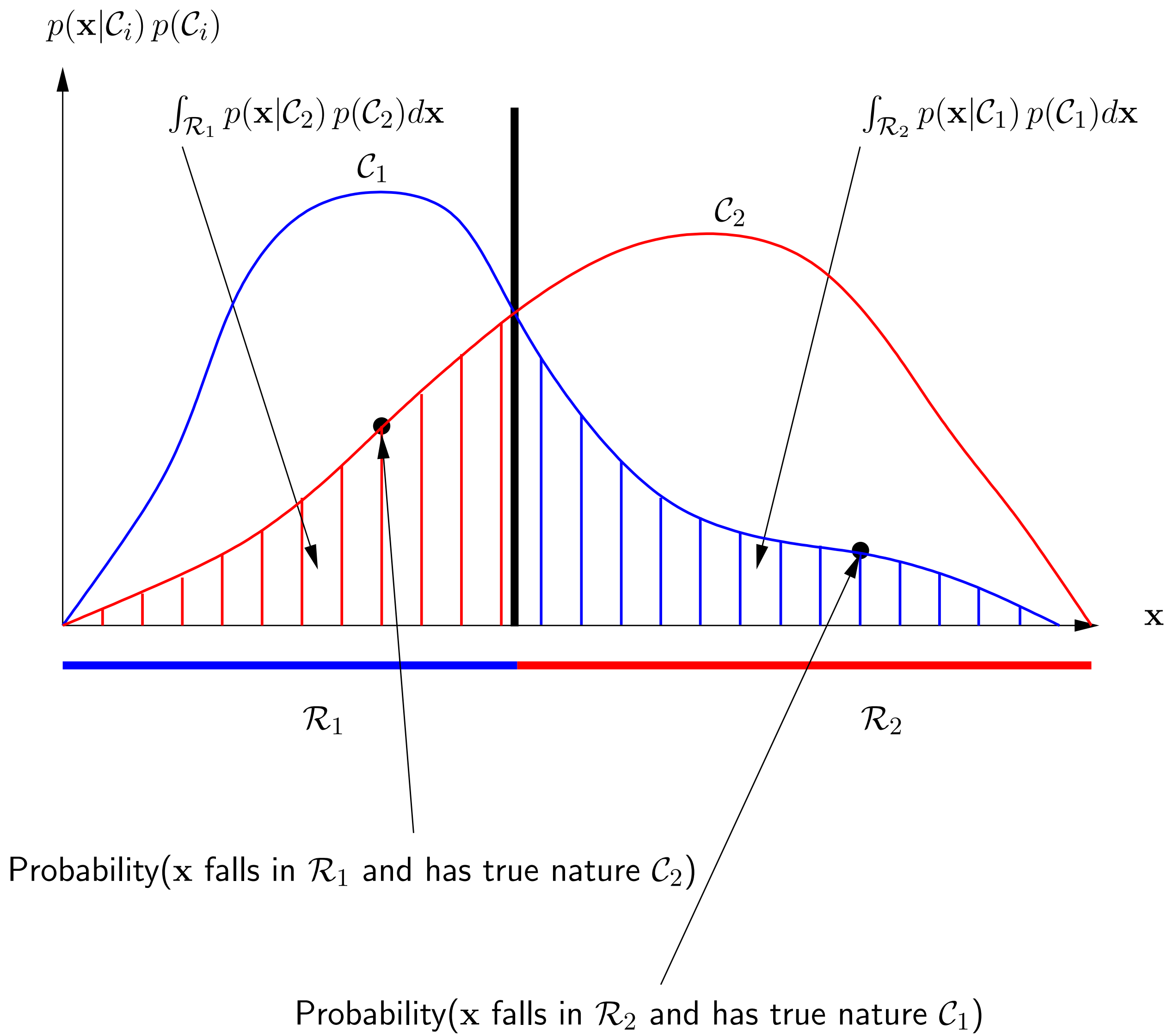
An error occurs when $\mathbf{x}$ belonging to class $\mathcal{C}_1$ is assigned to class $\mathcal{C}_2$ or vice versa. The probability of this occurring is given by:

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

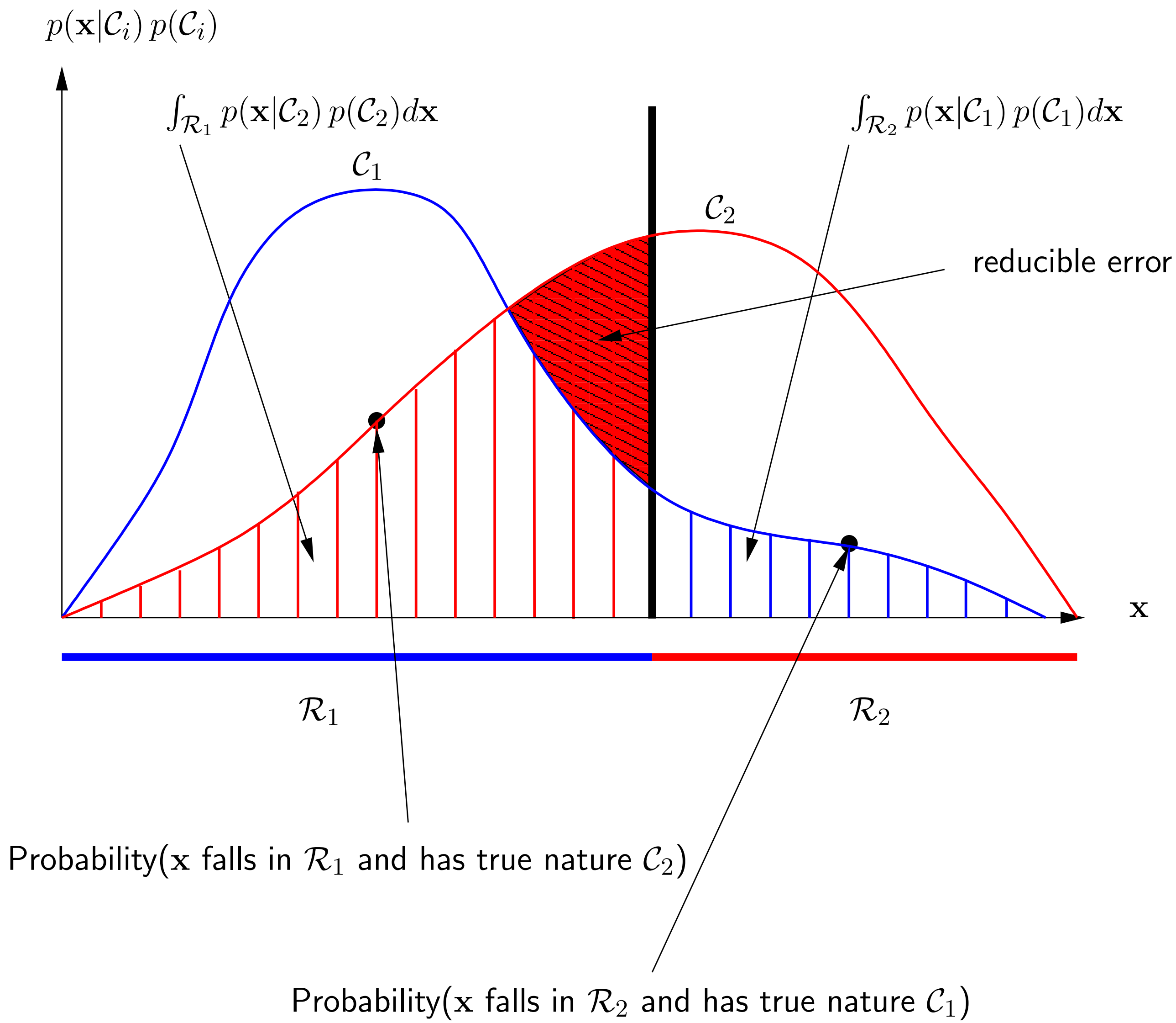
Error Probabilities in Bay. Decision Theory



Error Probabilities in Bay. Decision Theory



Error Probabilities in Bay. Decision Theory



Single-Layer Network & Probabilities (cont.)

Posterior probability for class $\mathcal{C}_1$ can be written as

$$
\begin{aligned}
p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\
&= \frac{1}{1 + \exp(-a)} = g(a)
\end{aligned}
$$

where

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

and $g(a)$ is our known sigmoid logistic function. Observe:

$$\frac{1}{1 + \exp\left(-\ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}\right)} = \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} = p(\mathcal{C}_1|\mathbf{x}) \qquad \boxed{\frac{1}{1+\frac{a}{b}} = \frac{b}{b+a}}$$

Single-Layer Network & Probabilities (cont.)

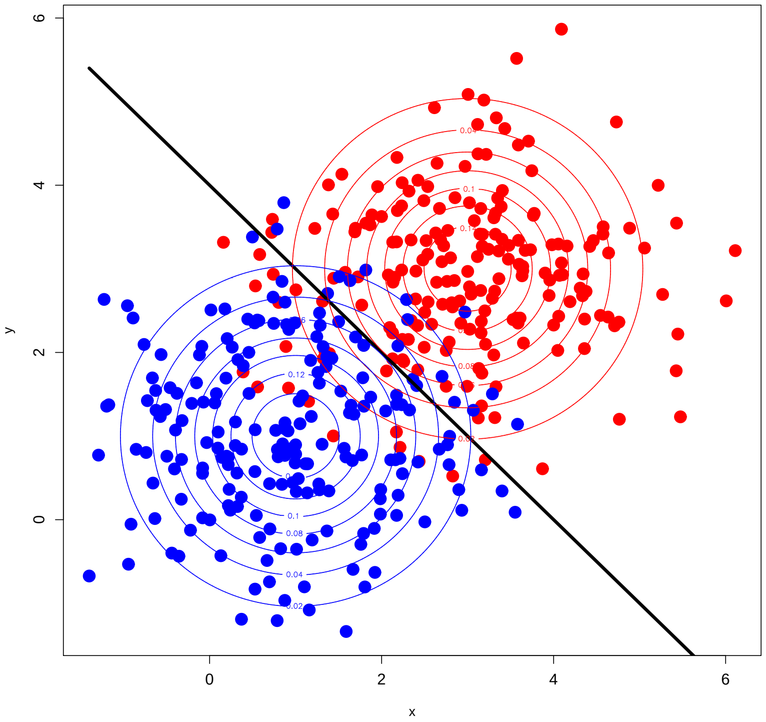
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\} \quad (1)$$

Substitute (1) in $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$ gives $a = \mathbf{w}^T\mathbf{x} + w_0$ where

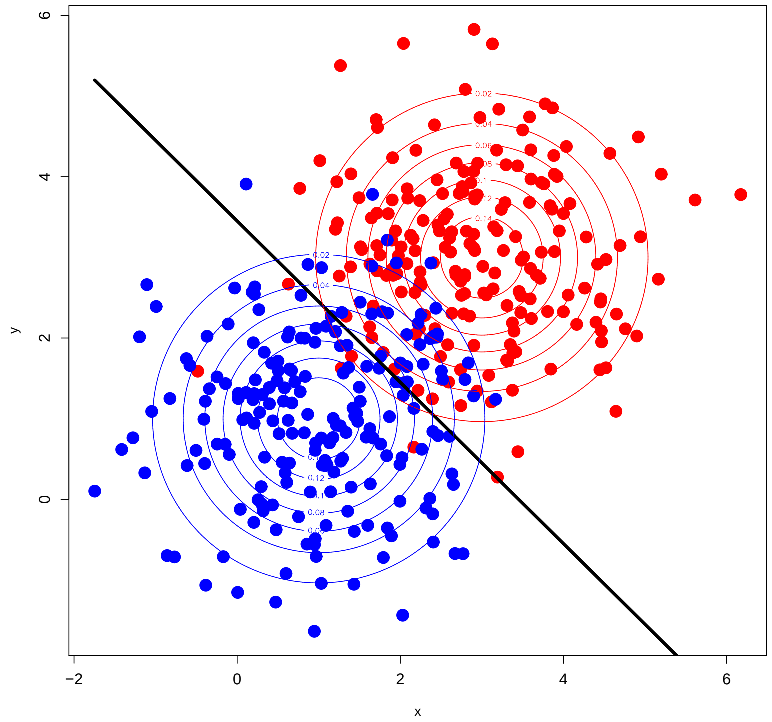
$$\begin{aligned} \mathbf{w} &= \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

Network output $g(\mathbf{w}^T\mathbf{x} + w_0) = p(\mathcal{C}_1|\mathbf{x})$ gives posterior probability. Observe that quadratic terms in $\mathbf{x}$ from $\exp\{\cdot\}$ have cancelled. This leads to a linear function of $\mathbf{x}$ and gives decision boundaries that are linear.

Single-Layer Network & Priors



$p(\mathcal{C}_1) = p(\mathcal{C}_2)$



$p(\mathcal{C}_1) = 3/4,\ p(\mathcal{C}_2) = 1/4$

Perceptron

Note, so far we have not seen a method for finding the weight vector $\mathbf{w}$ to obtain a linearly separation of the training set.

Let $g(a)$ be (sign) activation function

$$g(a) = \begin{cases} -1 & \text{if } a < 0 \\ +1 & \text{if } a \geq 0 \end{cases}$$

and decision function

$$y = g\left(\sum_{i=0}^{d} w_i x_i\right) = g(\mathbf{w}^T \mathbf{x}) \rightarrow \{-1, +1\}$$

Note: x_0 is set to $+1$, that is, $\mathbf{x} = (1, x_1, \ldots, x_d)$. Training pattern consists of $(\mathbf{x}, t) \in \mathbb{R}^d \times \{-1, +1\}$

Training the Perceptron

Training problem for the Perceptron is to find weight vector such that:

$\mathbf{w}^T\mathbf{x} \geq 0$ for every input pattern $\mathbf{x}$ belonging to class $\{+1\}$

$\mathbf{w}^T\mathbf{x} < 0$ for every input pattern $\mathbf{x}$ belonging to class $\{-1\}$

where the weights are updated whenever training example is missclassified, that is,

- $\mathbf{w}^{\text{new}} = \mathbf{w} - \eta\mathbf{x}$, if $\mathbf{w}^T\mathbf{x} \geq 0$ and $t \in \{-1\}$
- $\mathbf{w}^{\text{new}} = \mathbf{w} + \eta\mathbf{x}$, if $\mathbf{w}^T\mathbf{x} < 0$ and $t \in \{+1\}$
- no correction if correctly classified

Weight correction learning rule can be summarized as:

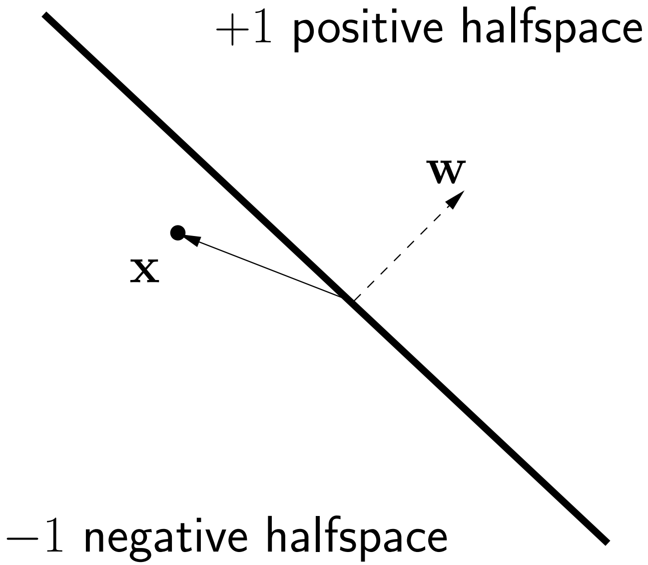
$$\mathbf{w}^{\text{new}} = \mathbf{w} + \eta\mathbf{x}\,t \qquad \text{if } \mathbf{x} \text{ is missclassified}$$

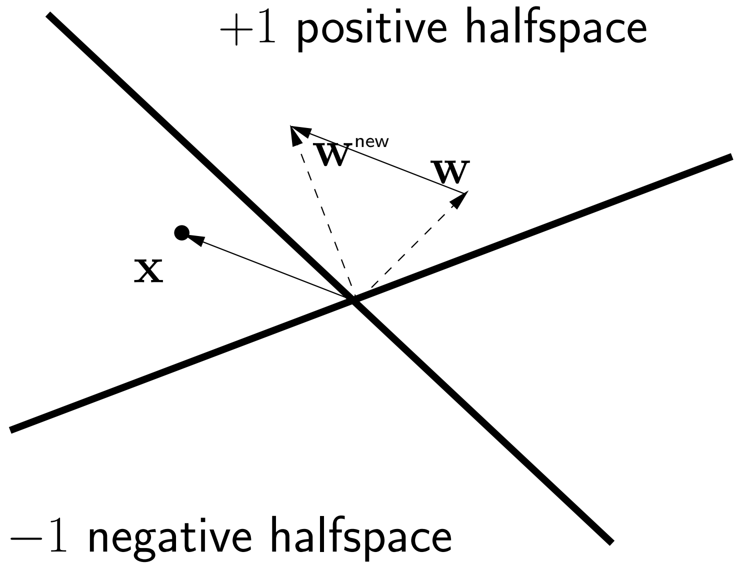
Perceptron Learning Algorithm

input : $(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N) \in \mathbb{R}^d \times \{-1, +1\}, \eta \in \mathbb{R}_+$, max.epoch $\in \mathbb{N}$
output: $\mathbf{w}$
begin
    Randomly initialize $\mathbf{w}$
    epoch $\leftarrow 0$
    repeat
        for $i \leftarrow 1$ **to** N **do**
            if $t_i(\mathbf{w}^T\mathbf{x}_i) \leq 0$ **then**
                $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{x}_i t_i$
        epoch $\leftarrow$ epoch $+ 1$
    until (*epoch* $=$ *max.epoch*) *or* (*no change in* $\mathbf{w}$)
    return $\mathbf{w}$
end

Training the Perceptron (cont.)

Geometrical explanation: If $\mathbf{x}$ belongs to $\{+1\}$ and $\mathbf{w}^T\mathbf{x} < 0 \Rightarrow$ angle between $\mathbf{x}$ and $\mathbf{w}$ is greater than 90°, rotate $\mathbf{w}$ in direction of $\mathbf{x}$ to bring missclassified $\mathbf{x}$ into the positive half space defined by $\mathbf{w}$. Same idea if $\mathbf{x}$ belongs to $\{-1\}$ and $\mathbf{w}^T\mathbf{x} \geq 0$.





Perceptron Error Reduction

Recall: missclassifcation results in:

$$\mathbf{w}^{\text{new}} = \mathbf{w} + \eta \mathbf{x}\, t,$$

this reduces the error since

$$\begin{aligned} -\mathbf{w}^{\text{new}}(\mathbf{x}\, t)^T &= -\mathbf{w}(\mathbf{x}\, t)^T - \underbrace{\eta}_{>0} \underbrace{(\mathbf{x}\, t)(\mathbf{x}\, t)^T}_{\|\mathbf{x}t\|^2>0} \\ &< -\mathbf{w}^T\mathbf{x}t \end{aligned}$$

How often one has to cycle through the patterns in the training set?

- A finite number of steps?

Perceptron Convergence Theorem

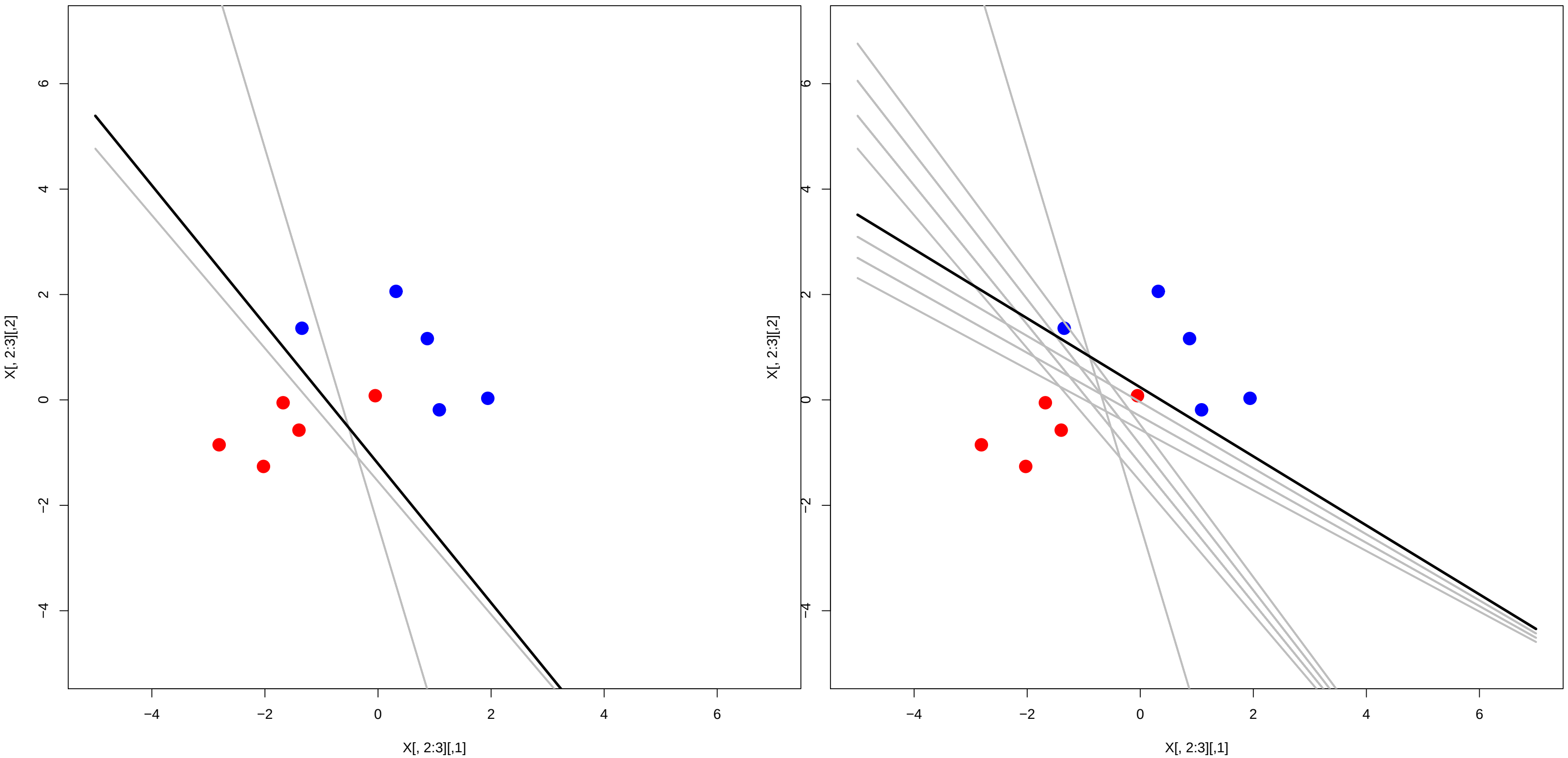
Proposition 1 *Given a finite and linearly separable training set. The perceptron converges after some finite steps.*

Proof: (see chalkboard)

Perceptron Algorithm (R-code)

```
###################################################
perceptron <- function(w,X,t,eta,max.epoch) {
###################################################
  N <- nrow(X)/2;
  epoch <- 0;
  repeat {
    w.old <- w;
    for (i in 1:(2*N)) {
      if ( t[i]*y(X[i,],w) <= 0 )
        w <- w + eta * t[i] * X[i,];
    }
    epoch <- epoch + 1;
    if ( identical(w.old,w) || epoch = max.epoch ) {
      break; # terminate if no change in weights or max.epoch reached
    }
  }
  return (w);
}
```

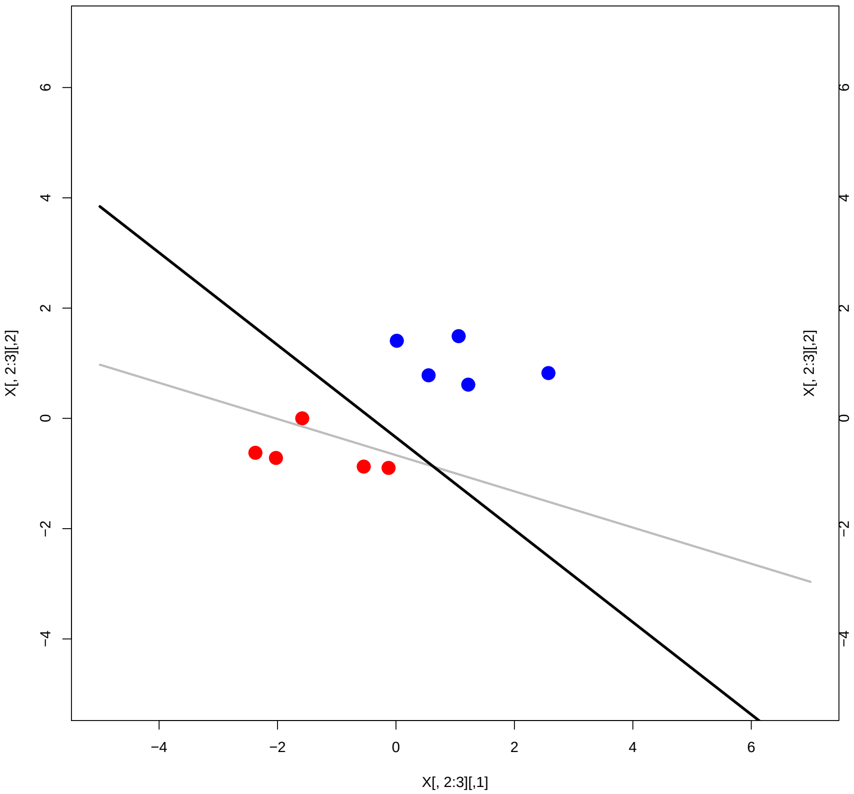
Perceptron Algorithm Visualization

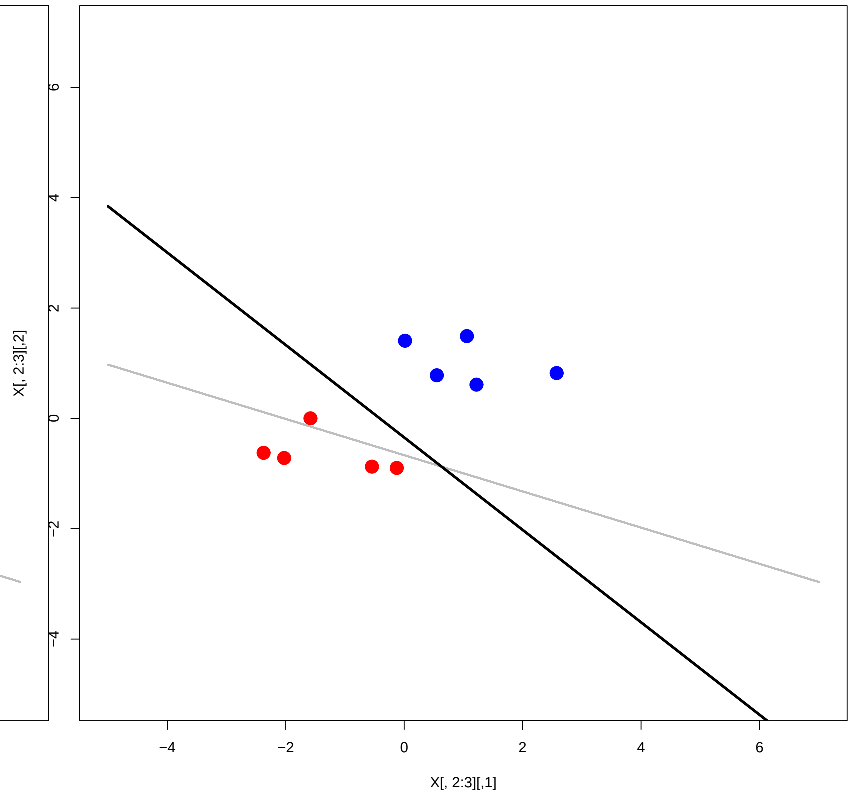


One epoch

terminate if no change in $\mathbf{w}$

Perceptron Algorithm Visualization





One epoch

terminate if no change in $\mathbf{w}$

Least Mean Square

Let us consider the weight correction in terms of an error function $E^{(i)} = \frac{1}{2}(\underbrace{y^{(i)}}_{g(\mathbf{w}^T\mathbf{x})} - t^{(i)})^2$, where $g(\cdot)$ is a differentiable function. Apply gradient descent rule

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \frac{\partial E^{(i)}}{\partial \mathbf{w}}, \text{ where } \quad \frac{\partial E^{(i)}}{\partial \mathbf{w}} = \underbrace{(y^{(i)} - t^{(i)})}_{\delta^{(i)}} \mathbf{x}^{(i)}$$

gives change in weights

$$\Delta \mathbf{w} = -\eta \delta^{(i)} \mathbf{x}^{(i)} = -\eta \frac{\partial E^{(i)}}{\partial \mathbf{w}}$$

Delta rule $\equiv$ {Adaline rule, Widrow-Hoff rule, Least Mean Square (LMS) }

Least Mean Square

Note, if we choose $g(a) = a$ to be the linear activation function $\mathbf{w}^T\mathbf{x}$, then there exists a closed analytical solution (pseudo-inverse solution).

Let $g(a)$ be a differentiable non-linear activation function, where $a = \mathbf{w}^T\mathbf{x}$.

$$\frac{\partial E^{(i)}}{\partial \mathbf{w}} = \delta^{(i)}\mathbf{x}^{(i)}, \text{ where } \quad \delta^{(i)} = g'(a)(y^{(i)} - t^{(i)})$$

gives change in weights

$$\Delta \mathbf{w} = -\eta\delta^{(i)}\mathbf{x}^{(i)} = -\eta\frac{\partial E^{(i)}}{\partial \mathbf{w}}$$

LMS Online/Batch Learning

Online learning:

- Update weight $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \frac{\partial E^{(i)}}{\partial \mathbf{w}}$ (pattern by pattern).

This type of online learning is also called *stochastic gradient descent*, it is an approximation of the true gradient.

Batch learning:

- Update weight $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \sum_{i=1}^{N} \frac{\partial E^{(i)}}{\partial \mathbf{w}}$ by computing derivatives for each pattern separately and then sum over all patterns.

Minimum Squared Error and Pseudoinverse

Recall that we want to minimize the squared error

$$E(\mathbf{w}) = \sum_{i=1}^{N} \frac{1}{2}\left(y^{(i)} - t^{(i)}\right)^2 \qquad \text{where } y^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)}$$

Let $\mathbf{X}$ be the $N \times \widetilde{d}$ matrix where $\widetilde{d} = d + 1$ and ith row denotes training pattern $\mathbf{x}^{(i)\,T}$, $\mathbf{w}$ is weight vector, $\mathbf{t}$ class label vector.

$$\begin{pmatrix} x_{10} & x_{11} & \cdots & x_{1d} \\ x_{20} & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_{N0} & x_{N1} & \cdots & x_{Nd} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix} = \begin{pmatrix} t_0 \\ t_1 \\ \vdots \\ \vdots \\ t_N \end{pmatrix} \qquad \mathbf{X}\mathbf{w} = \mathbf{t}$$

MSE and Pseudoinverse (cont.)

Problem: find weight vector $\mathbf{w}$, that is, solve $\mathbf{X}\mathbf{w} = \mathbf{t}$.

If $\mathbf{X}$ is non-singular solve $\mathbf{w} = \mathbf{X}^{-1}\mathbf{t}$, however, if $\mathbf{X}$ is rectangular (which is usually the case), then there are more equations than unknowns, that is, the equation system is overdetermined.

Let us search for $\mathbf{w}$ that minimizes the error

$$\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{t}$$

one approach is to minimize the squared length of the error vector $\mathbf{e}$

$$J(\widetilde{\mathbf{w}}) = \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2 = \sum_{i=1}^{N} \left(\mathbf{w}^T \mathbf{x}^{(i)} - t^{(i)} \right)^2$$

MSE and Pseudoinverse (cont.)

Forming the gradient

$$\nabla J = \sum_{i=1}^{N} 2\left(\mathbf{w}^T \mathbf{x}^{(i)} - t^{(i)}\right)\mathbf{x}^{(i)} = 2\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{t})$$

and setting ∇J to zero gives $\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{t}$. Observe that $\mathbf{X}^T\mathbf{X}$ is a $\widetilde{d} \times \widetilde{d}$ matrix which often is non-singular. In the non-singular case, one can solve $\mathbf{w}$ uniquely as

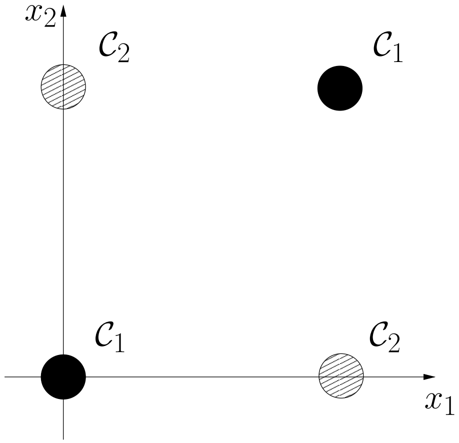
$$\begin{aligned} \mathbf{w} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{t} \\ &= \mathbf{X}^{\dagger}\mathbf{t} \end{aligned}$$

The $\widetilde{d} \times N$ matrix $\mathbf{X}^{\dagger} \equiv \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ is called *pseudoinverse* of $\mathbf{X}$.

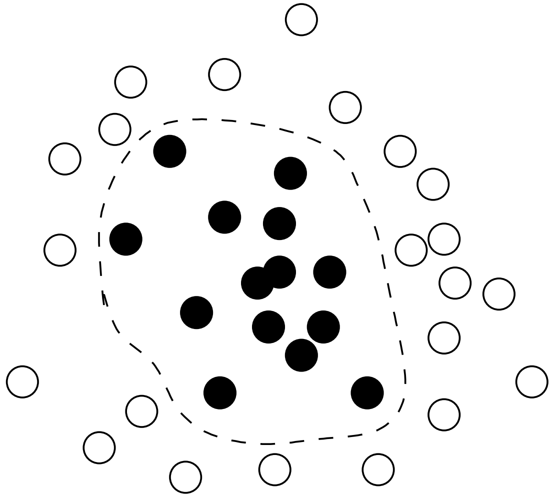
Linear Separability

Decision boundaries of single-layer networks are linear (hyperplanar in higher dimensions).

- Very restricted class of decision boundaries
- Examples:



XOR-Problem



Points are not linearly separable

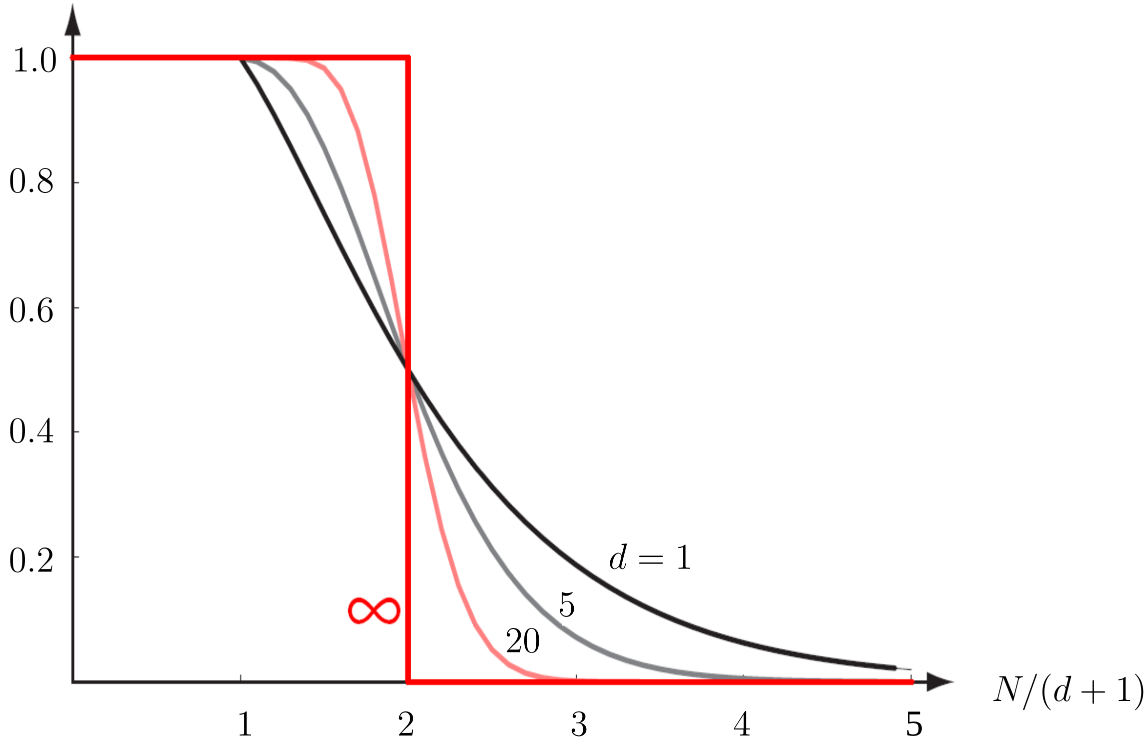
Probability for Linear Separability

- Probability that a random set of points will be linearly separable
- Suppose we have N points distributed at random in d dimensions in general position (not collinear)
- Randomly assign each of the points to one of the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ (with eq. probability)
- For N data points there are 2^N possible class assignments (dichotomies $\equiv$ binary partitions)

Question: What fraction $F(N, d)$ of these dichotomies is linearly separable?

Probability for Linear Separability (cont.)

$$F(N,d) = \begin{cases} 1 & \text{when } N \leq d+1 \\ \frac{1}{2^{N-1}} \sum_{i=0}^{d} \binom{N-1}{i} & \text{when } N \geq d+1 \end{cases}$$



If number of points is $\leq d+1$, then any labeling leads to a separable problem.