Machine Learning for Signal Processing
Bayes Classification and Regression

Instructor: Bhiksha Raj

Recap: KNN

- A very effective and simple way of performing classification
- Simple model: For any instance, select the class from the instances close to it in feature space

Multi-class Image Classification

k-Nearest Neighbors

Given a query item:
Find k closest matches
in a labeled dataset ↓

k-Nearest Neighbors

Given a query item:
Find k closest matches

Return the most
Frequent label

k-Nearest Neighbors

k = 3 votes for “cat”

k-Nearest Neighbors

2 votes for cat,

1 each for Buffalo,

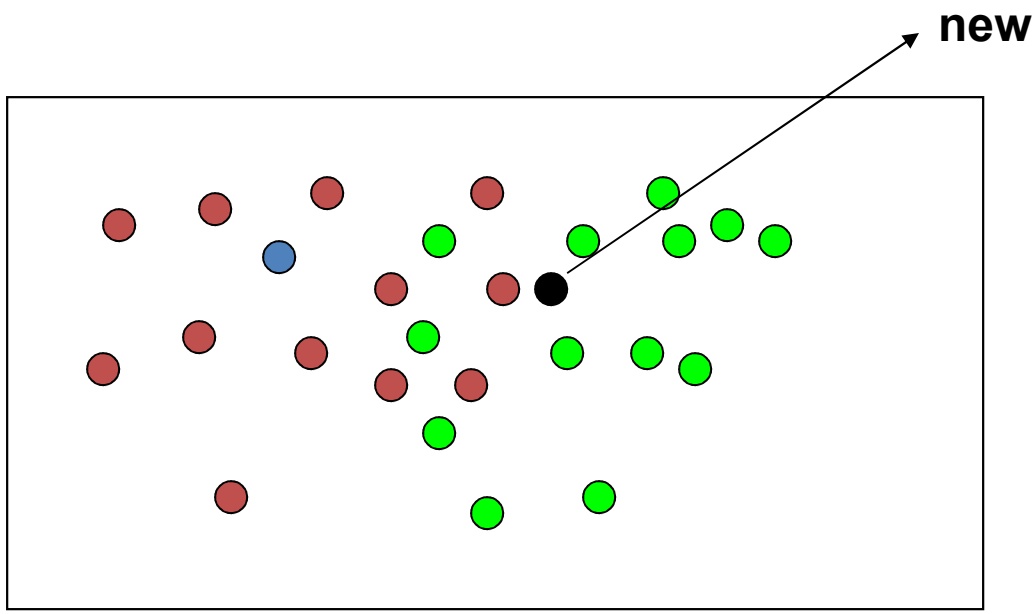
Deer, Lion

Cat wins…

Nearest neighbor method

- Weighted majority vote within the k nearest neighbors

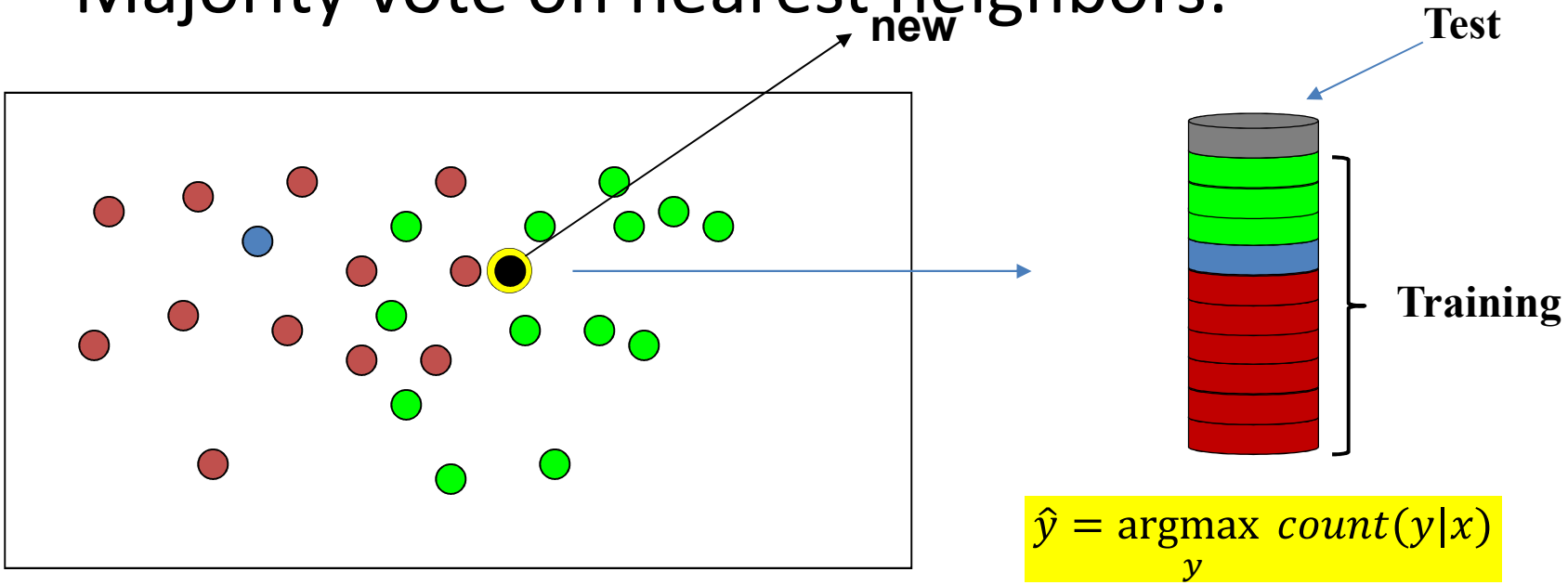
$$\hat{Y}(x) = argmax_c \sum_{x_{i \in N_k(x)}, y_i = c} w(x, x_i) y_i$$



K= 1: blue
K= 3: green

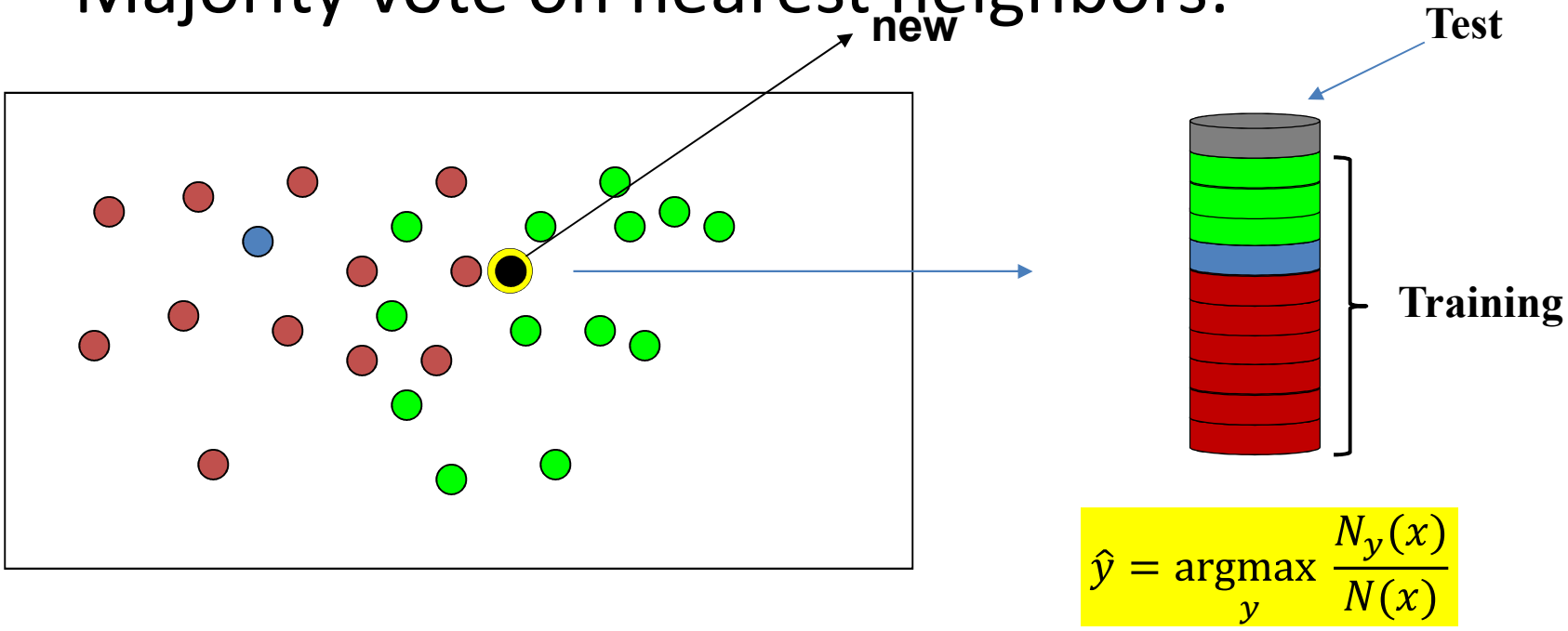
But what happens if..

- You have many training instances at exactly that value of x?
- Majority vote on nearest neighbors:



But what happens if..

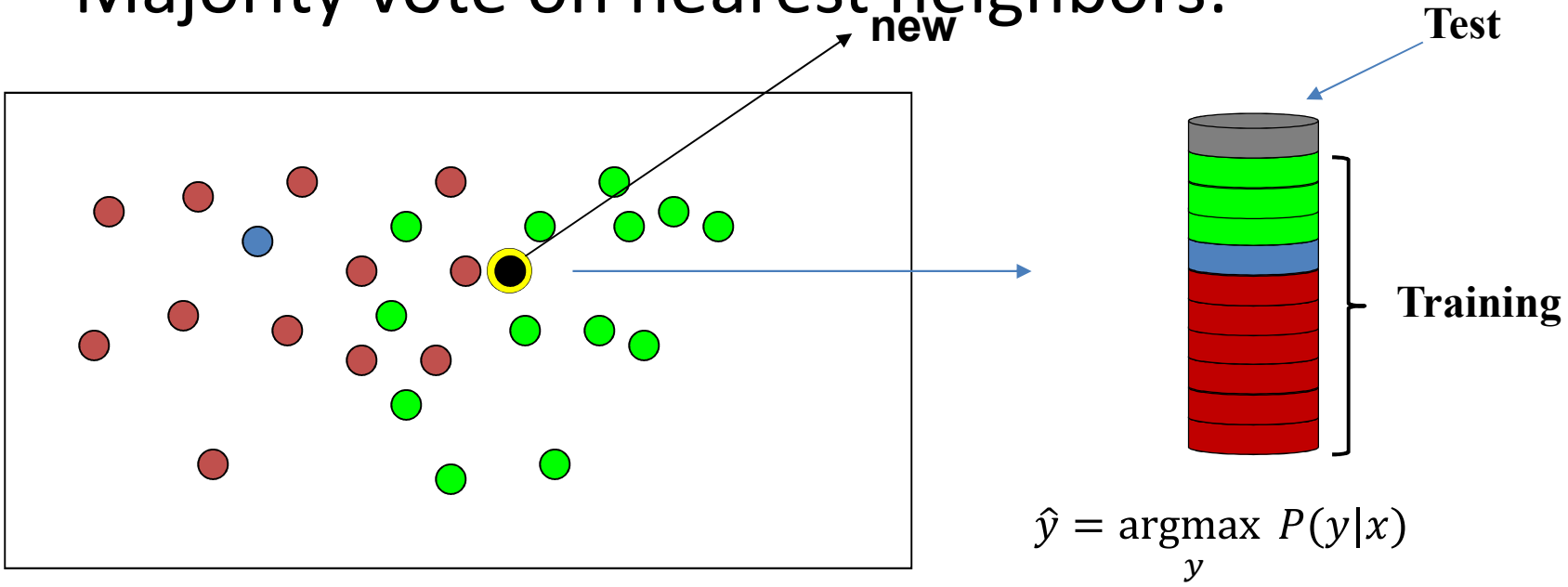
- You have many training instances at exactly that value of x?
- Majority vote on nearest neighbors:



$$\hat{y} = \underset{y}{\text{argmax}} \frac{N_y(x)}{N(x)}$$

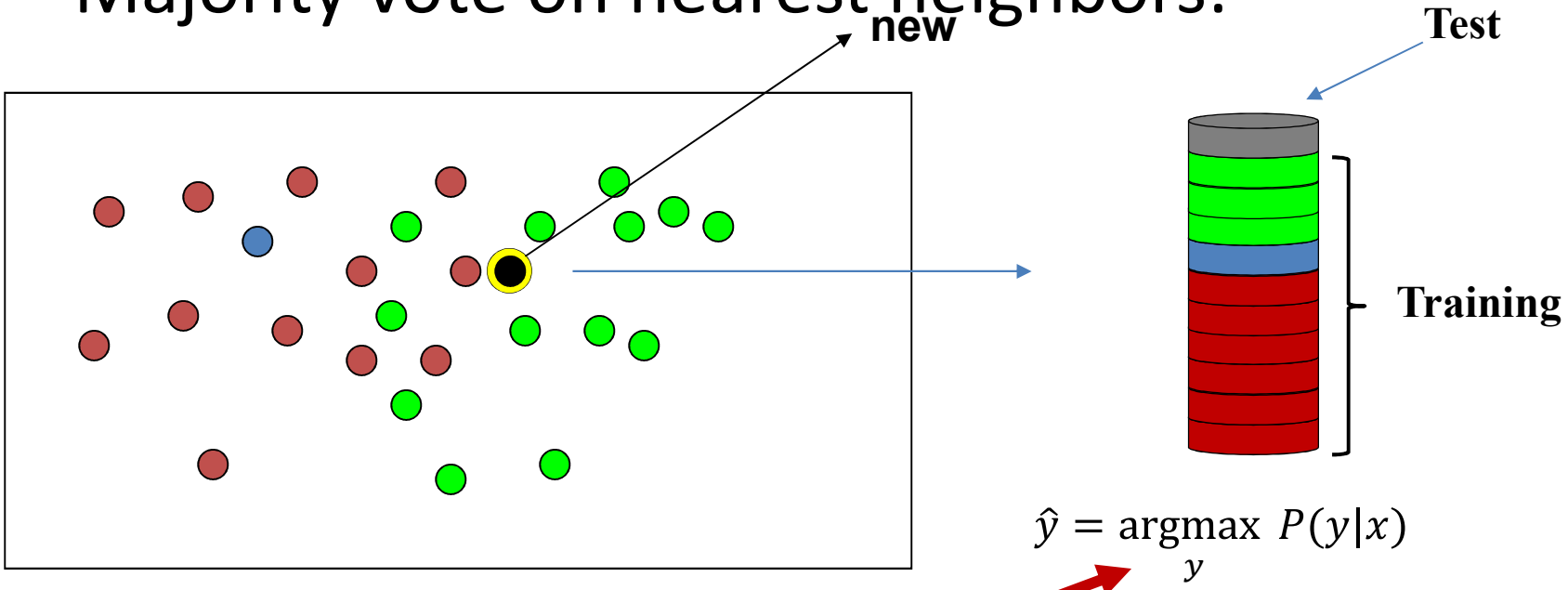
But what happens if..

- You have many training instances at exactly that value of x?
- Majority vote on nearest neighbors:



But what happens if..

- You have many training instances at exactly that value of x?
- Majority vote on nearest neighbors:



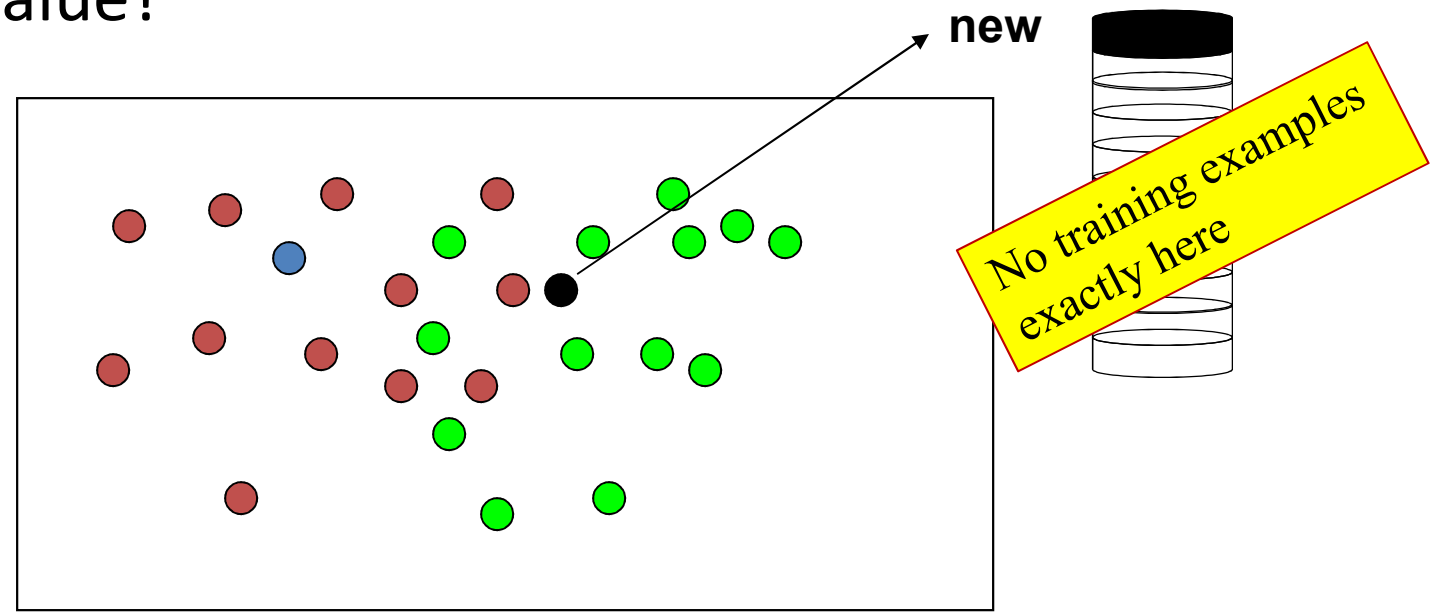
- Bayes Classification Rule

Bayes Classification Rule

- For any observed feature X, select the class value Y that is most frequent
  - Also applies to continuous valued predicted variables
    - I.e. regression
- Select Y to maximize the *a posteriori* probability $P(Y|X)$
  - Bayes classification is an instance of *maximum a posteriori* estimation

Bayes classification

- What happens if there are no *exact* neighbors
  - No training instances with exactly the same X value?



Bayes Classification Rule

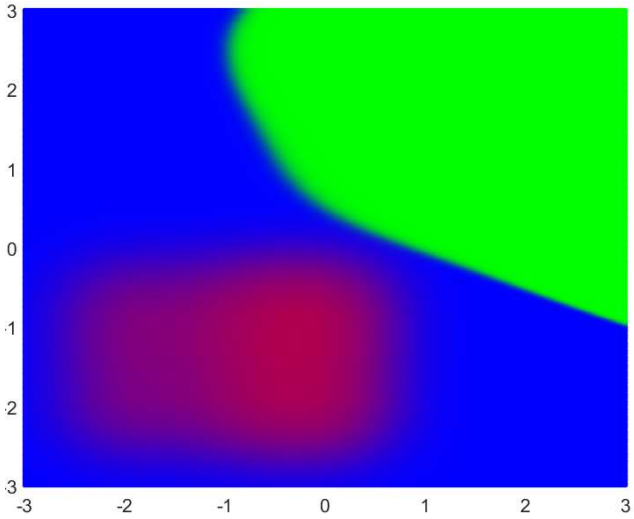
- Given
  - a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$
  - Conditional probability distributions $P(C|X)$
  - Classification performed as

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C|X)$$

- Problem: How do you characterize $P(C|X)$
  - Require a function that, given an X, computes $P(C|X)$ for every class C

Modelling P(C|X)

Each pixel is a combination of red green and blue weighted by the a posteriori probability of the classes



Blue: Class 1
Red: Class 2
Green: Class 3

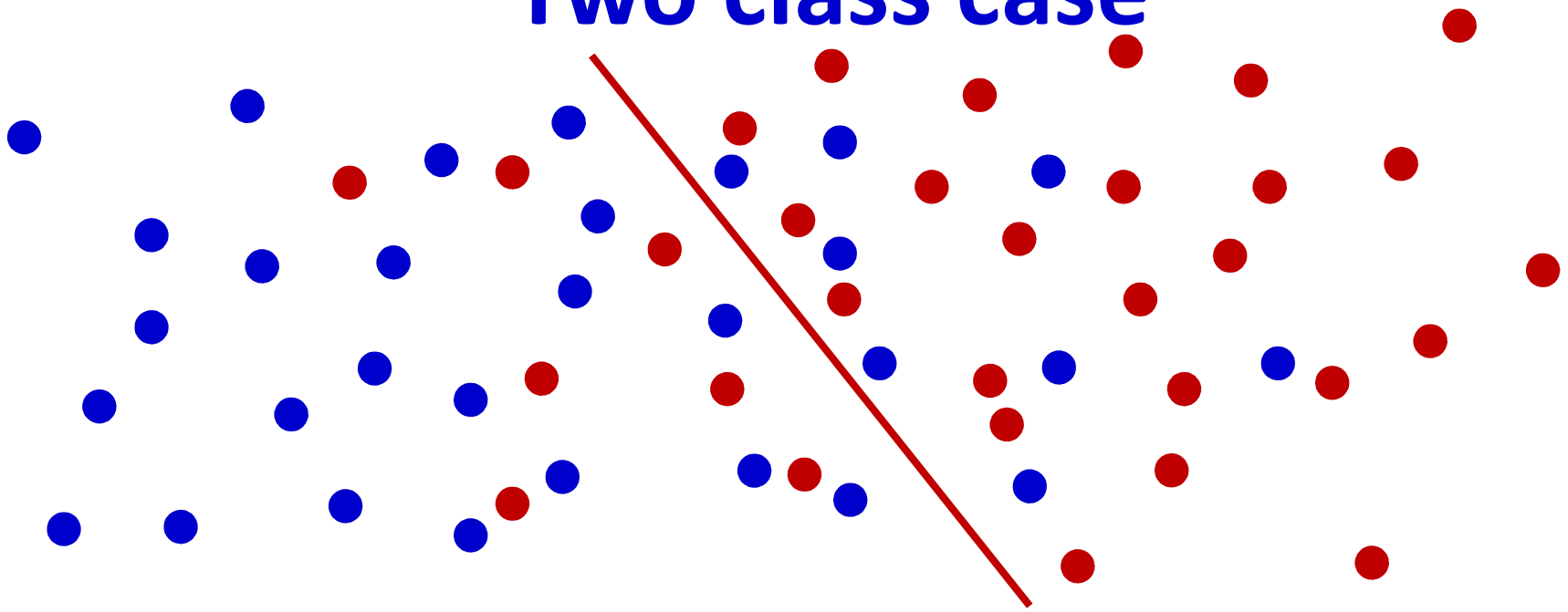
- Assumption: there's a continuous function that, at every X, produces a vector of outputs $P(C|X)$ for every class C
  - The "decision boundary" for any class is the boundary within which its own posterior has the highest value
- This function accurately represents the *actual* a posteriori probabilities for the classes
- Objective: Estimate this function

Modelling the posterior

- To model the posterior, we need a functional form for $P(C|X)$ which can be learned
- Typically this functional form is expressed in terms of distance from a decision boundary

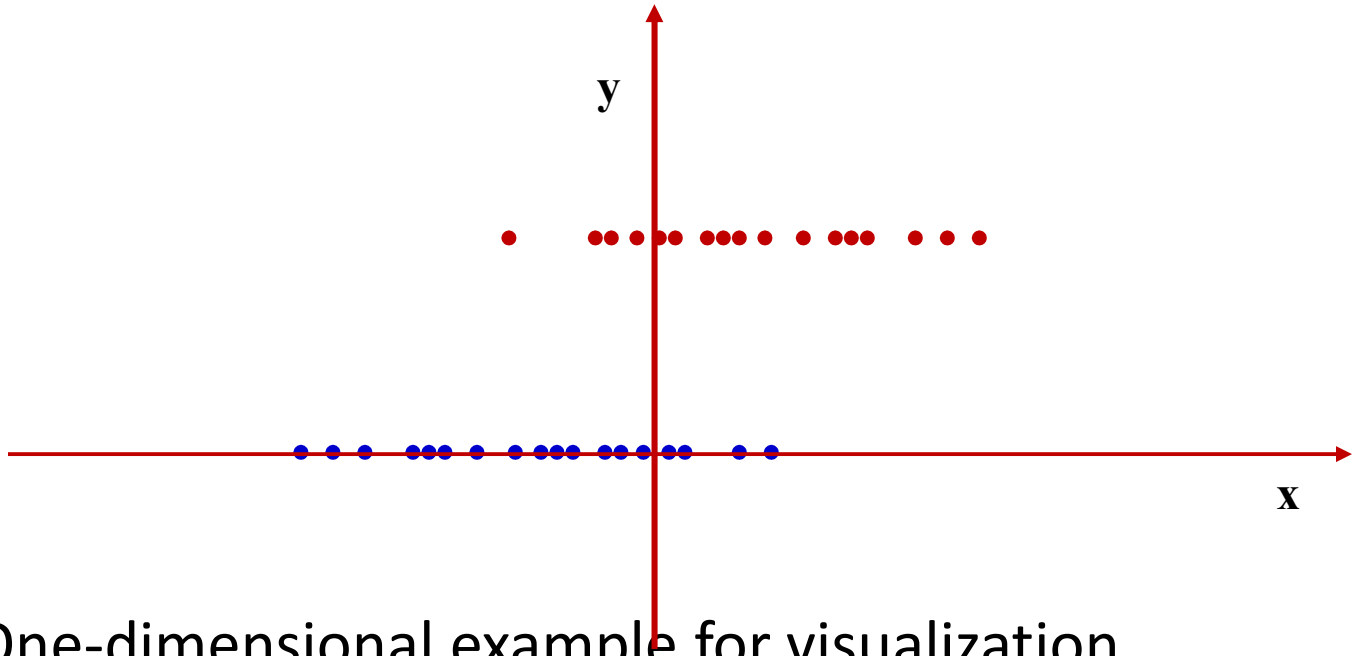
- The simplest decision boundary is the linear boundary

Bayesian Linear Classification: Two class case



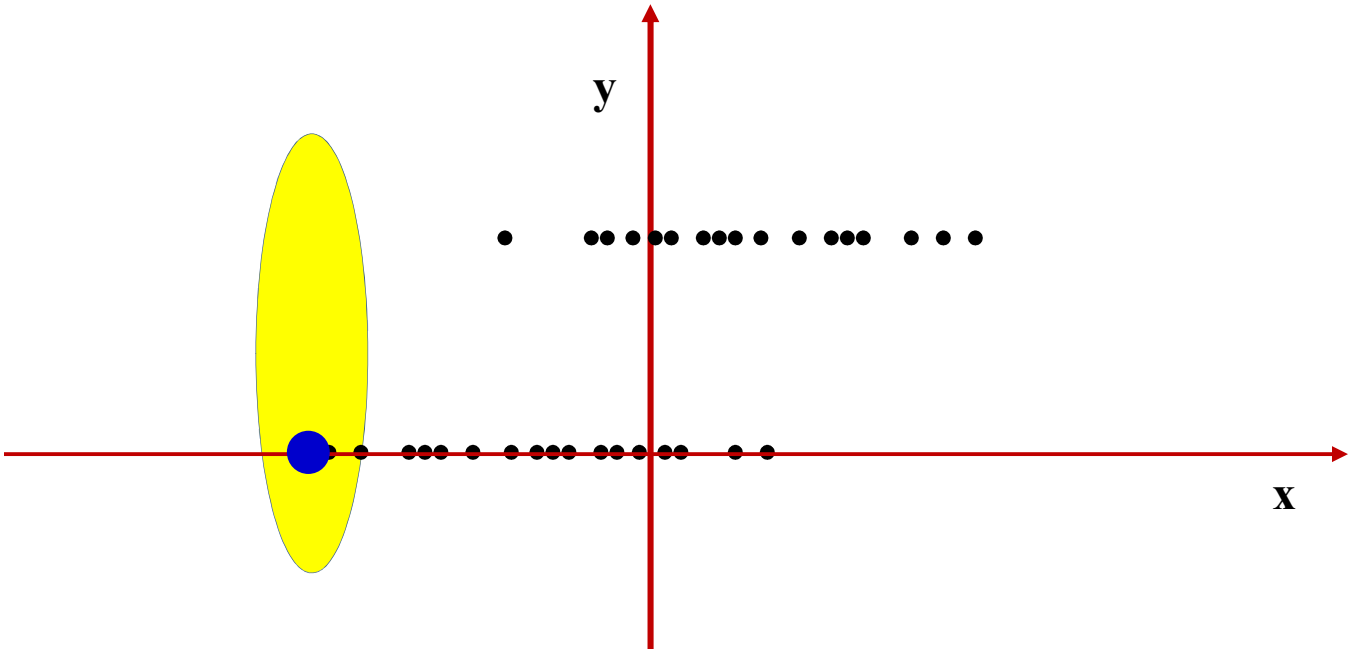
- First: Two-class classification
- Assumption: the decision boundary between the classes is a simple hyperplane
- As you go away from the hyperplane, the fraction of data from one class increases, while that from the other decreases
  - Will also hold for any *sample* of data

1-D binary class example



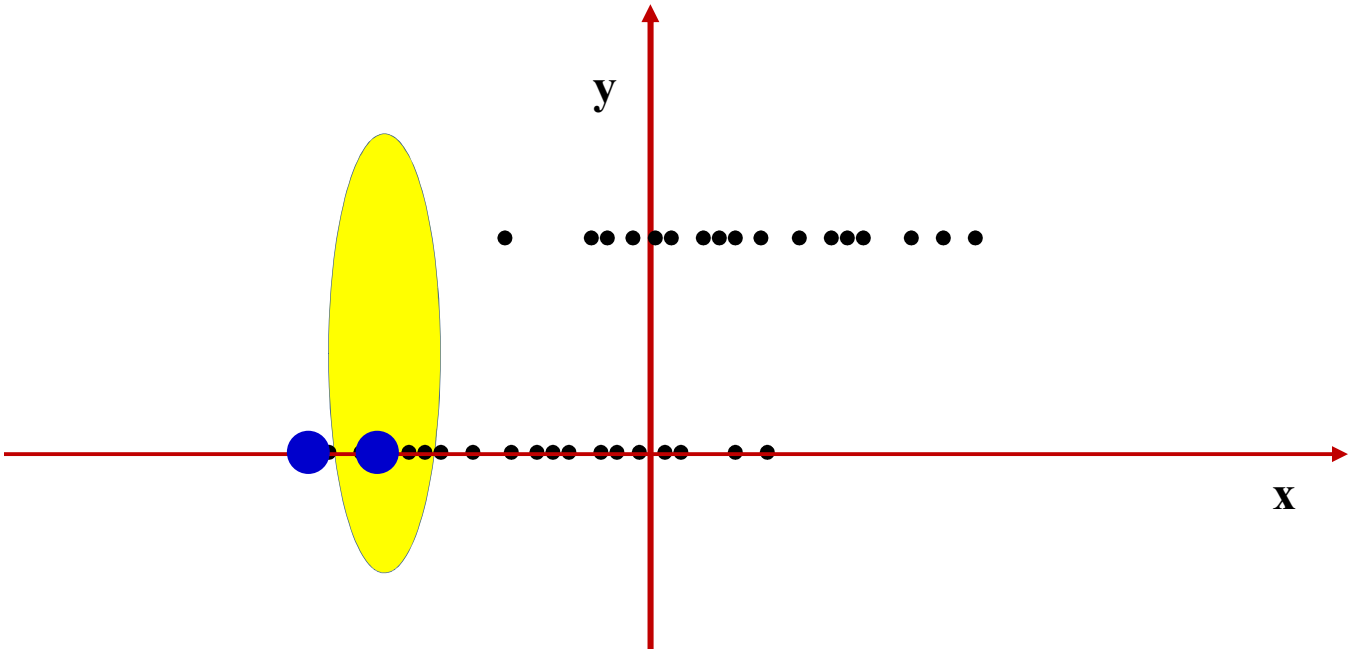
- One-dimensional example for visualization
- Only two classes (represented by y=0 and y=1)
  - All (red) dots at Y=1 represent instances of class Y=1
  - All (blue) dots at Y=0 are from class Y=0
  - The data are not linearly separable
    - In this 1-D example, a linear separator is a threshold
    - No threshold will cleanly separate red and blue dots

The *probability* of y=1



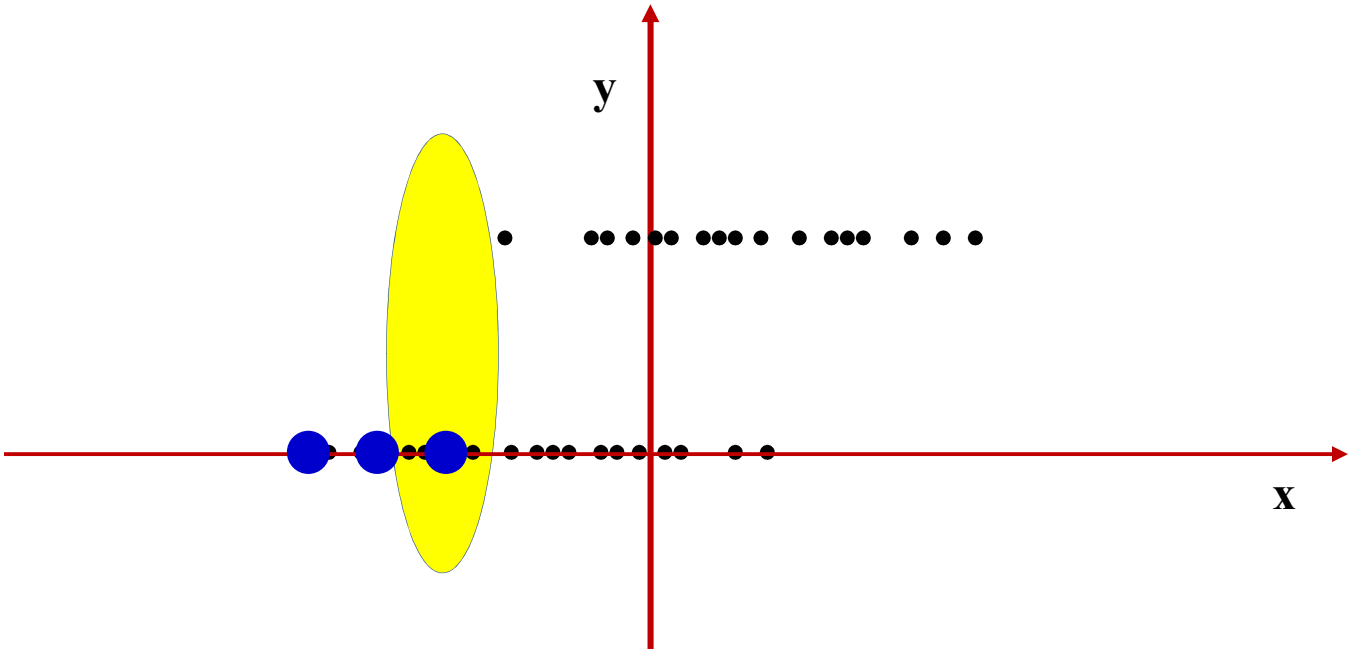
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of Y=1 at that point

The *probability* of y=1



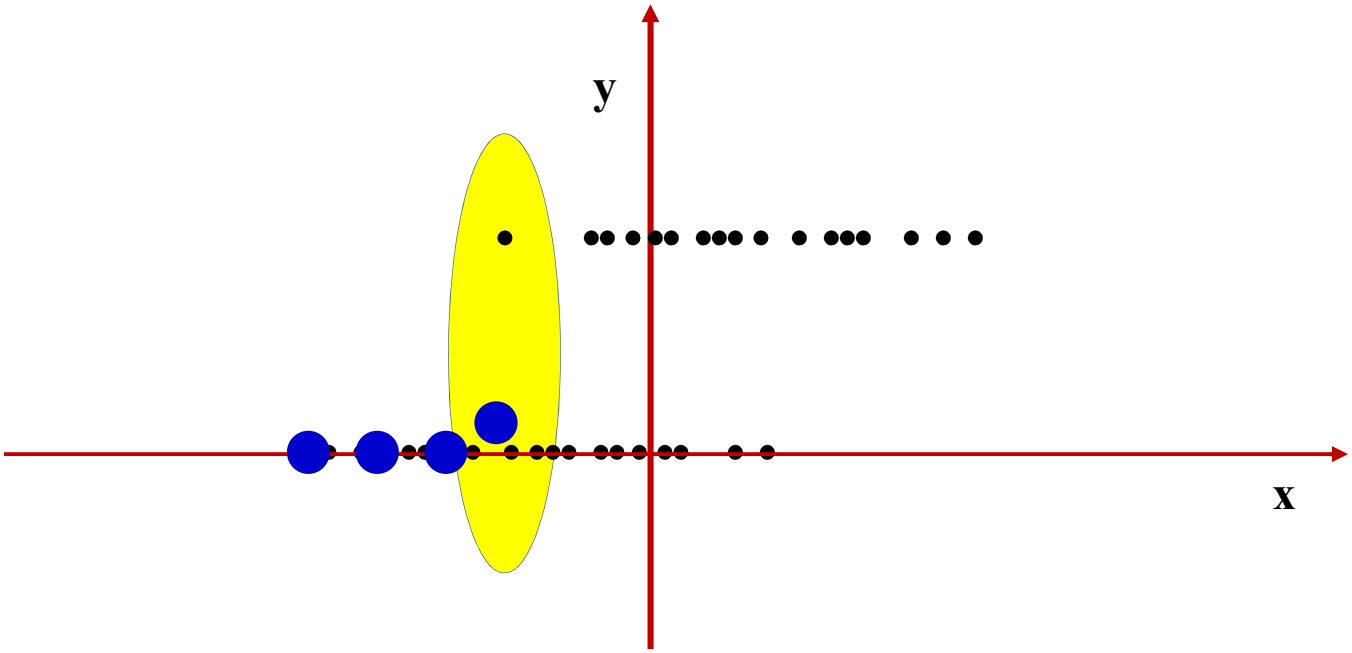
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



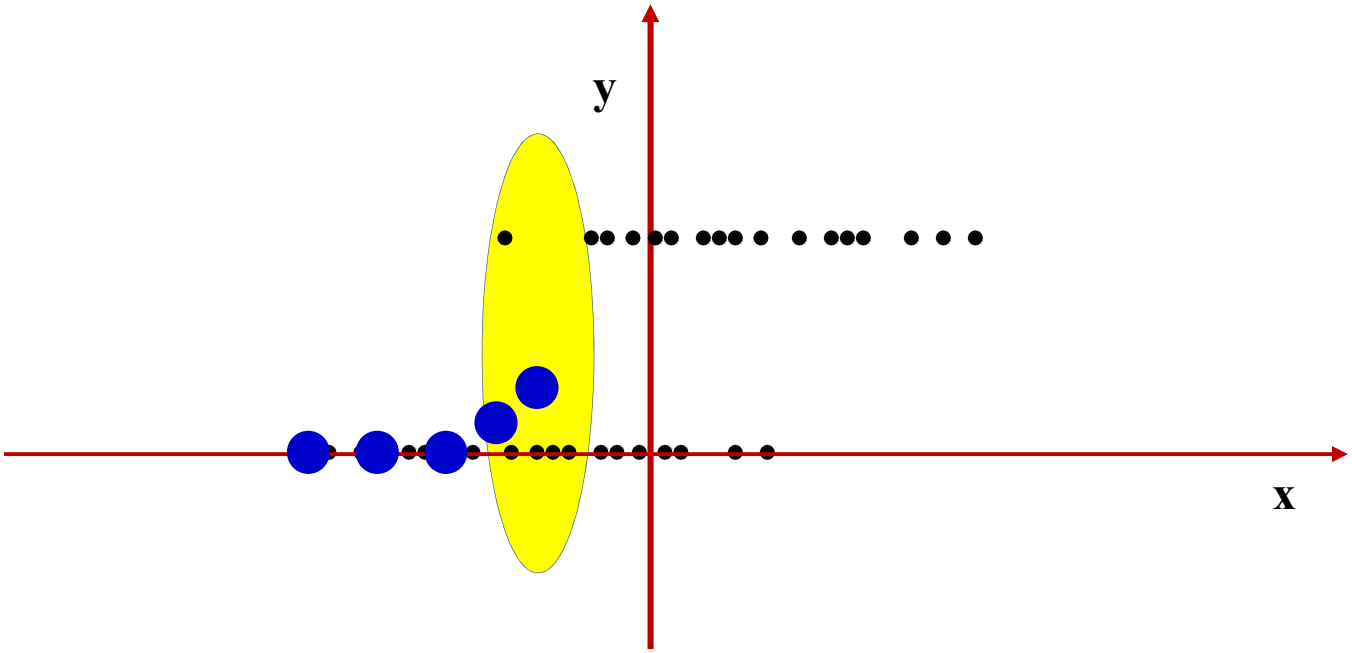
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



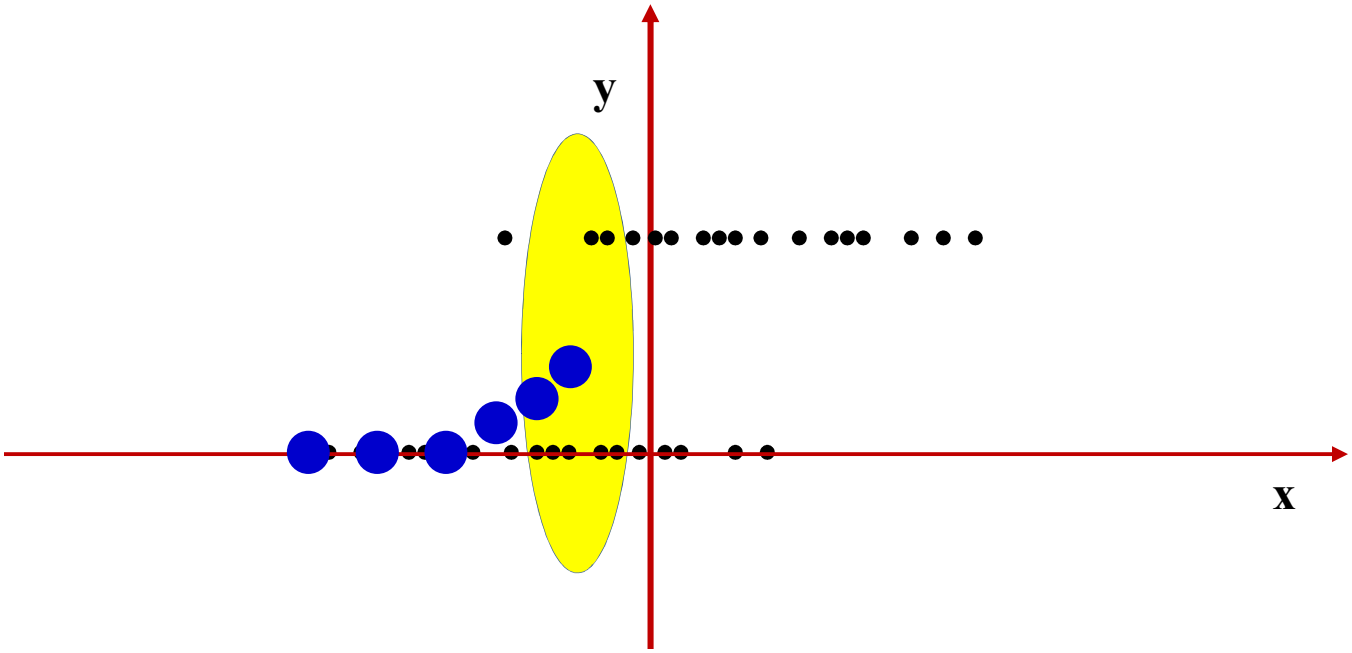
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



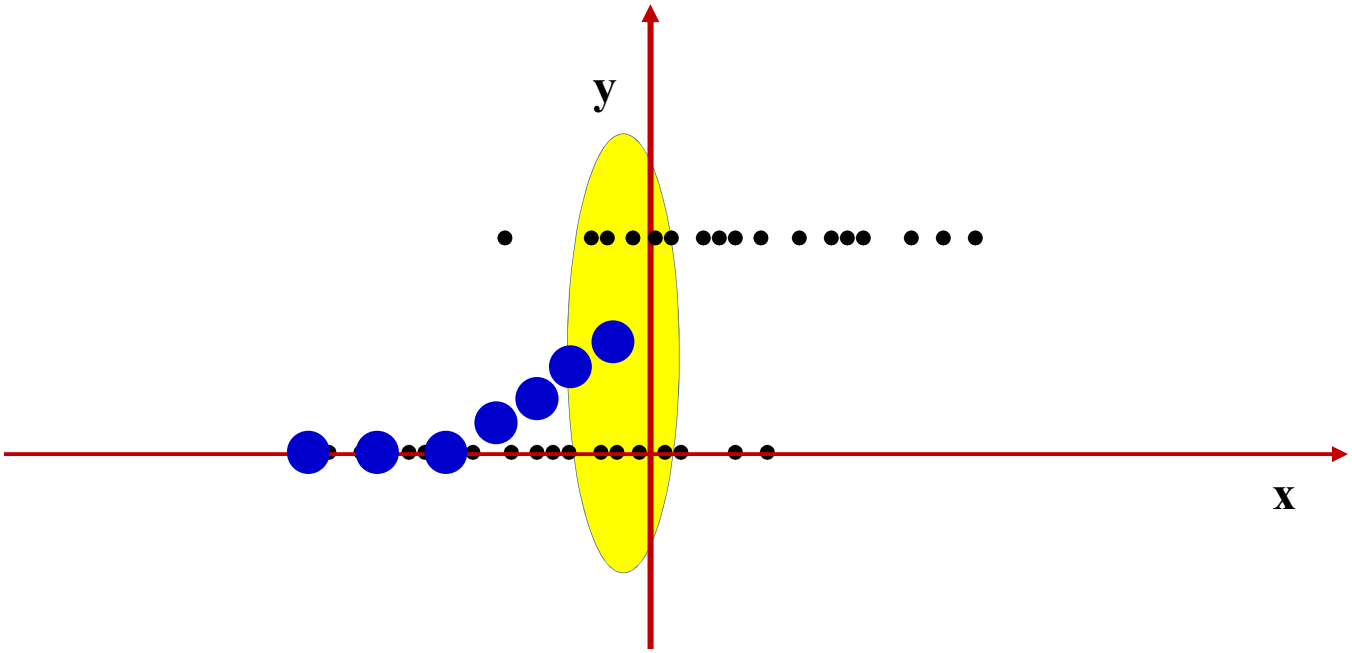
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



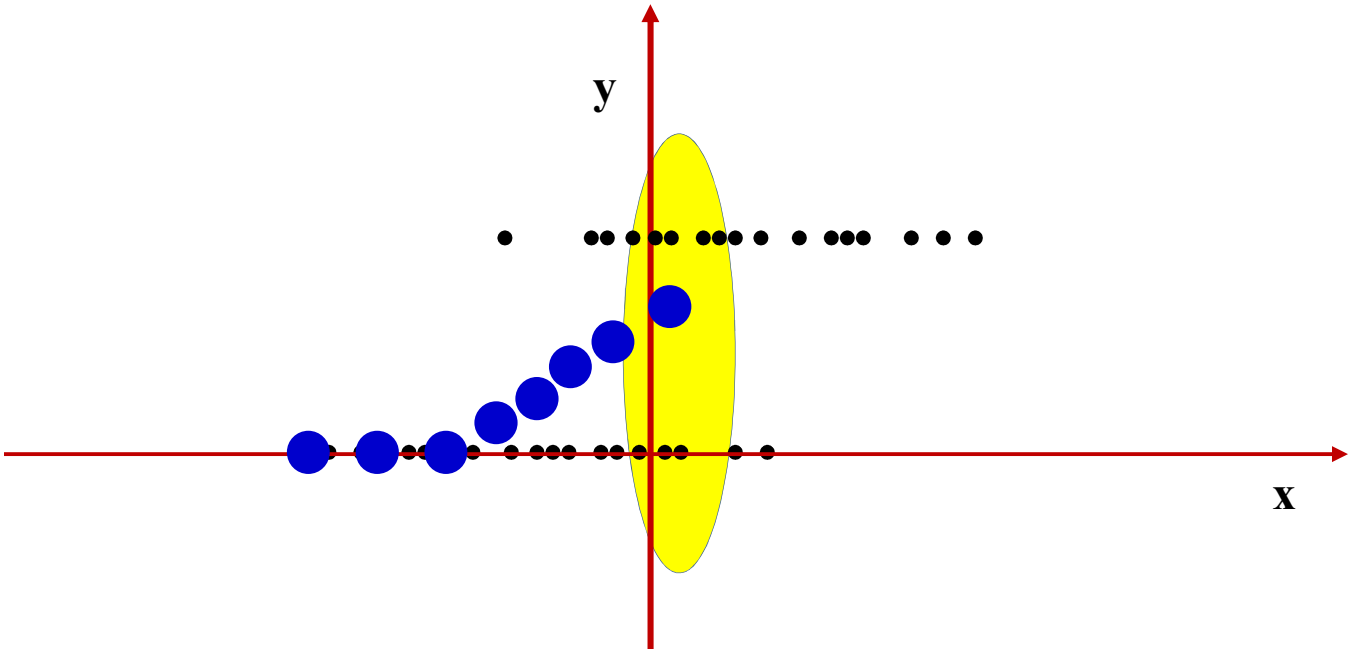
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



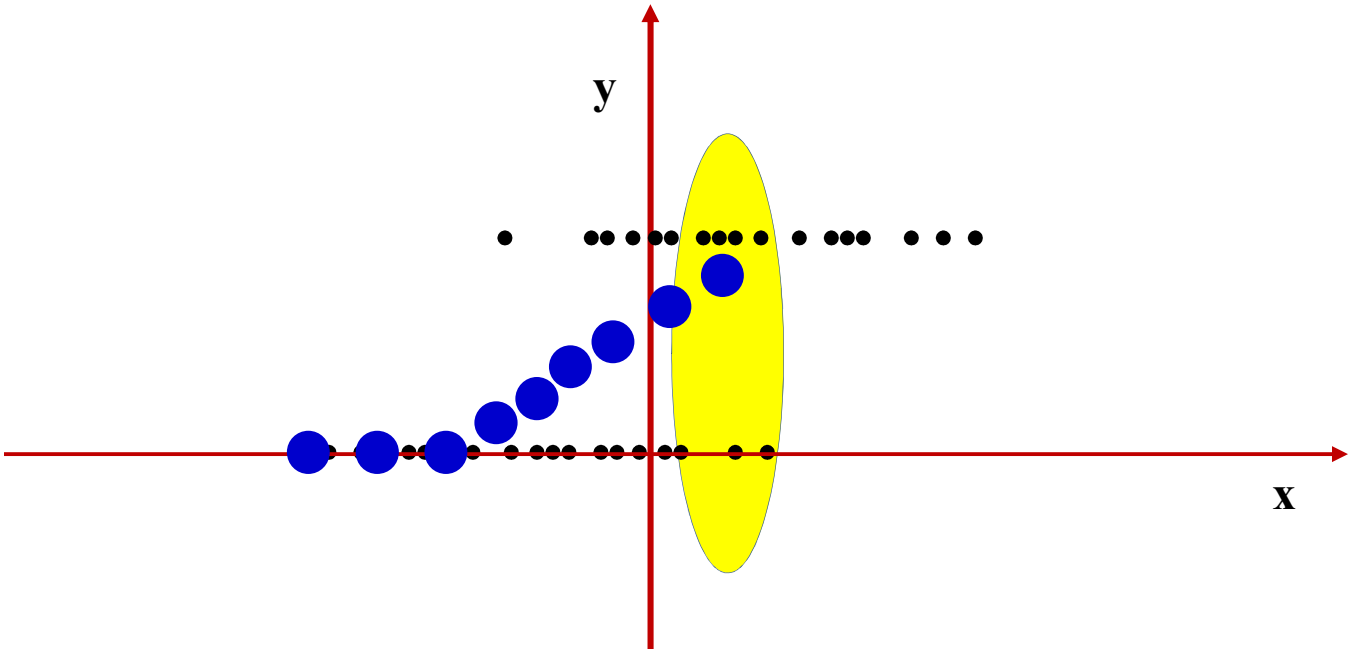
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



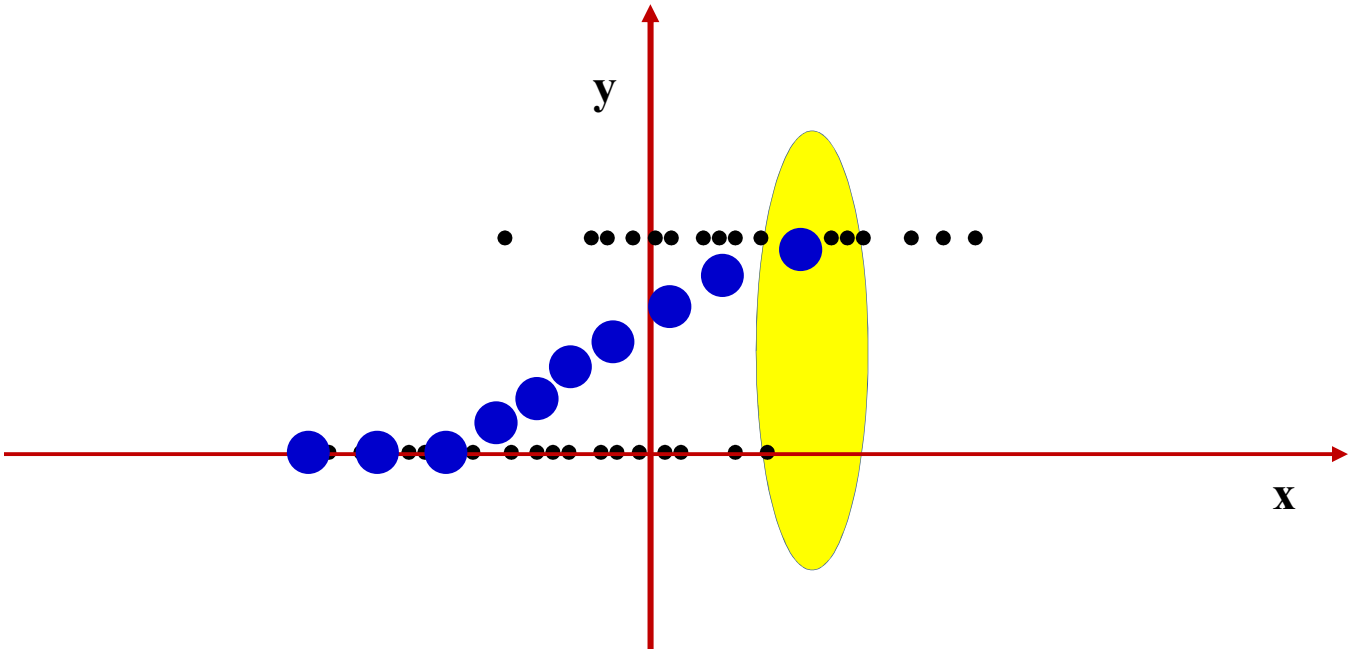
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



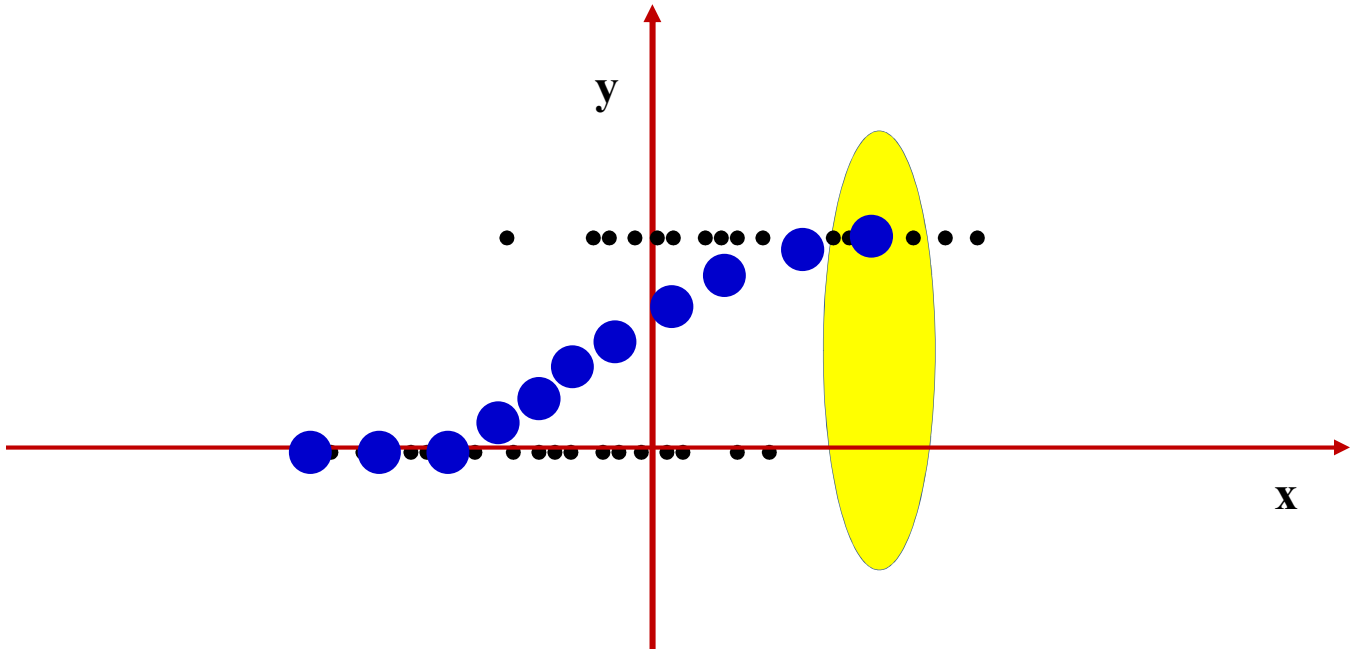
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



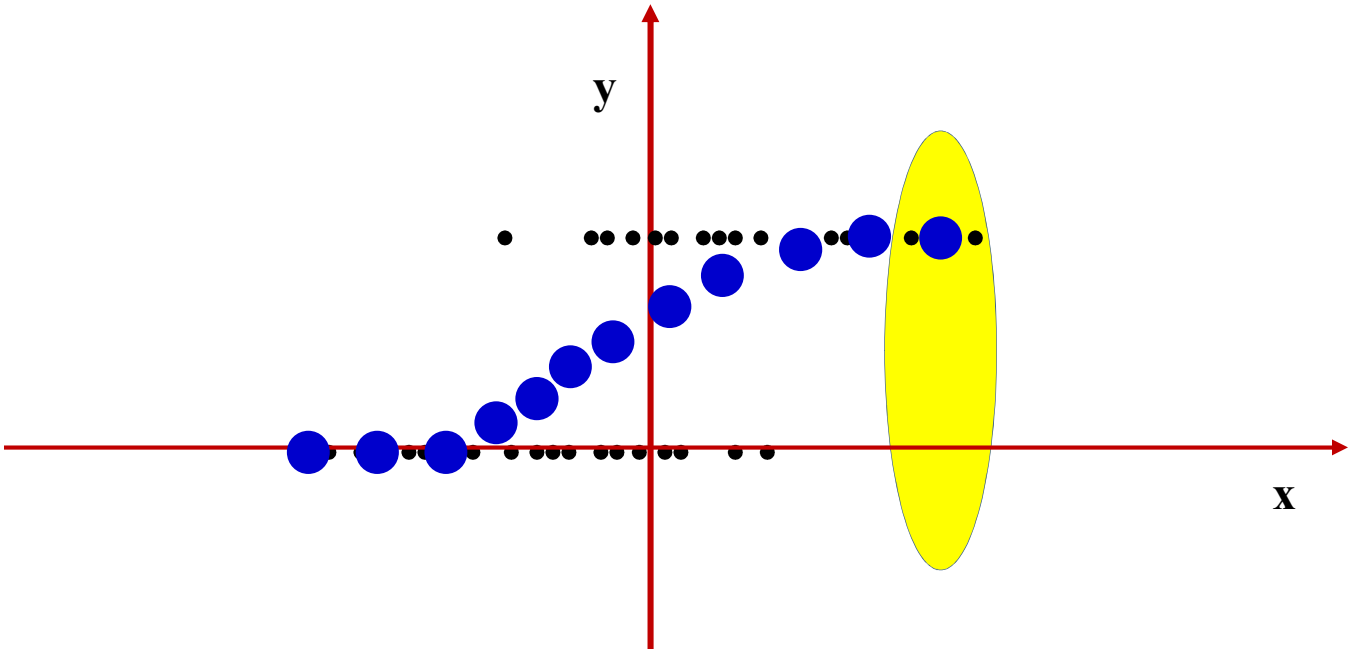
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



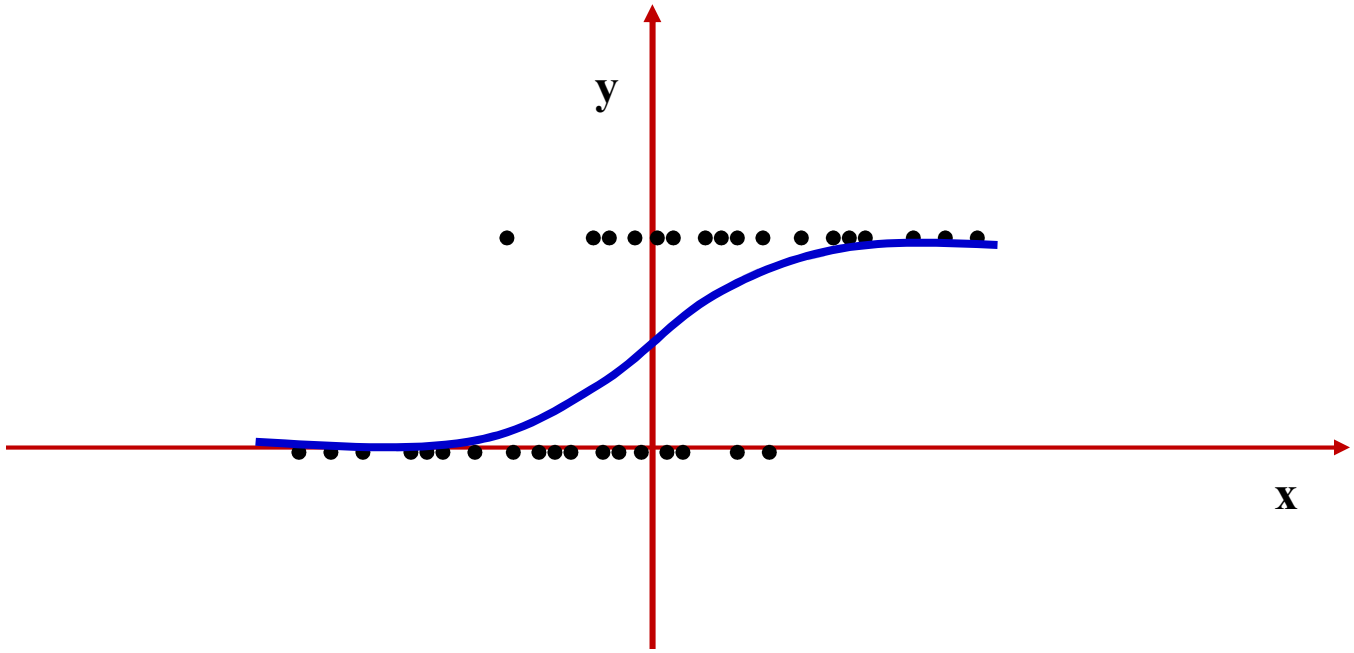
- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1



- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1

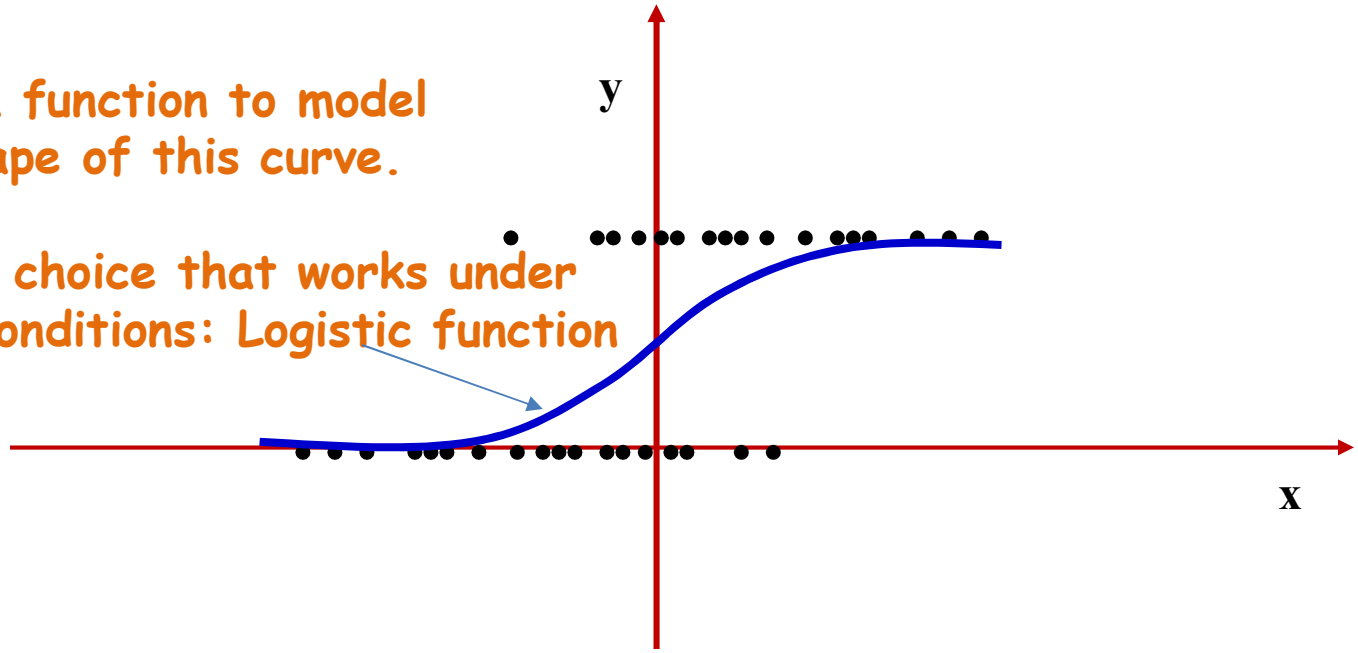


- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

The *probability* of y=1

Need a function to model the shape of this curve.

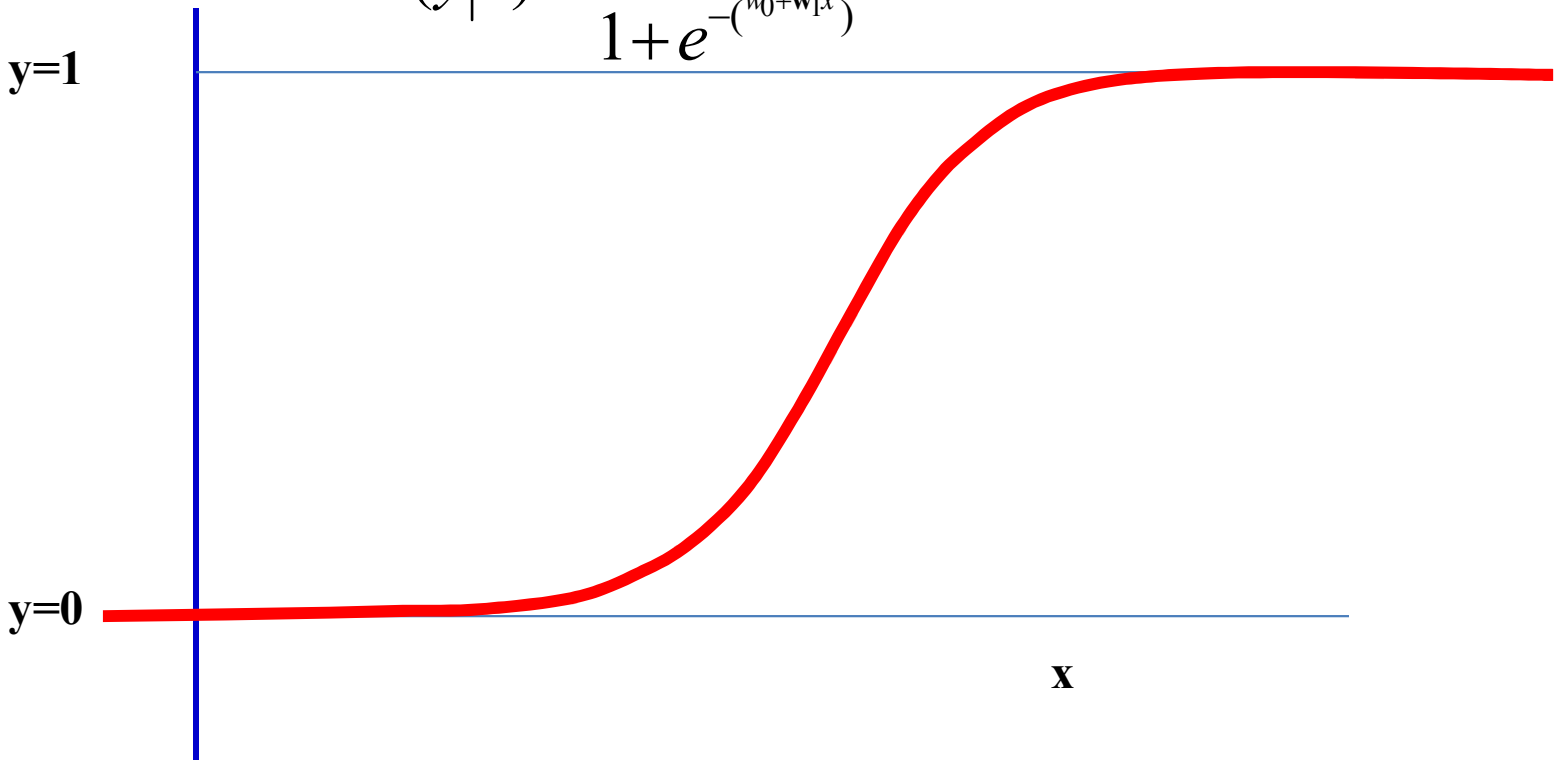
A good choice that works under many conditions: Logistic function



- Consider this differently: at each point look at a small window around that point
- Plot the average value within the window
  - This is an approximation of the *probability* of 1 at that point

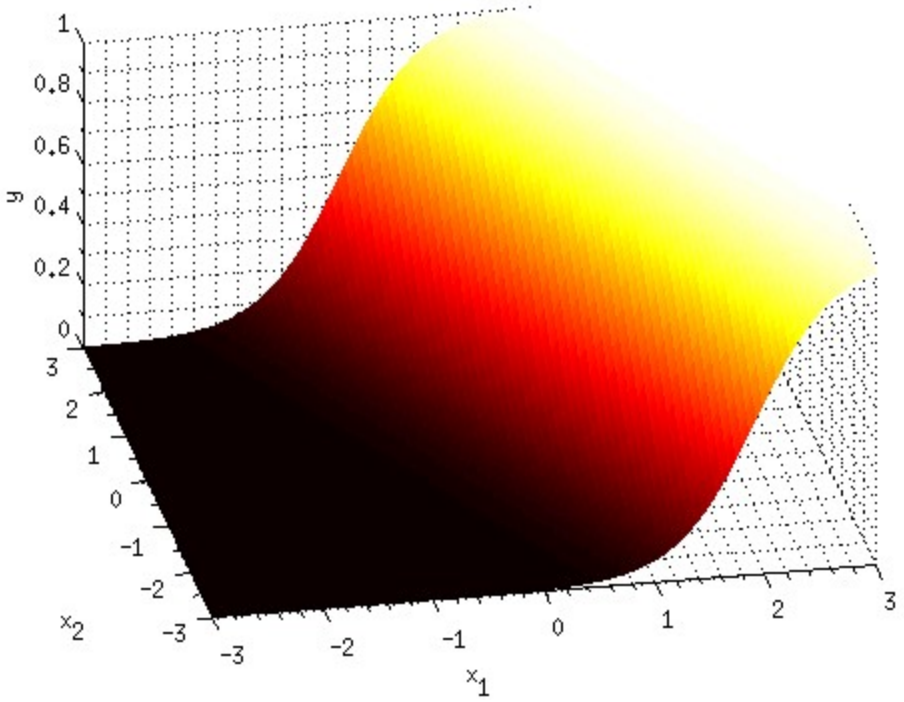
The logistic regression model

$$P(y|x) = \frac{1}{1+e^{-(w_0+\mathbf{w}_1 x)}}$$

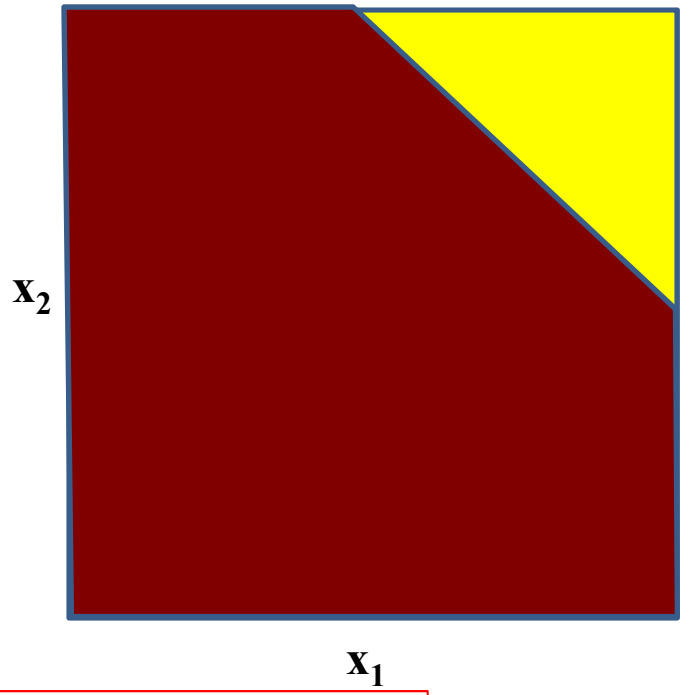


- Class 1 becomes increasingly probable going left to right
  - Very typical in many problems
  - The logistic is a function of the distance from the $P(y|x) = 0.5$ boundary

For two-dimensional input



Decision: y > 0.5?



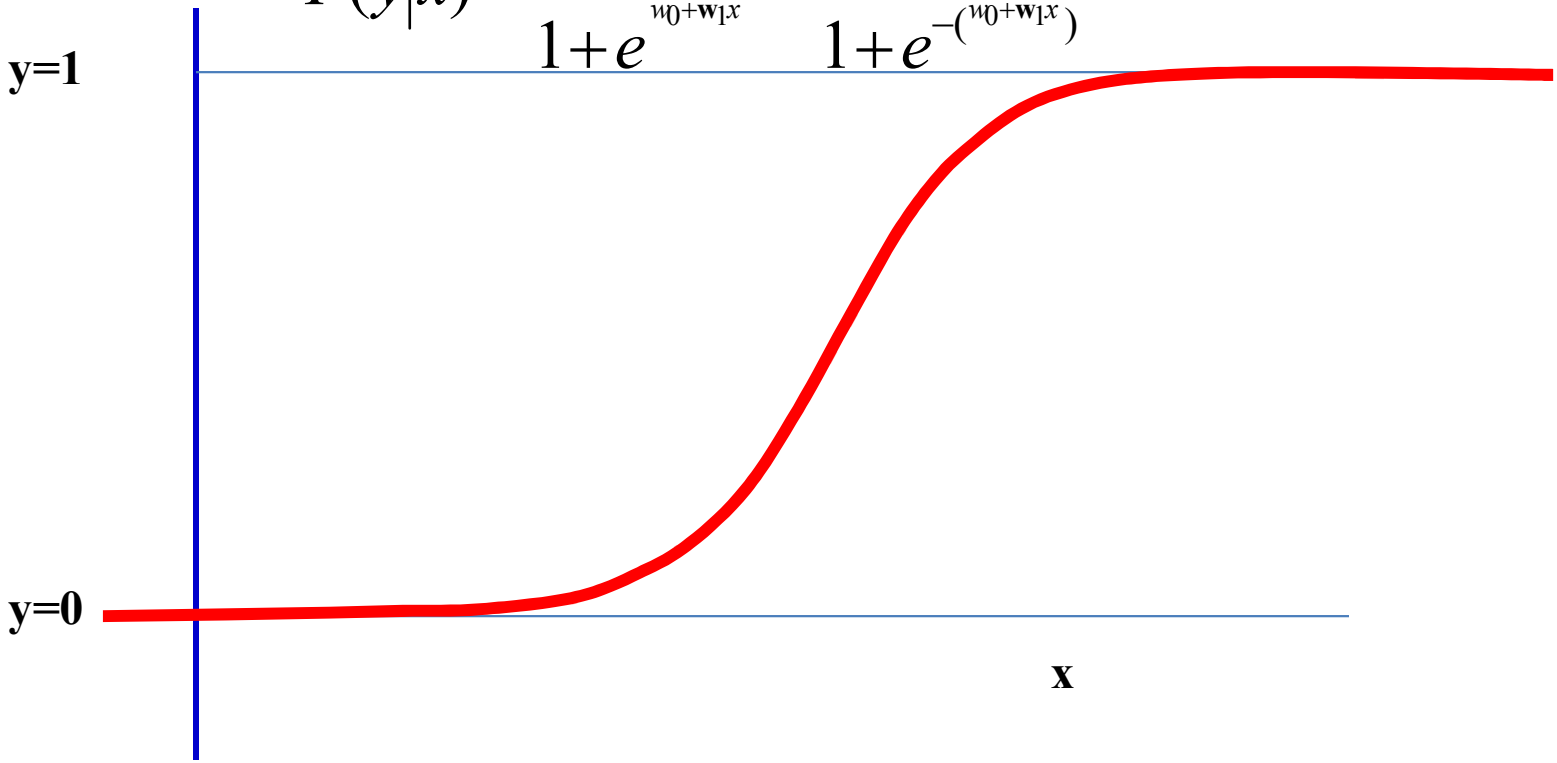
When X is a 2-D variable

$$P(y|x) = \frac{1}{1+e^{-(w_0+\mathbf{w}_1 x)}}$$

- The decision boundary for P(Y|X)=0.5 is a hyperplane
  - It is a linear model

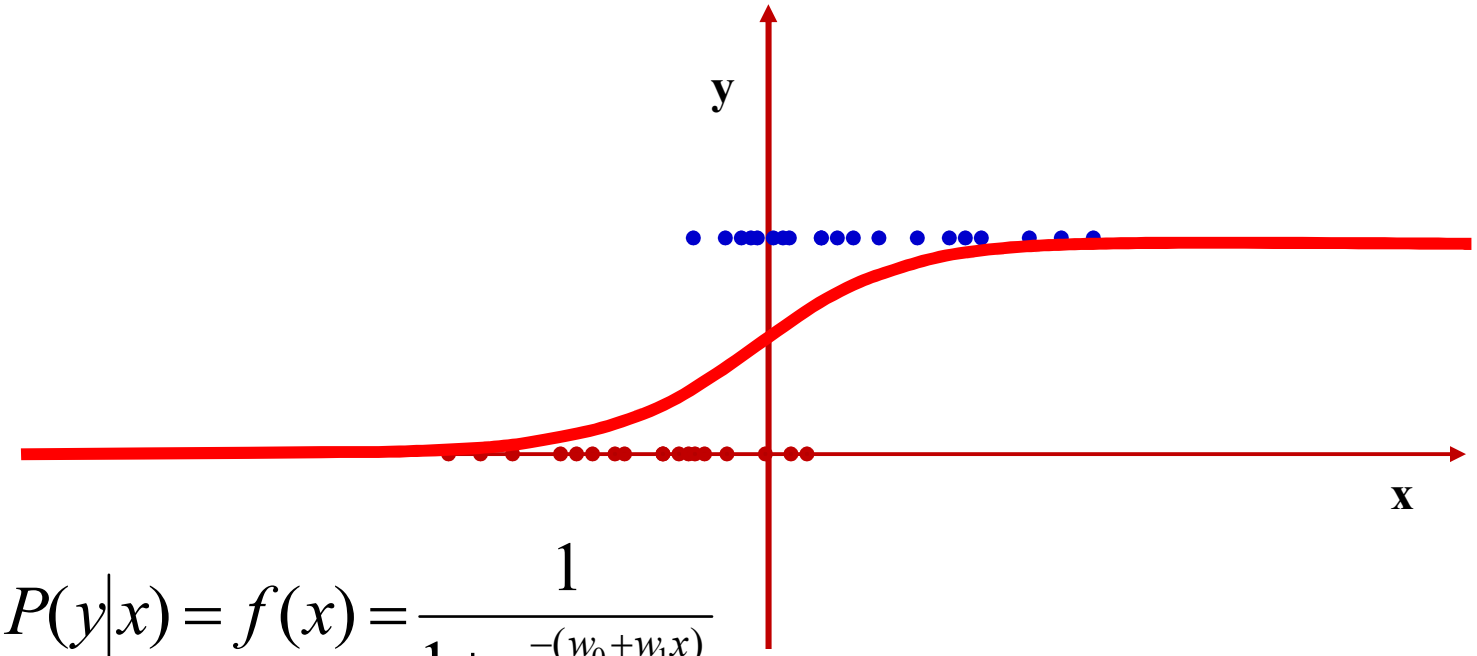
The logistic regression model

$$P(y|x) = \frac{e^{w_0 + \mathbf{w}_1 x}}{1 + e^{w_0 + \mathbf{w}_1 x}} = \frac{1}{1 + e^{-(w_0 + \mathbf{w}_1 x)}}$$



- Note how it varies with $w_0 + w_1\mathbf{x}$

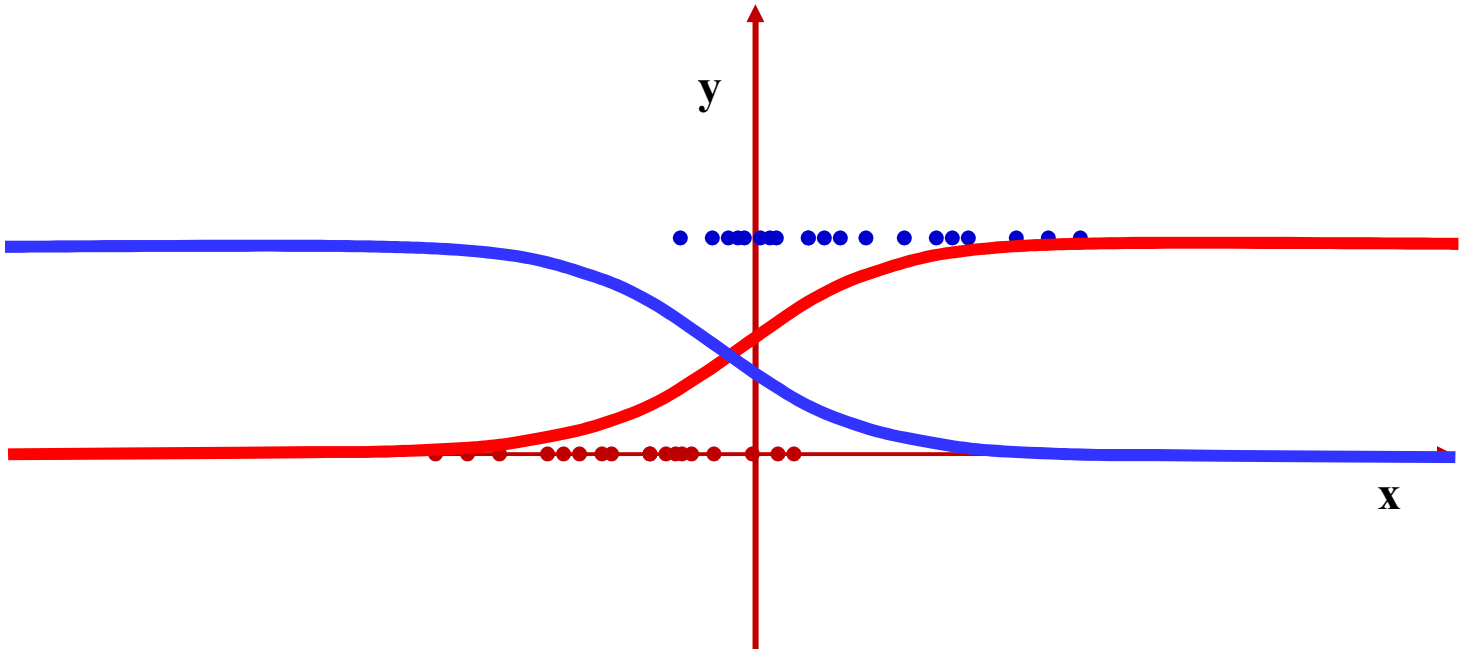
Estimating the model



$$P(y|x) = f(x) = \frac{1}{1+e^{-(w_0+w_1x)}}$$

- Given the training data (many (x, y) pairs represented by the dots), estimate w_0 and w_1 for the curve

Estimating the model



- **Easier to represent using a y = +1/-1 notation**

$$P(y=1|x)=\frac{1}{1+e^{-(w_0+w_1x)}} \qquad P(y=-1|x)=\frac{1}{1+e^{(w_0+w_1x)}}$$

$$P(y|x)=\frac{1}{1+e^{-y(w_0+w_1x)}}$$

Estimating the model

- Given: Training data

$$(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)$$

- Xs are vectors, ys are binary (0/1) class values
- Total probability of data

$$P\big((X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)\big) = \prod_i P(X_i, y_i)$$

$$= \prod_i P(y_i|X_i)\, P(X_i) = \prod_i \frac{1}{1 + e^{-y_i(w_0 + w^T X_i)}} P(X_i)$$

Estimating the model

- Likelihood

$$P(Training\ data) = \prod_i \frac{1}{1 + e^{-y_i(w_0 + w^T X_i)}} P(X_i)$$

- Log likelihood

$$\log P(Training\ data) =$$

$$\sum_i \log P(X_i) - \sum_i \log\left(1 + e^{-y_i(w_0 + w^T X_i)}\right)$$

Maximum Likelihood Estimate

$$\widehat{w}_0, \widehat{w}_1 = \underset{w_0, w_1}{\operatorname{argmax}} \log P(Training\ data)$$

- Equals (note argmin rather than argmax)

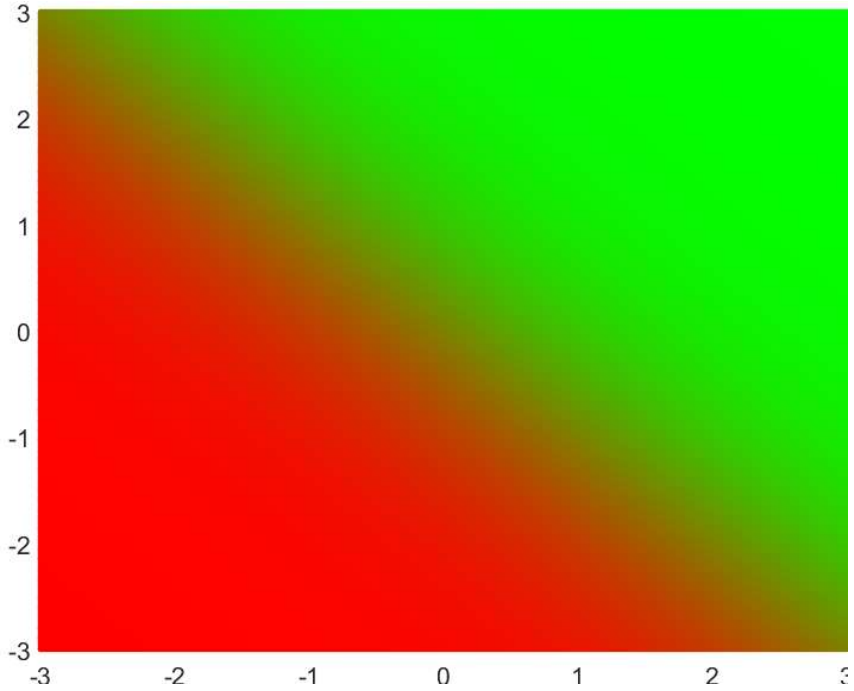
$$\widehat{w}_0, \widehat{w}_1 = \underset{w_0, w}{\operatorname{argmin}} \sum_i \log\left(1 + e^{-y_i(w_0 + w^T X_i)}\right)$$

- Minimizing the KL divergence between the desired output y and actual output $\frac{1}{1+e^{-(w_0+w^T X_i)}}$

- Cannot be solved directly, needs gradient descent

Model learned by logistic regression

Pure Red: 0
Pure Green: 1



- The figure shows the class probability over a two-dimensional feature space
- Any decision threshold $P(C|X) = Const$ is a hyperplane
  - Diagonal line in this case

Multi-class logistic regression

- The simple logistic regression model can be extended to multiple classes:

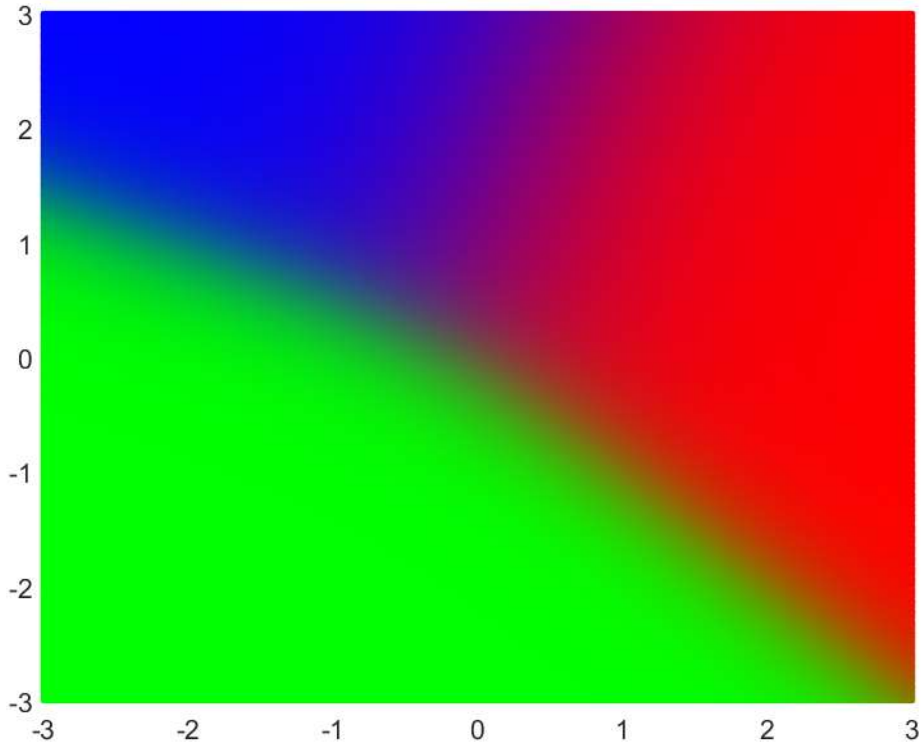
$$P(C|X) = \frac{\exp(W_C^T X)}{\sum_{C'} \exp(W_{C'}^T X)}$$

  - $W_C^T X$ is, in fact, the discriminant function of the classes
    - We've encountered discriminant functions earlier

- Also called a softmax
- Each class C_i has a probability that is exponentially related to the closeness of the vector X to a "representative" vector w_i for the class

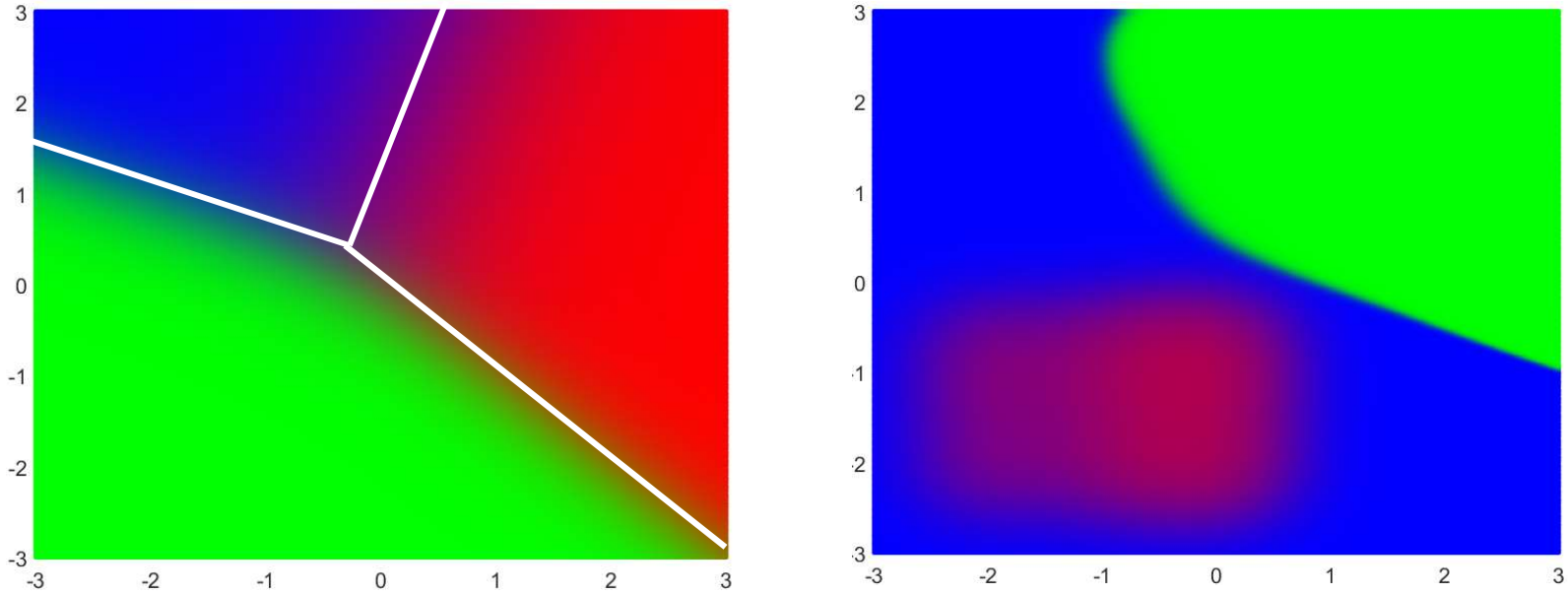
- This too can be learned from training data via maximum likelihood estimation
  - Just like the two-class case

Multi-class logistic regression



- The boundary between adjacent classes is a hyperplane (line)
- The decision boundary for any class is convex polytope with hyperplane segments
  - I.e. still a linear classifier

Multi-class Bayes



- In many classification problems, linear boundaries are not sufficient
- We need to be able to model more complex boundaries
- This too can be supported by the logistic regression classifier

Logistic regression with non-linear boundaries

- The logistic regression can be modified to have non-linear discriminants:

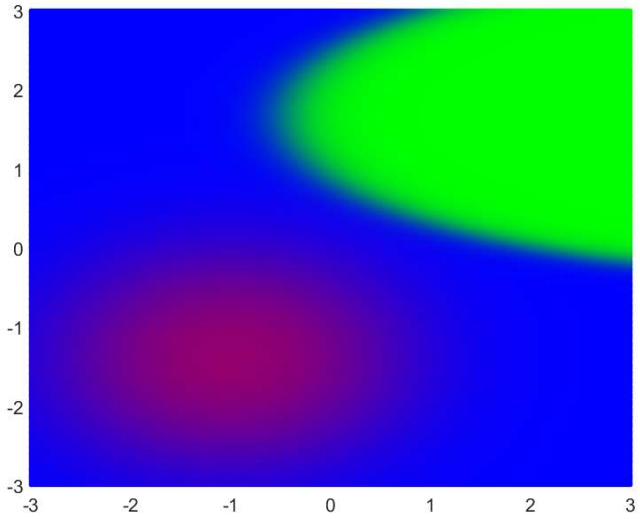
$$P(C|X) = \frac{\exp(f(X;\theta_C))}{\sum_{C'} \exp(f(X;\theta_{C'}))}$$

  - $f(X;\theta_C)$ is the discriminant for class C, and has parameter θ_C
  - The discriminants determine the shape of the decision boundary

- Non-linear discriminants result in non-linear decision boundaries
  - The parameters θ_C for all classes can be learned by maximum likelihood (or MAP) estimation as before

Quadratic discriminant

$$P(C|X) = \frac{\exp(f(X;\theta_C))}{\sum_{C'} \exp(f(X;\theta_{C'}))}$$



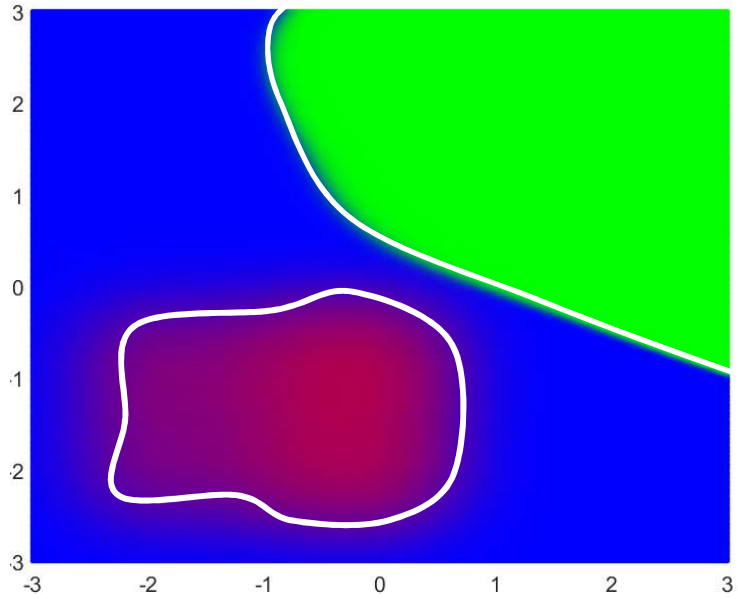
- With quadratic discriminants:

$$f(X;\theta_C) = (X - \alpha_C)\beta_C(X - \alpha_C)^T$$

- Note that decision boundaries are quadratic
- The probability of a class increases (or decreases) as we go away from a boundary

Logistic regression with non-linear boundaries

$$P(C|X) = \frac{\exp(f(X;\theta_C))}{\sum_{C'} \exp(f(X;\theta_{C'}))}$$



- For complex decision boundaries, the function $f(X;\theta_C)$ must be correspondingly complex
- Currently the most successful approach in these cases is to model $f(X;\theta_{C'})$ by a neural network
  - In fact neural networks with soft-max decision layers may be seen as an instance of a logistic regression with a non-linear discriminant
- Topic for another class

Logistic regression with non-linear boundaries

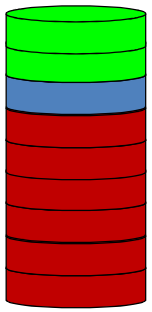
- The logistic regression can be modified to have non-linear discriminants:

$$P(C|X) = \frac{\exp(f(X;\theta_C))}{\sum_{C'} \exp(f(X;\theta_{C'}))}$$

- Note: This can also be viewed as non-linearly transforming the data X into a space where a simple linear logistic regression models posteriors well
- $Z(X) = [f(X;\theta_1)\ f(X;\theta_2)\ \ldots f(X;\theta_K)]^T$
  - I.e. into a space where the data are most linearly separable
  - We will discuss this in a later lecture on neural networks

Problem with modelling P(C|X)

- We have considered modelling the a posteriori probability of the classes directly
- This implicitly assumes that
  - The characteristics of the data for any class remain the same between train and test
  - The *relative proportions* of the classes too remain the same
- Often the second assumption will not hold
  - The data characteristics remain, but the relative proportions change
  - E.g. the shapes of the differently colored coins don't change, but the relative proportions of the colors changes between train and test



- We must then modify our approach to Bayes classification to a *generative* framework

The Bayesian Classifier..

- $\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}} P(C|X)$
  - Choose the class that is most frequent for the given X

$$P(C|X) = \frac{P(C)P(X|C)}{P(X)}$$

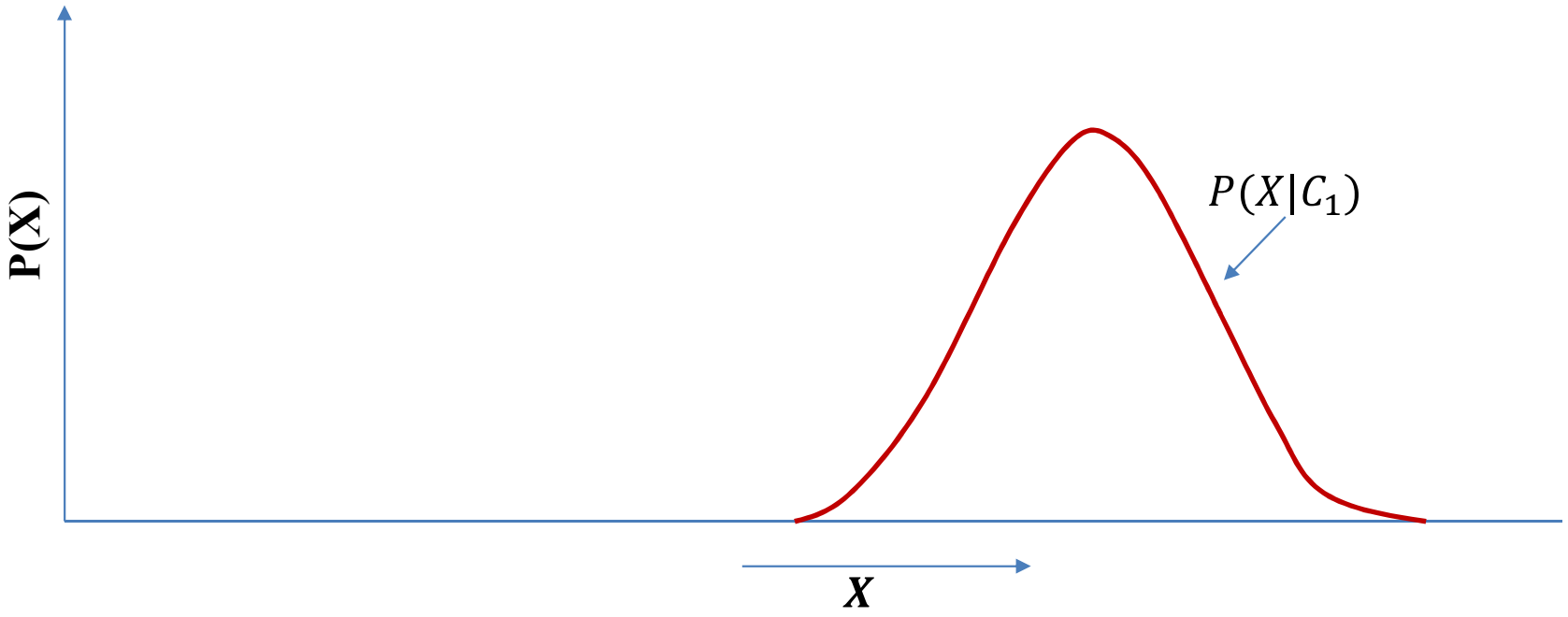
- $\underset{C \in \mathcal{C}}{\operatorname{argmax}} P(C|X) = \underset{C \in \mathcal{C}}{\operatorname{argmax}} P(C)P(X|C)$
  - Choose the class that is most likely to have produced X
    - While accounting for the relative frequency of C

Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(X|C_i)$ measures the probability that a random instance of class C_i will take the value X

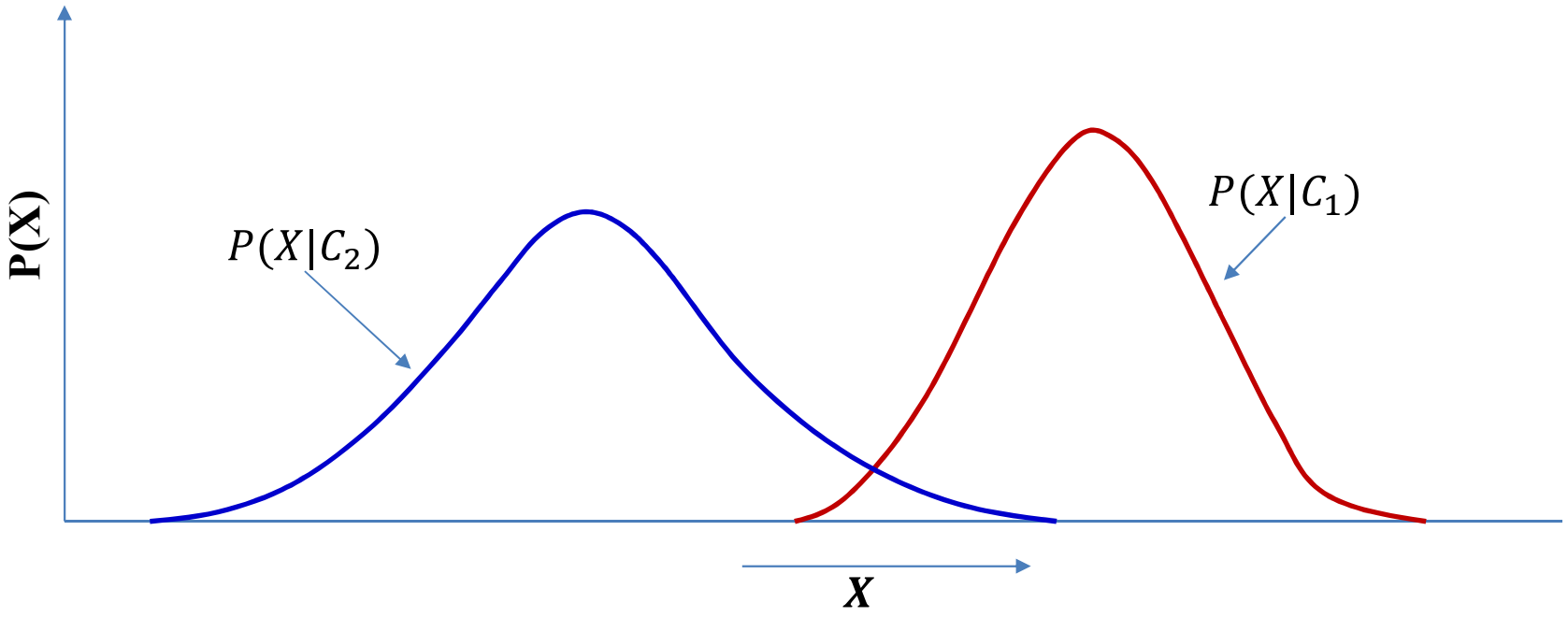


Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(X|C_i)$ measures the probability that a random instance of class C_i will take the value X

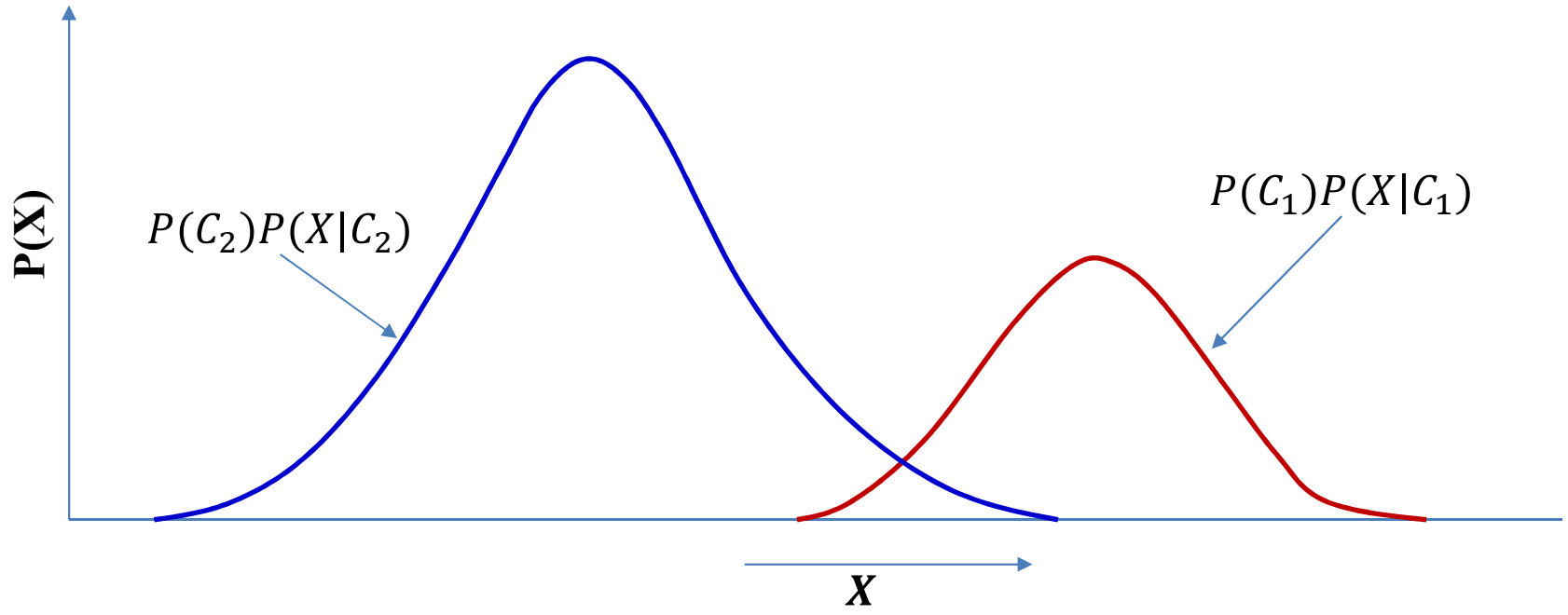


Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(C_i)$ scales them up to match the expected relative proportions of the classes

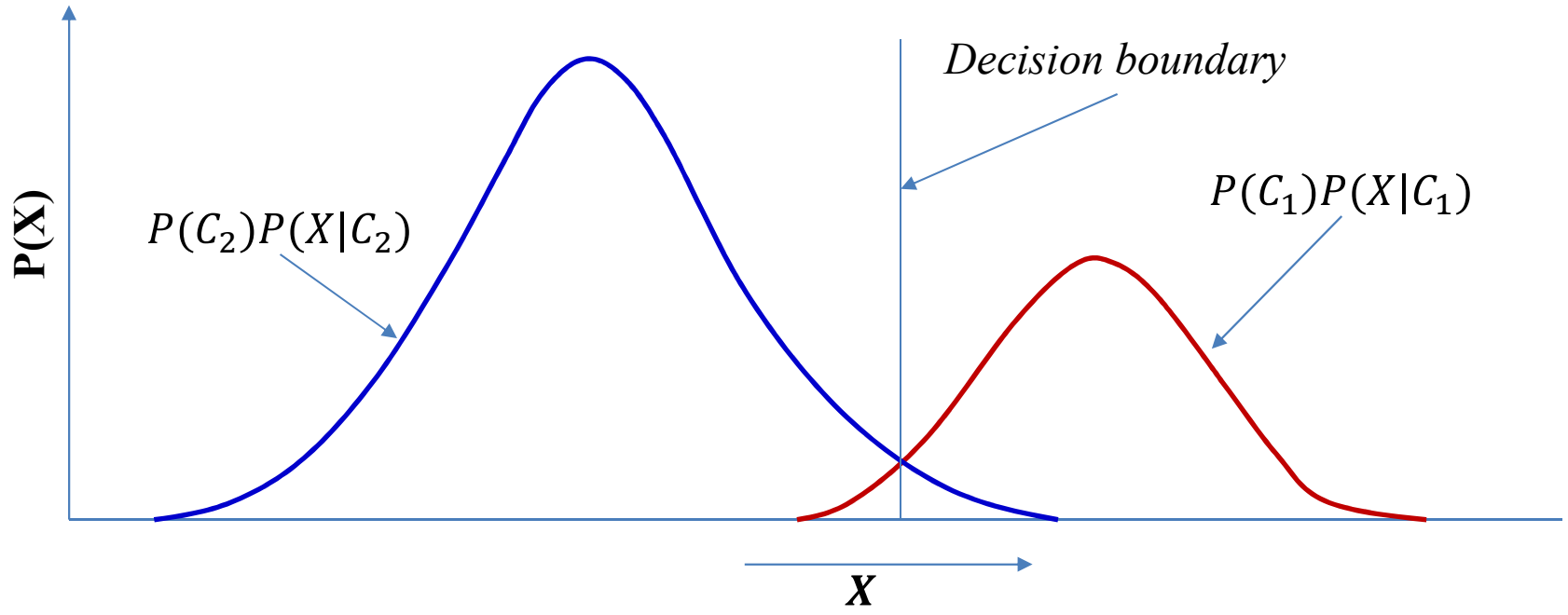


Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(C_i)$ scales them up to match the expected relative proportions of the classes

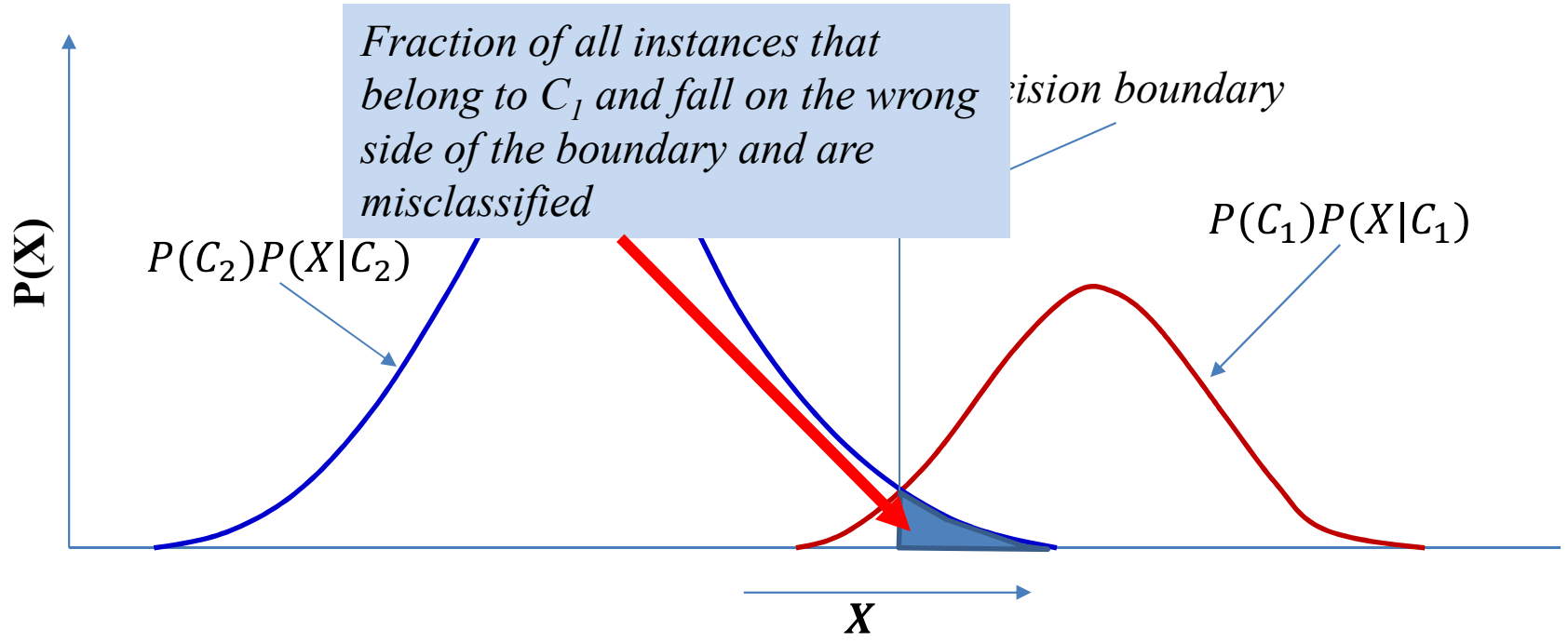


Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(C_i)$ scales them up to match the expected relative proportions of the classes

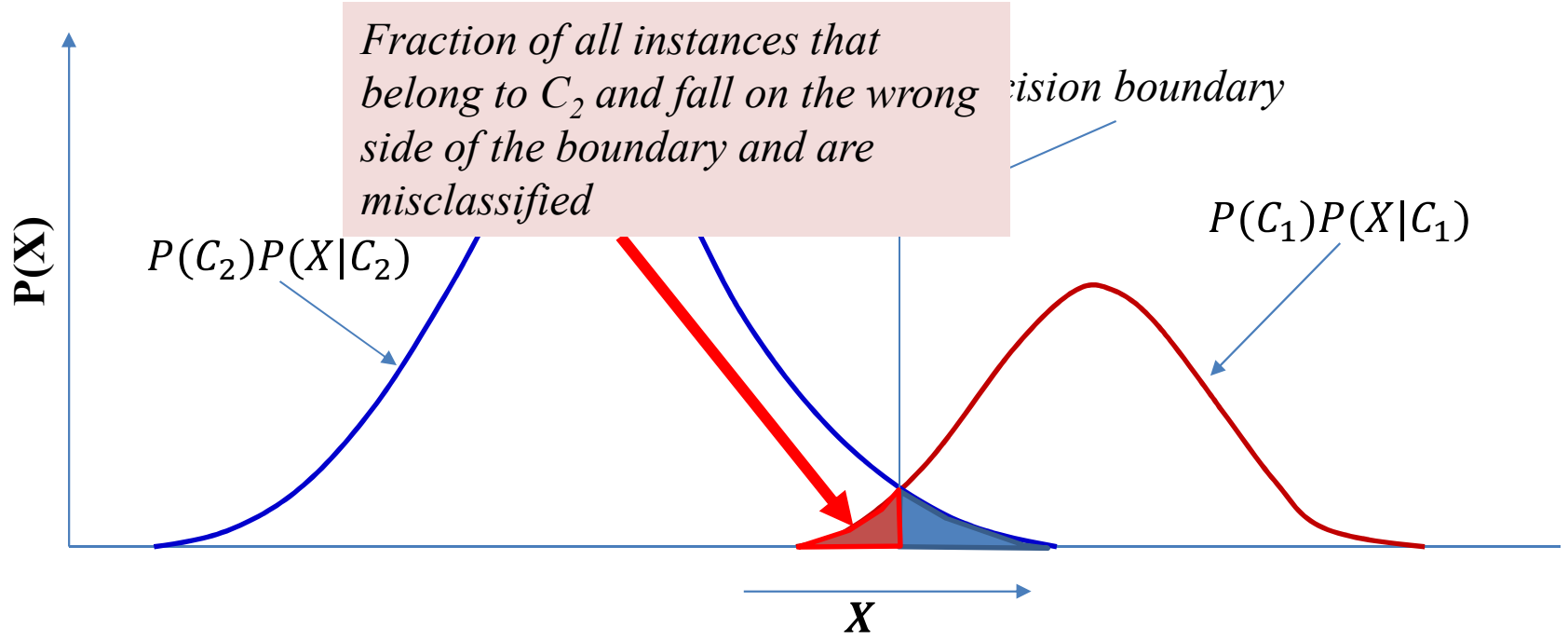


Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(C_i)$ scales them up to match the expected relative proportions of the classes

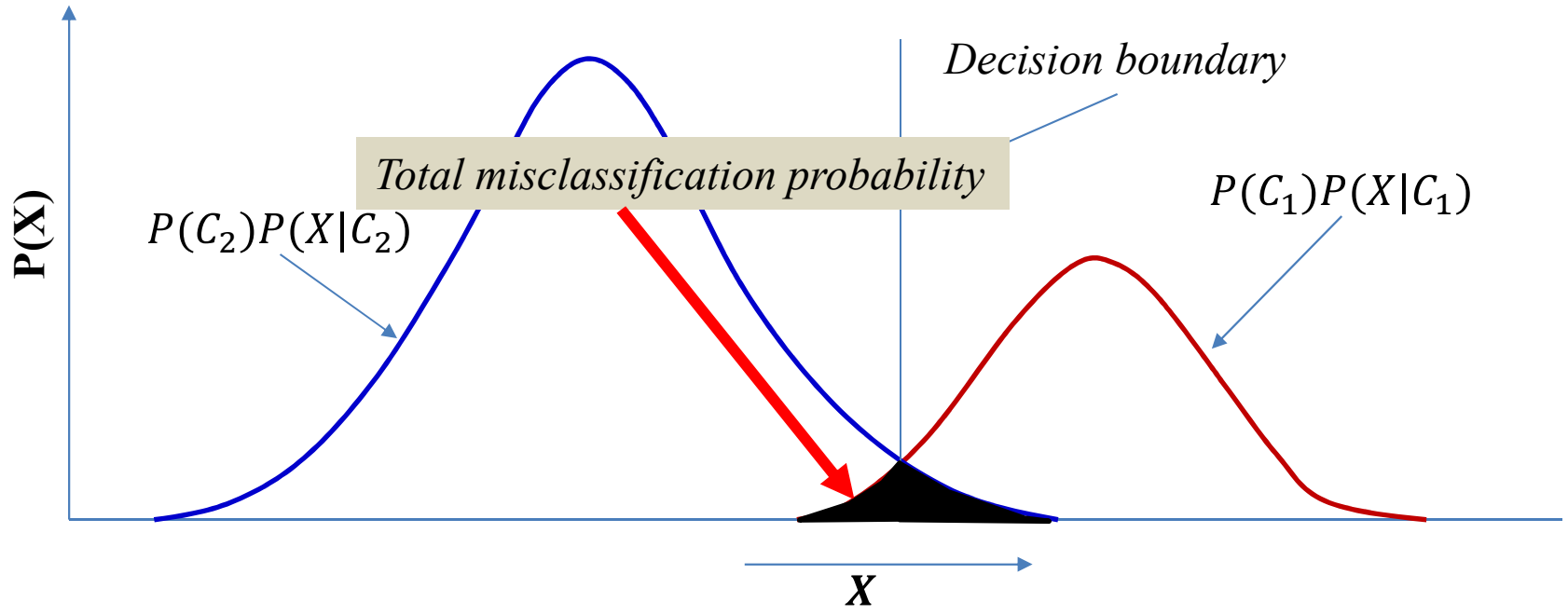


Bayes Classification Rule

- Given a set of classes $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$

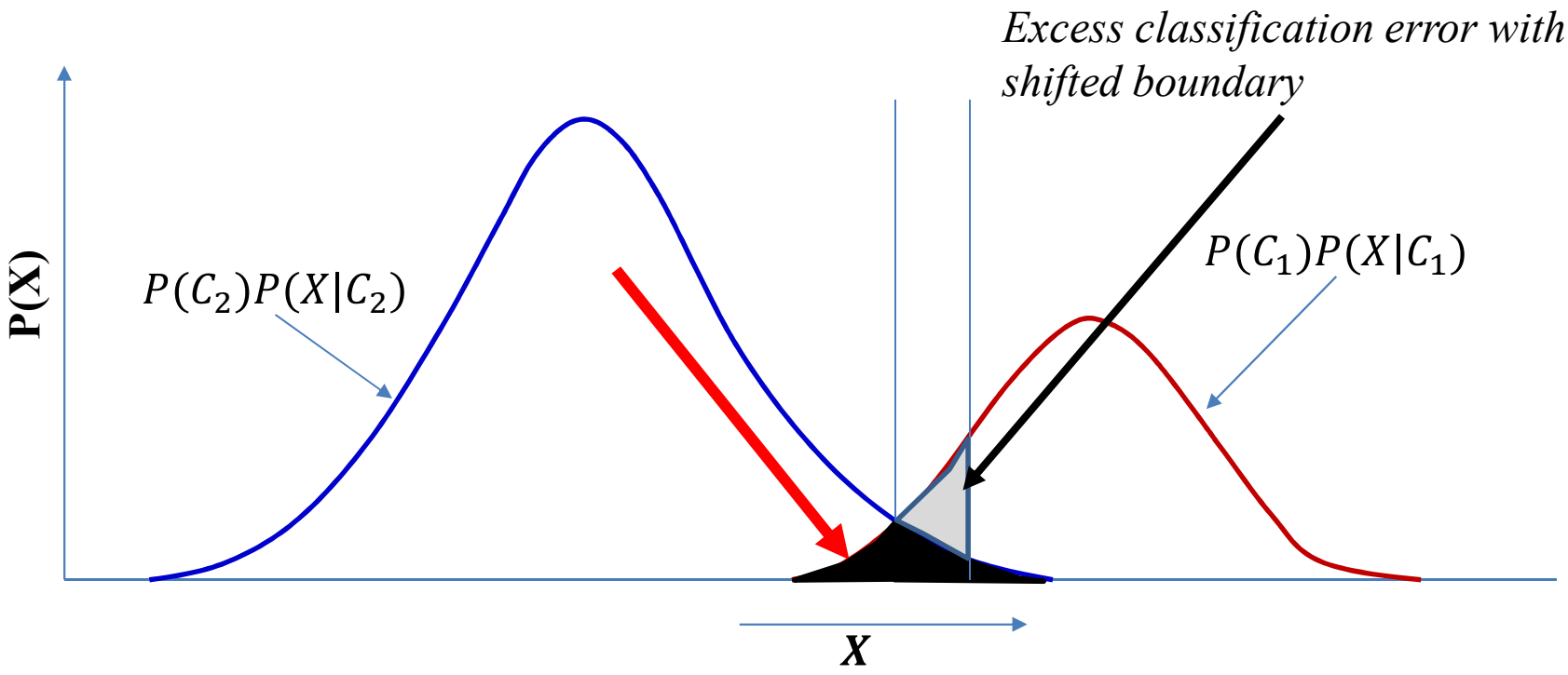
$$\hat{C} = \underset{C \in \mathcal{C}}{\operatorname{argmax}}\, P(C)P(X|C)$$

$P(C_i)$ scales them up to match the expected relative proportions of the classes



Bayes Classification Rule

- The Bayes classification rule is the statistically optimal classification rule
  - Moving the boundary in either direction will always *increase* the classification error



The Bayesian Classifier..

- $\hat{C} = \underset{C \in \mathcal{C}}{\text{argmax}}\, P(C|X) = \underset{C \in \mathcal{C}}{\text{argmax}}\, P(C)P(X|C)$

- We can now directly learn the class-specific statistical characteristics $P(X|C)$ from the training data

- The relative frequency of C, $P(C)$, can be independently adjusted to our expectations of these frequencies in the test data
  - These need not match the training data

Modeling $P(X|C)$

- Challenge: How to learn $P(X|C)$
  - This will not be known beforehand and must be learned from examples of X that belong to class C
- Will generally have unknown and unknowable shape
  - We only observe *samples* of X
- Must make some assumptions about the form of $P(X|C)$

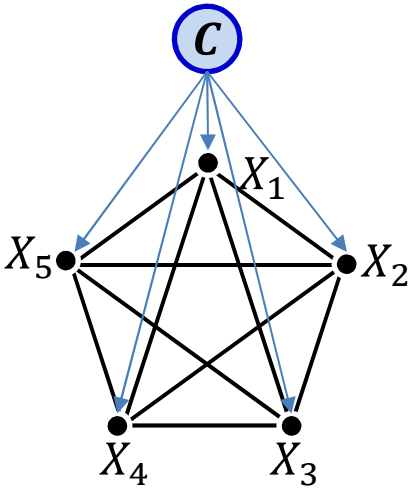
The problem of dependent variables

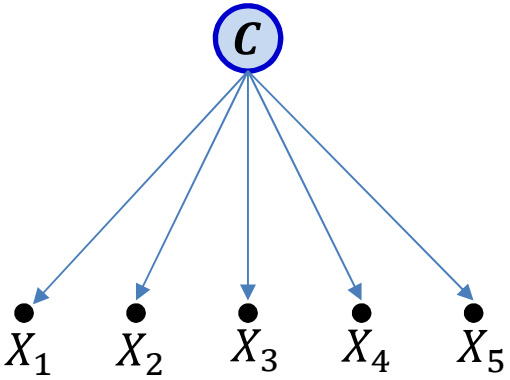
- $P(X|C) = P(X_1, X_2, \ldots, X_D|C)$ must be defined for every combination of $X_1, X_2, \ldots, X_D$
  - Too many parameters to describe explicitly
  - Most combinations unseen in training data
- $P(X|C)$ may have an arbitrary scatter/shape
  - Hard to characterize mathematically
  - Typically do so by assigning a functional form to it

The problem of dependent variables

- $P(X|C) = P(X_1, X_2, \ldots, X_D|C)$ must be defined for every combination of $X_1, X_2, \ldots, X_D$
  - Too many parameters to describe explicitly
  - Most combinations unseen in training data
- $P(X|C)$ may have an arbitrary scatter/shape
  - Hard to characterize mathematically
  - Typically do so by assigning a functional form to it

The Naïve Bayes assumption





- Assume all the components are independent of one another
  - The joint probability is the product of the *marginal*

$$P(X|C) = P(X_1, X_2, \ldots, X_D|C) = \prod_i P(X_i|C)$$

- Sufficient to learn marginal distributions $P(X_i|C)$
  - The problem of having to observe all combinations of $X_1, X_2, \ldots, X_D$ never arises

Naïve Bayes – estimating $P(X_i|C)$

- $P(X_i|C)$ may be estimated using conventional maximum likelihood estimation
  - Given a number of training instances belonging to class C
    - Select the i-th component of all instances
    - Estimate $P(X_i|C)$
  - For discrete-valued X_i this will be a multinomial distribution
  - For continuous valued X_i a form must be assumed
    - E.g Gaussian, Laplacian etc

Naïve Bayes – Binary Case

$$P(X|C) = P(X_1, X2, ..., X_D|C)$$

if $X_i \in \{0, 1\}$ $\quad$ $2^D - 1$ parameters for each C

$$P(X|C) = P(X_1|C) \cdot ... \cdot P(X_D|C)$$

D parameters for each C

The problem of dependent variables

- $P(X|C) = P(X_1, X_2, \ldots, X_D|C)$ must be defined for every combination of $X_1, X_2, \ldots, X_D$
  - Too many parameters
  - Most combinations unseen in training data
- $P(X|C)$ may have an arbitrary scatter/shape
  - Hard to characterize mathematically
  - Typically do so by assigning a functional form to it

Assigning a functional form to P(X|C)

- Assign a functional form to P(X|C)

- Common assumptions:
  - Unimodal forms: Gaussian, Laplacian
  - Multimodal forms: Gaussian mixtures
  - Time series: Hidden Markov models
  - Multi-dimensional structures: Markov random fields

Assigning a functional form to P(X|C)

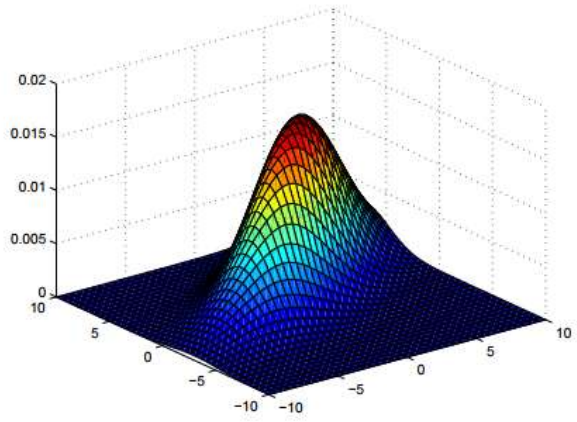
- Assign a functional form to P(X|C)

- Common assumptions:
  - Unimodal forms: **Gaussian**, Laplacian
    - Most common of all
  - Multimodal forms: Gaussian mixtures
  - Time series: Hidden Markov models
  - Multi-dimensional structures: Markov random fields

Gaussian Distribution

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

$\mathbf{x} \in \mathbb{R}^n$



μ Mean Vector

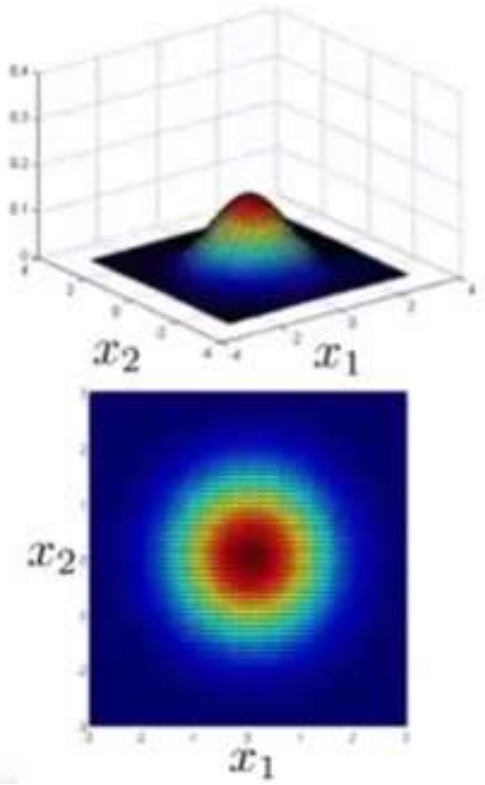
Σ Covariance Matrix

$n \times n$

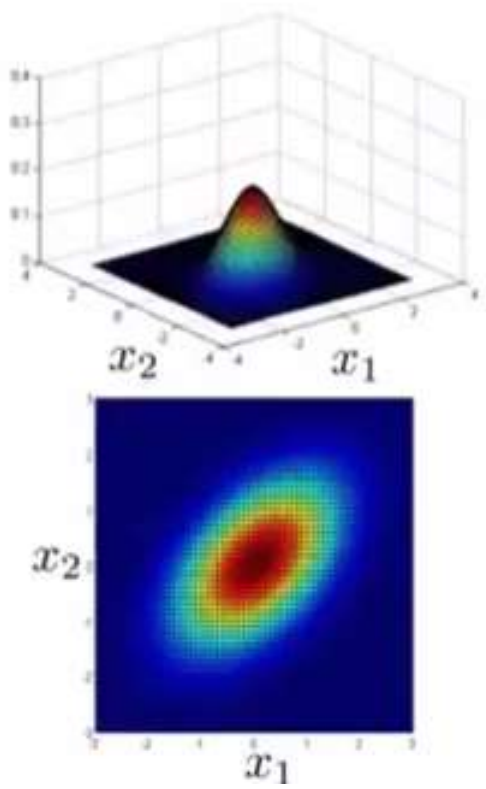
→ - Symmetric
- Positive Definite

Gaussian Distribution

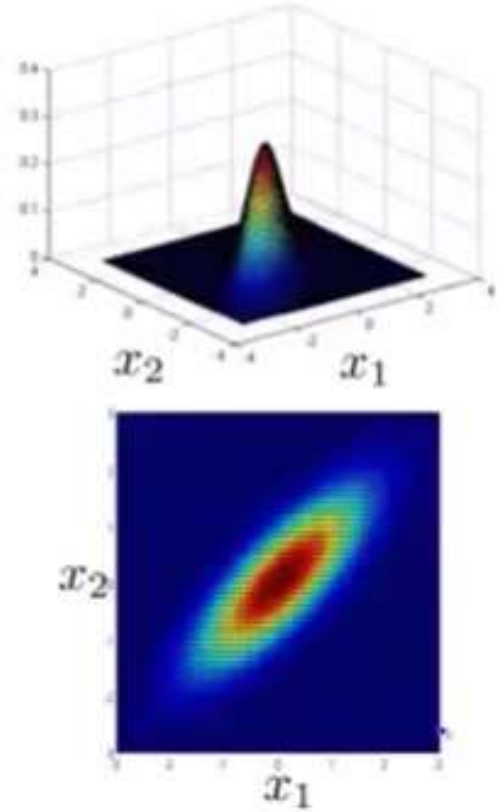
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

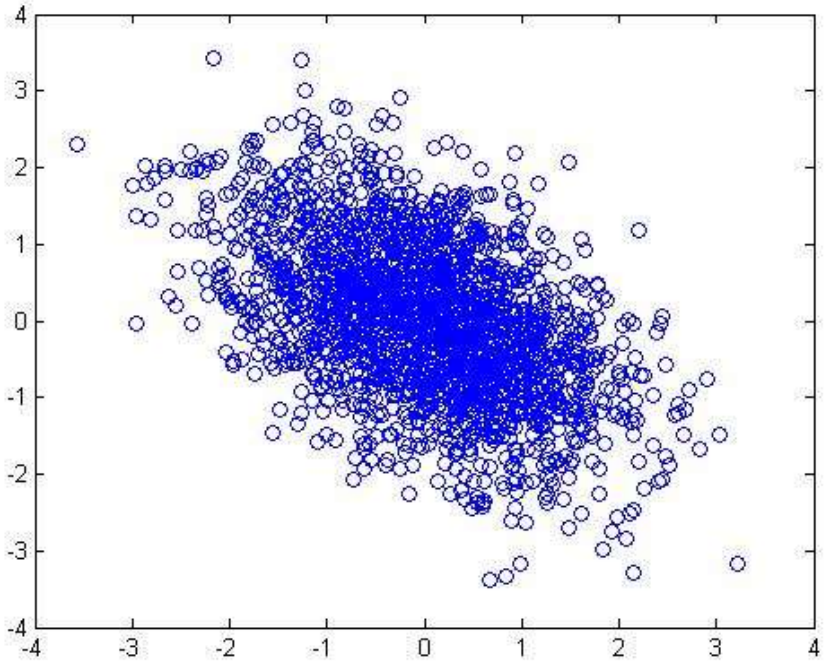


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



Parameter Estimation

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



Maximum Likelihood Estimators

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^\top$$

Gaussian classifier

$$p(\mathbf{x}, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \mathbf{\Sigma^{-1}}(\mathbf{x}-\mu)\right)$$

Different Classes, different Gaussians

$$p(\mathbf{x}|C_1) = p(\mathbf{x}, \mu_1, \Sigma_1) = \frac{1}{(2\pi)^{n/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{1})^\top \mathbf{\Sigma_1^{-1}}(\mathbf{x}-\mu_\mathbf{1})\right)$$

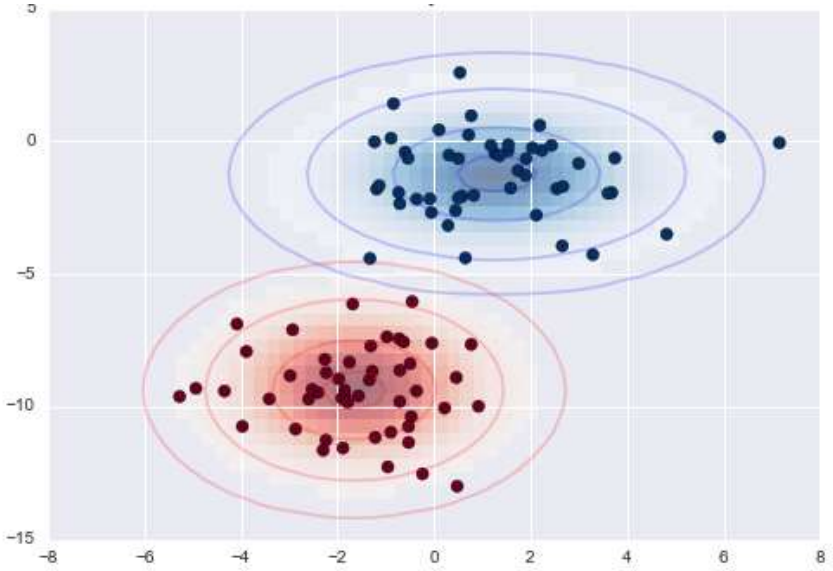
$$p(\mathbf{x}|C_2) = p(\mathbf{x}, \mu_2, \Sigma_2) = \frac{1}{(2\pi)^{n/2}|\Sigma_2|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{2})^\top \mathbf{\Sigma_2^{-1}}(\mathbf{x}-\mu_\mathbf{2})\right)$$

$$\vdots$$

$$p(\mathbf{x}|C_k) = p(\mathbf{x}, \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_k)^\top \Sigma_k^{-1}(\mathbf{x}-\mu_k)\right)$$

Gaussian Classifier

- For each class we need:
  - Mean Vector
  - Covariance Matrix
- Training
  - "Fit" a Gaussian to each class
    - Find the best Gaussian to explain the distribution for the class



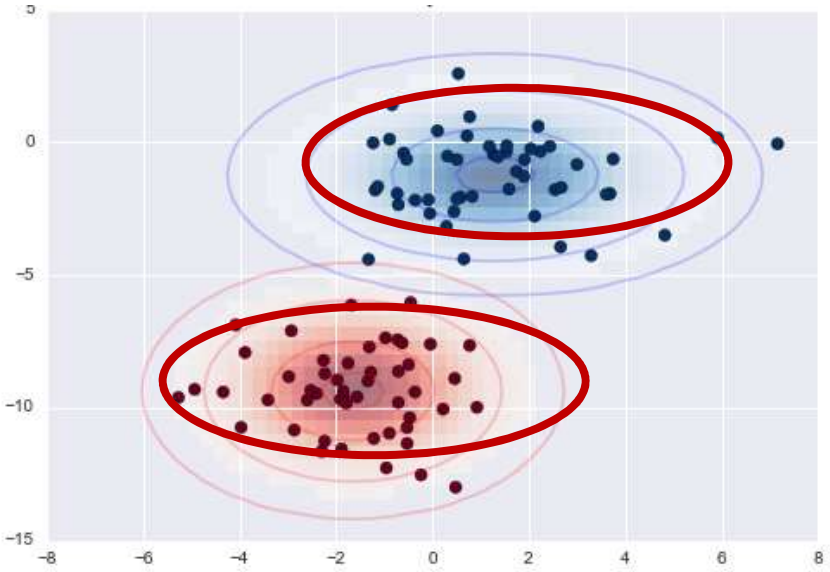
- Classification:

$$\arg\max_i P(C_i)p(\mathbf{x}, \mu_i, \Sigma_i)$$

- Problem:
  - Many parameters to train!
  - Dominated by covariance: for D-dimensional data the covariance matrices requires D^2 parameters each
  - For N_c classes, a total of $N_c D^2$ parameters

Homo-skedasticity assumption

- Assume all distributions have the same covariance
  - $\Sigma_i = \Sigma \; \forall \; i$
  - Assumption, may not be true
  - But still works in many cases



- Fewer parameters to train
  - One common covariance matrix for all classes
  - Only D^2 total parameters
    - As opposed to $N_c D^2$ if each class has its own covariance matrix

Homo-skedastic Gaussians

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_K = \Sigma$$

- For the binary classification case ($K = 2$)

Decision boundary:

$$P(X|C_2)P(C_2) = P(X|C_2)P(C_2)$$

$$\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right) \cdot p(C_1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right) \cdot p(C_2)$$

$$\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right) \cdot p(C_1) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right) \cdot p(C_2)$$

Homo-skedastic Gaussians

$$\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^\top\Sigma^{-1}(\mathbf{x}-\mu_1)\right)\cdot p(C_1) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_2)^\top\Sigma^{-1}(\mathbf{x}-\mu_2)\right)\cdot p(C_2)$$

taking log

$$\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^\top\Sigma^{-1}(\mathbf{x}-\mu_1)\right)+\log p(C_1) = \left(-\frac{1}{2}(\mathbf{x}-\mu_2)^\top\Sigma^{-1}(\mathbf{x}-\mu_2)\right)+\log p(C_2)$$

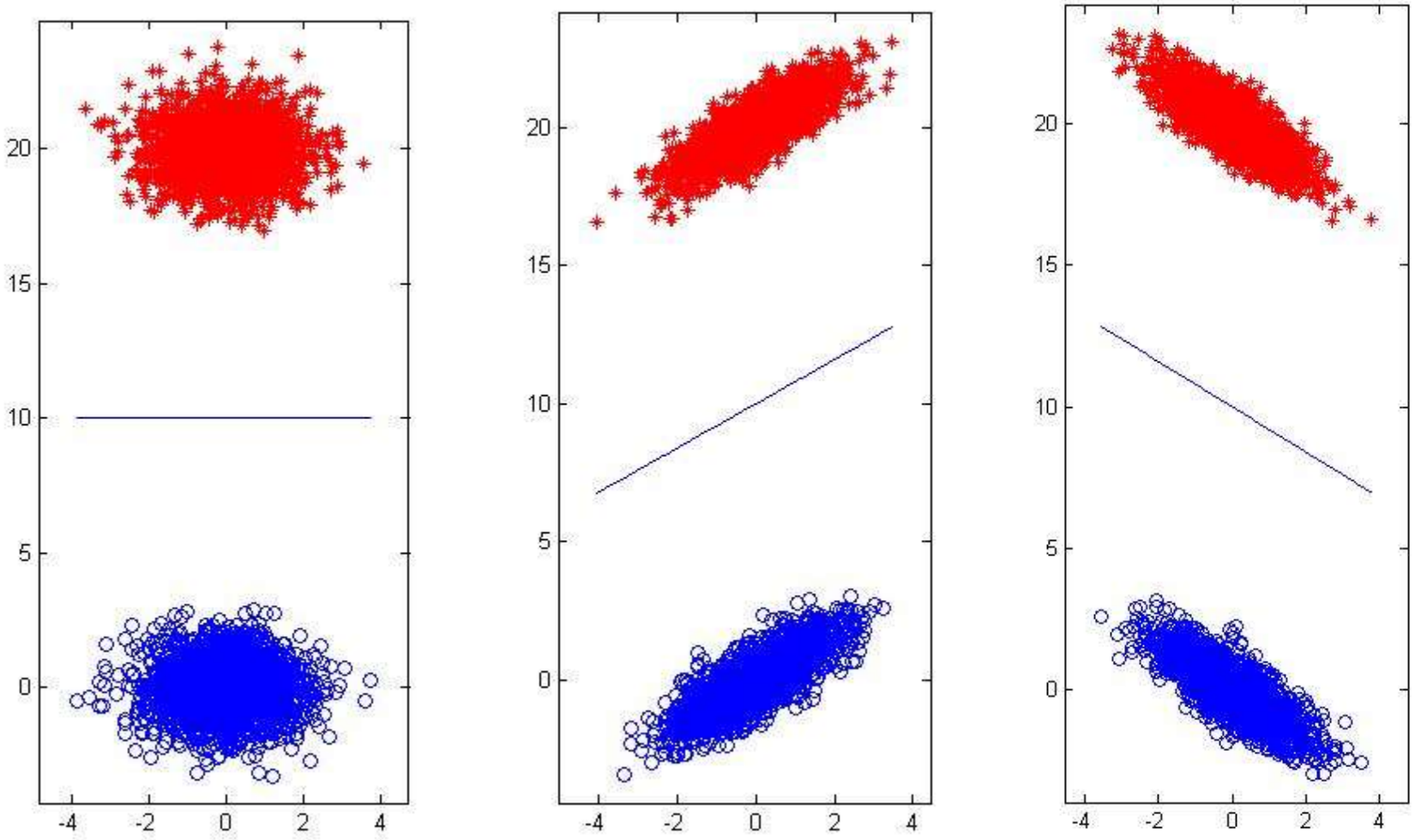
$$\mu_1^\top\Sigma^{-1}\mathbf{x}-\frac{1}{2}\mu_1^\top\Sigma^{-1}\mu_1+\log\cdot p(C_1) = \mu_2^\top\Sigma^{-1}\mathbf{x}-\frac{1}{2}\mu_2^\top\Sigma^{-1}\mu_2+\log p(C_2)$$

$$(\mu_1-\mu_2)^\top\Sigma^{-1}\mathbf{x}-\frac{1}{2}\mu_1^\top\Sigma^{-1}\mu_1+\frac{1}{2}\mu_2^\top\Sigma^{-1}\mu_2+\log p(C_1)-\log p(C_2)=0$$

$$\boxed{\mathbf{w}^\top\mathbf{x}+c=0}$$

Linear Boundary!!!

Homo-skedastic Gaussians



Homo-skedastic Gaussians, K > 2

- Case K > 2 (more than two classes)
- Classification performed as:

$$\underset{i}{\operatorname{argmax}}\, P(C_i)p(\mathbf{x}, \mu_i, \Sigma_i)$$

- Taking logs and ignoring the common constant

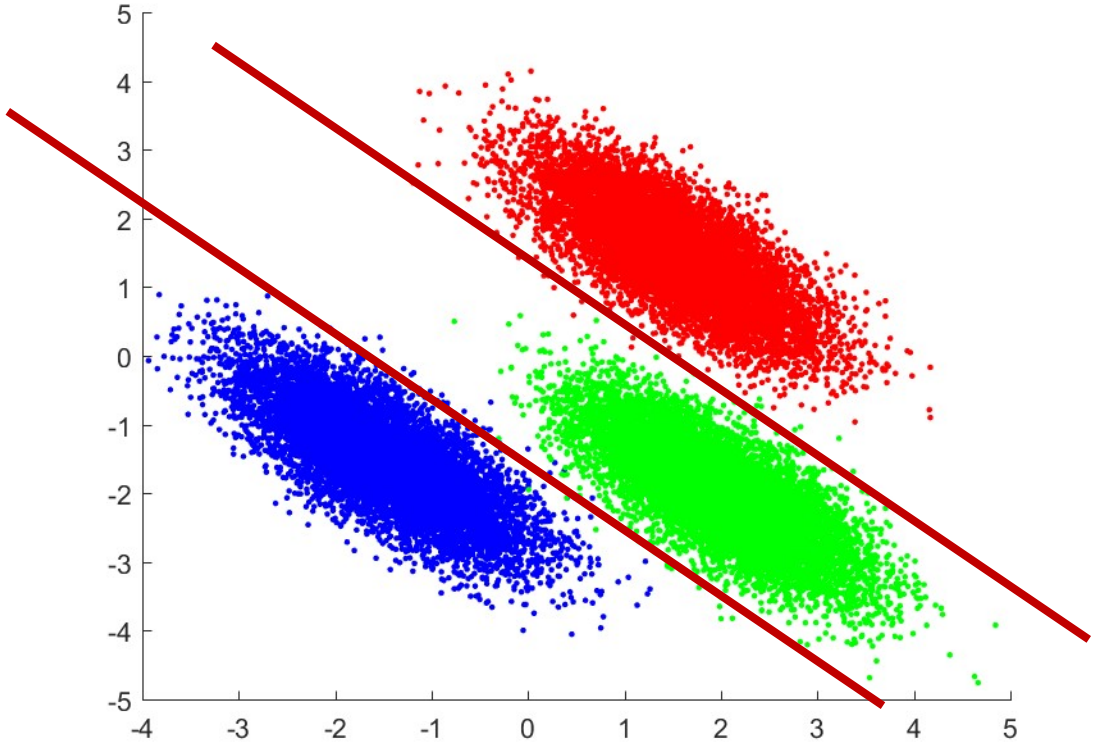
$$\underset{i}{\operatorname{argmax}} -\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma^{-1}(\mathbf{x}-\mu_i) + \log P(C_i)$$

- Expanding out and ignoring common terms

$$\underset{i}{\operatorname{argmax}} -\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mu_i - \frac{1}{2}\mu_i{}^T\Sigma^{-1}\mu_i + \log P(C_i)$$

- This is just a linear classifier

Homo-skedastic Gaussians, K > 2



- Decision boundaries for

$$\underset{i}{\operatorname{argmax}}\, P(C_i)p(\mathbf{x}, \mu_i, \Sigma_i)$$

- Linear classifier: Decision boundaries are hyperplanes

Homo-skedastic Gaussians, K > 2

- Case K > 2 (more than two classes)
- Classification performed as:

$$\underset{i}{\operatorname{argmax}}\, P(C_i)p(\mathbf{x}, \mu_i, \Sigma_i)$$

- Taking logs and ignoring the common constant

$$\underset{i}{\operatorname{argmax}} -\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma^{-1}(\mathbf{x}-\mu_i) + \log P(C_i)$$

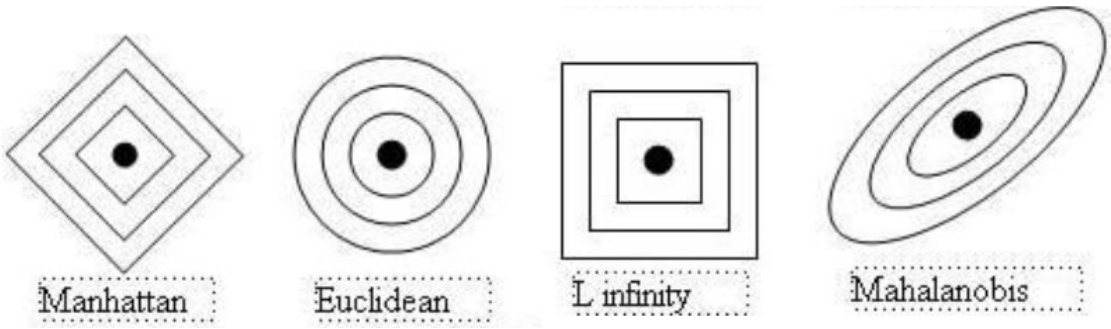
- Changing the sign and rewriting it as argmin

$$\underset{i}{\operatorname{argmin}} \quad (\mathbf{x}-\mu_i)^T\Sigma^{-1}(\mathbf{x}-\mu_i) - 2\log P(C_i)$$

Homo-skedastic Gaussians

Mahalanobis Distance

$$D_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top \Sigma^{-1} (\mathbf{x}-\mathbf{y})}$$



- A Gaussian Classifier with common Covariance Matrix is similar to a Nearest Neighbor Classifier
- Classification corresponds to the nearest mean vector

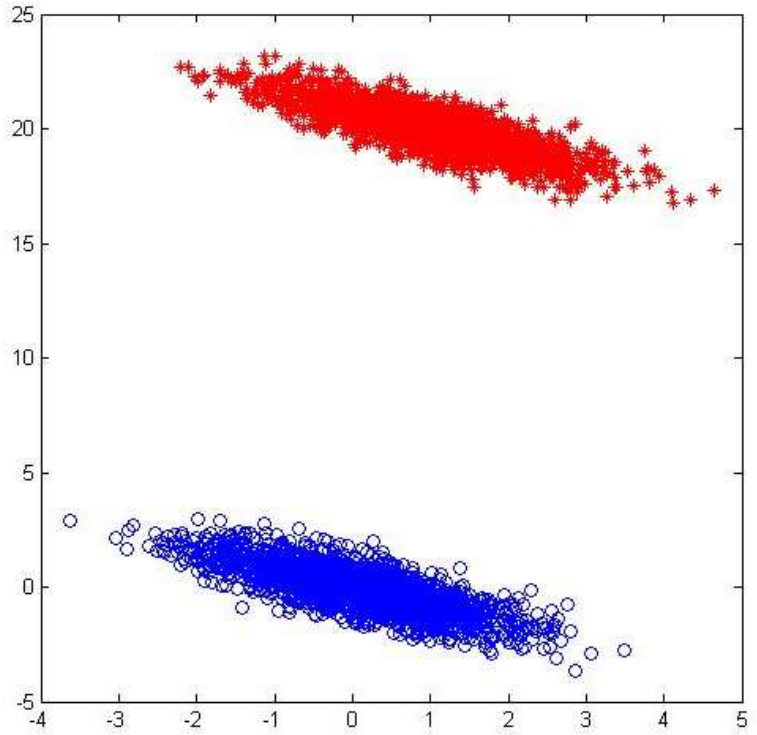
How to estimate the Covariance Matrix?

- Maximum likelihood estimate of covariances of individual classes:

$$\Sigma_C = \frac{1}{N_C}\sum_{i=1}^{N_C}\left(\mathbf{x}_i^{(c)} - \mu_C\right)\left(\mathbf{x}_i^{(c)} - \mu_C\right)^T$$

- Estimate of common covariance for all classes

$$\Sigma = \frac{1}{\sum_{C\prime=1}^{K} N_{C\prime}}\sum_{C=1}^{K} N_C\Sigma_C$$



Hetero skedastic Gaussians..

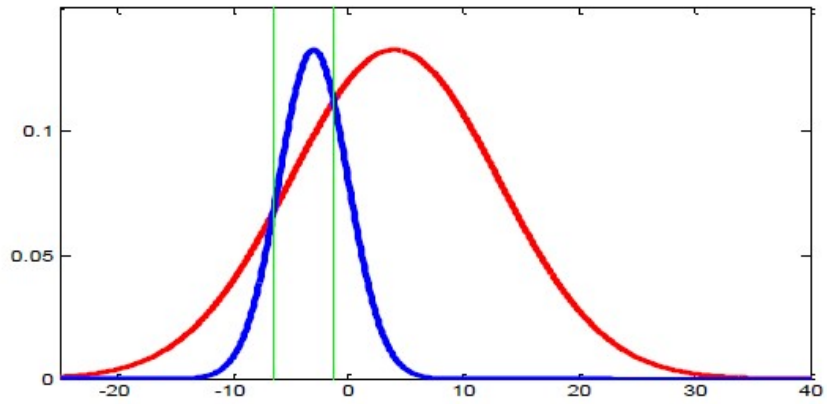
- Homoskedastic Gaussians do not capture non-linear decision boundaries
- Also, the assumption that all Gaussians have the same covariance is questionable

- Permitting each Gaussian to have its own covariance results in non-linear decision boundaries
  - "Hetero skedastic" Gaussians

Hetero-skedastic Gaussians

Different Covariance Matrices

1D case. K = 2



Decision Boundary

$$p(\mathbf{x}|C_1)p(C_1) = p(\mathbf{x}|C_2)p(C_2)$$

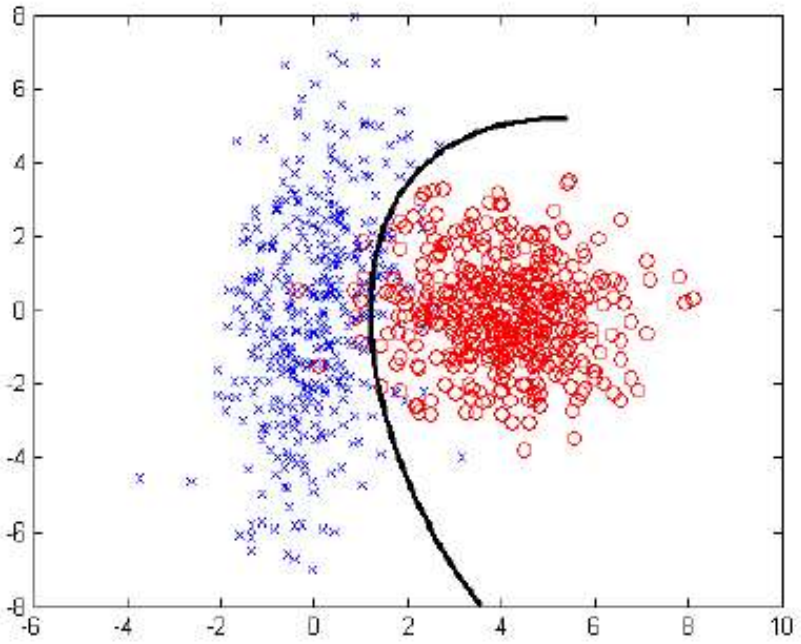
$$\log\frac{\sigma_2}{\sigma_1} + \frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2 - \frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2 - \log\frac{P(C_2)}{P(C_1)} = 0$$

$$(x - x_1)(x - x_2) = 0$$

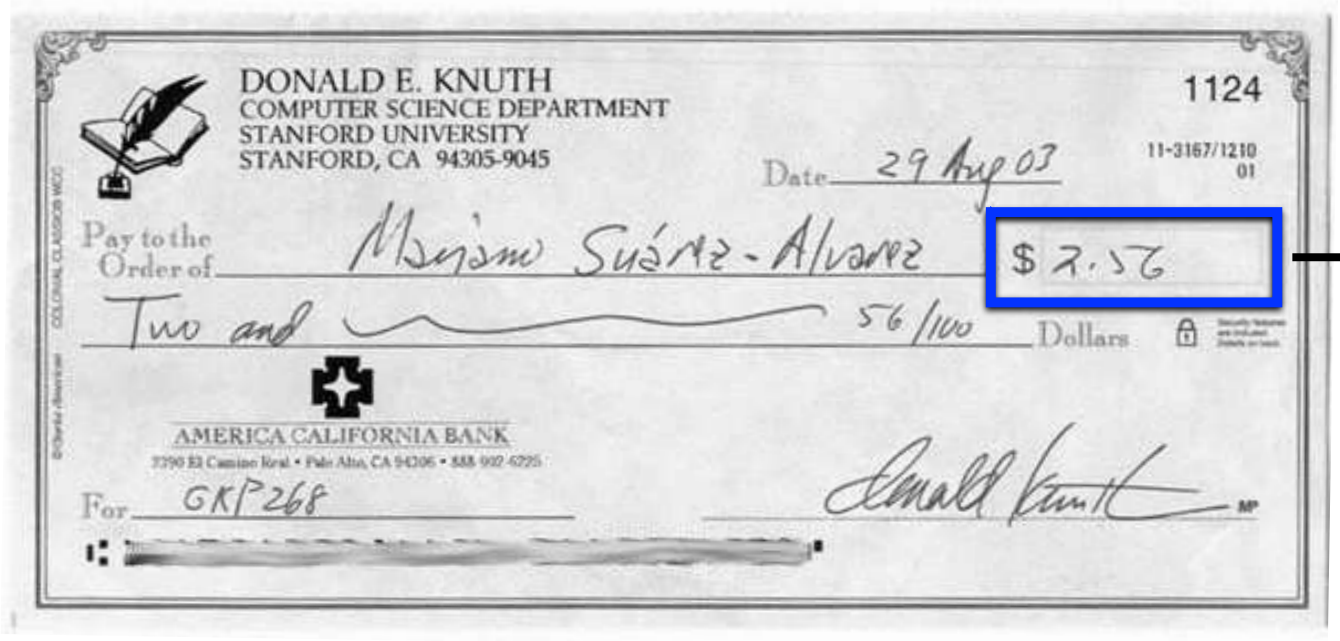
Quadratic Boundary!!!

Hetero-skedastic Gaussians

$$\mathbf{x}_1 \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 1\\1 & 8\end{bmatrix}\right), \ \mathbf{x}_2 \sim \mathcal{N}\left(\begin{bmatrix}4\\0\end{bmatrix}, \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}\right)$$

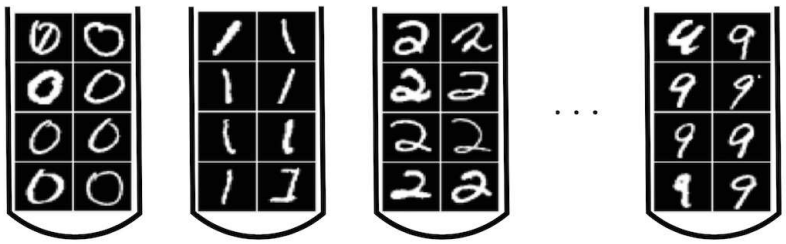


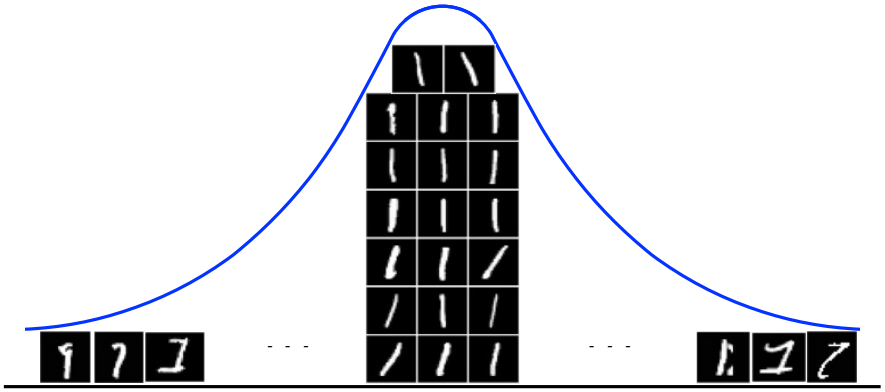
Digit recognition



$ 2.56

Gaussian Classifier for Digit recognition

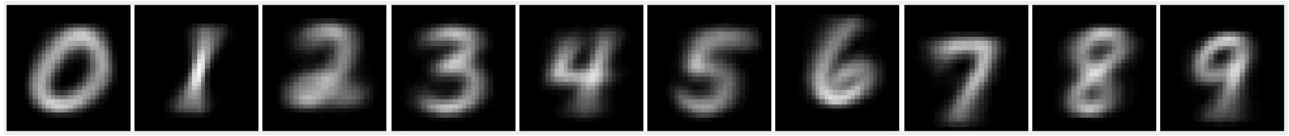




$$p(\mathbf{x}, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mu)\right)$$

Showing the average of each digit

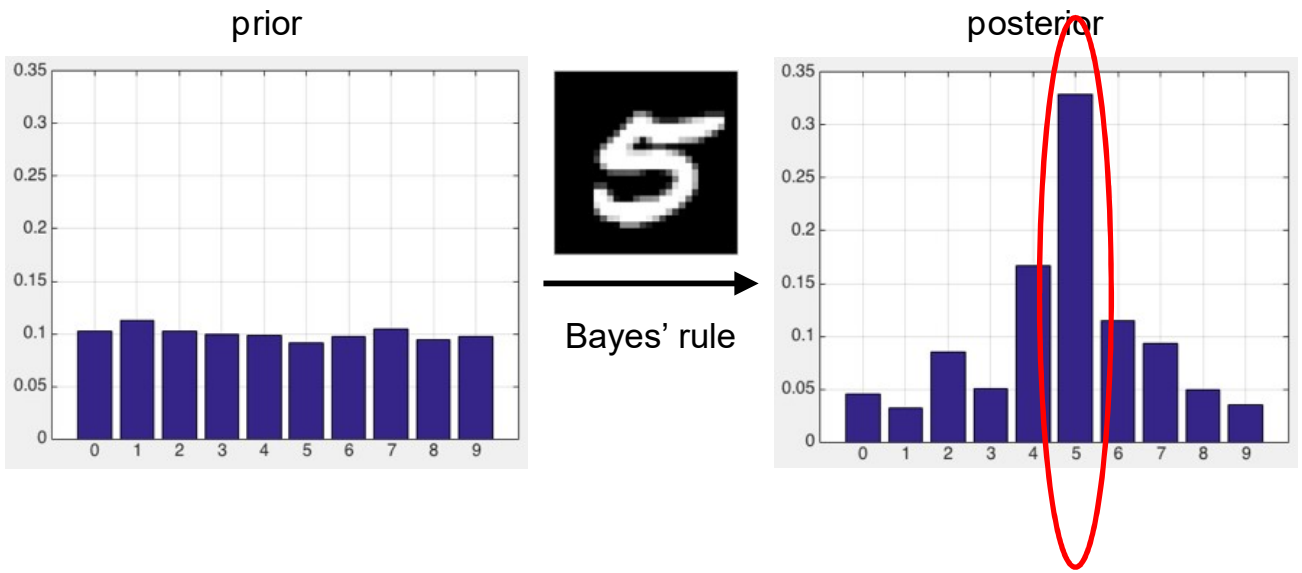
- Average digit



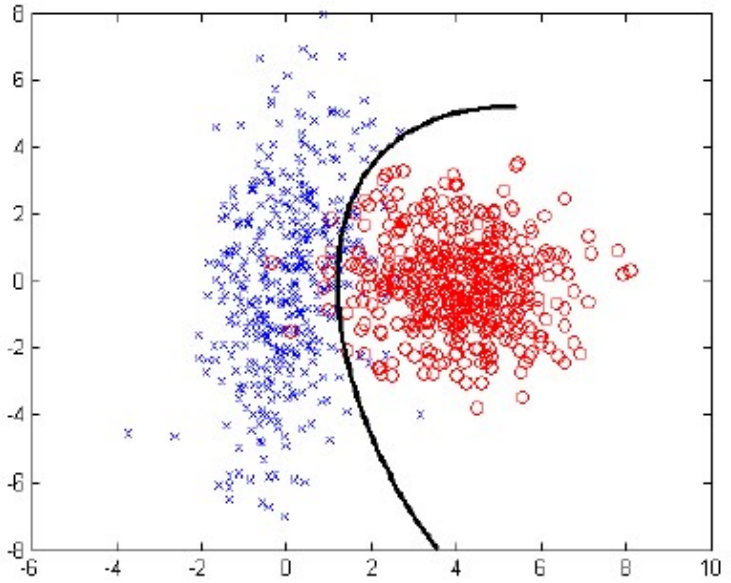
Bayes classifcation

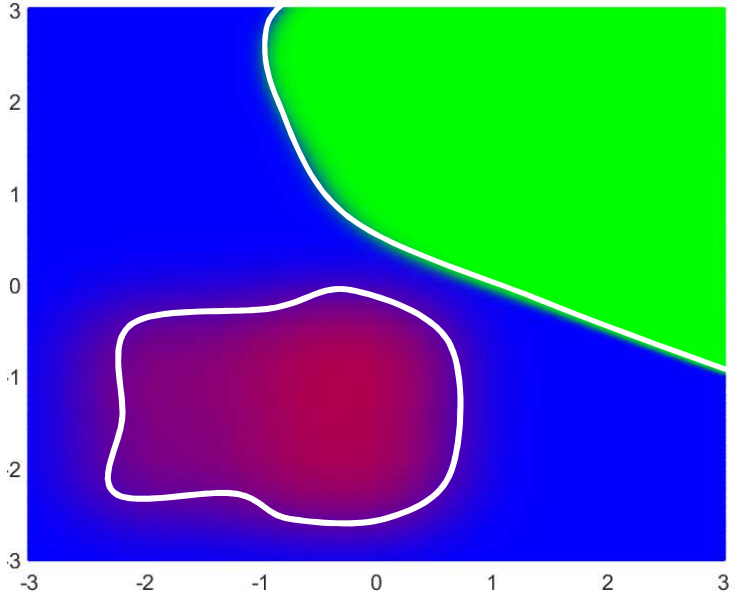
$$\arg\max_i \; p(C_i)p(\mathbf{x}, \mu_i, \Sigma)$$

- Normalize the Posterior



Inadequacy of Gaussian classifiers



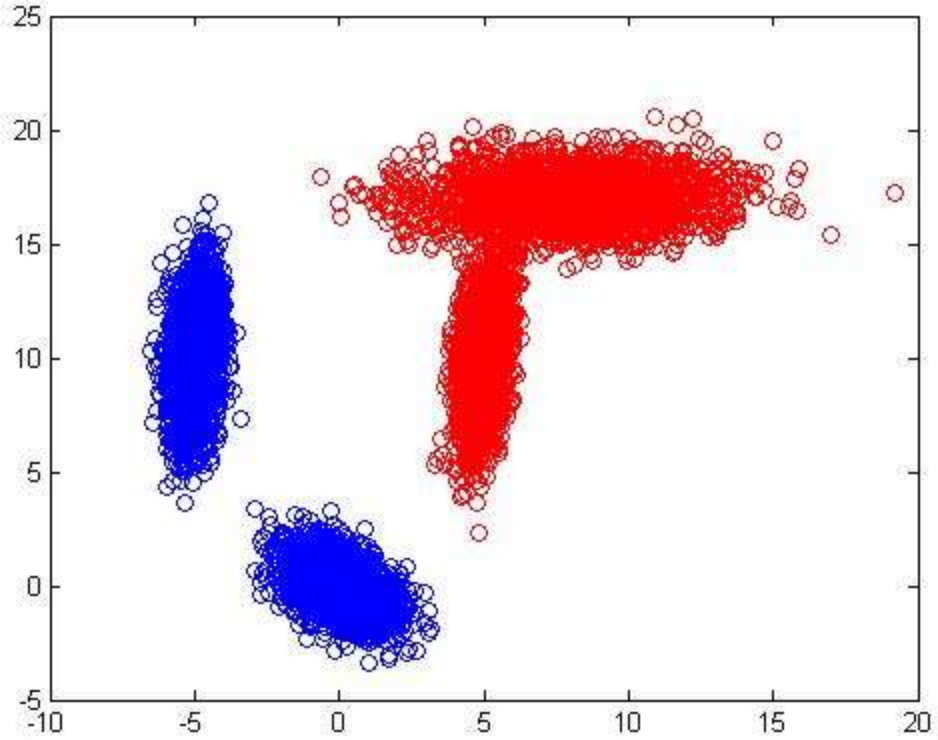


- Gaussian classifiers can only capture simple linear or quadratic decision boundaries
- Often, the decision boundaries required are more complex
- In this case we must employ a *Gaussian Mixture* classifier

$$\underset{i}{\operatorname{argmax}}\, P(C_i)p(\mathbf{x}|C_i)$$

- $p(\mathbf{x}|C_i)$ is modelled by a Gaussian mixture

GMM classifier



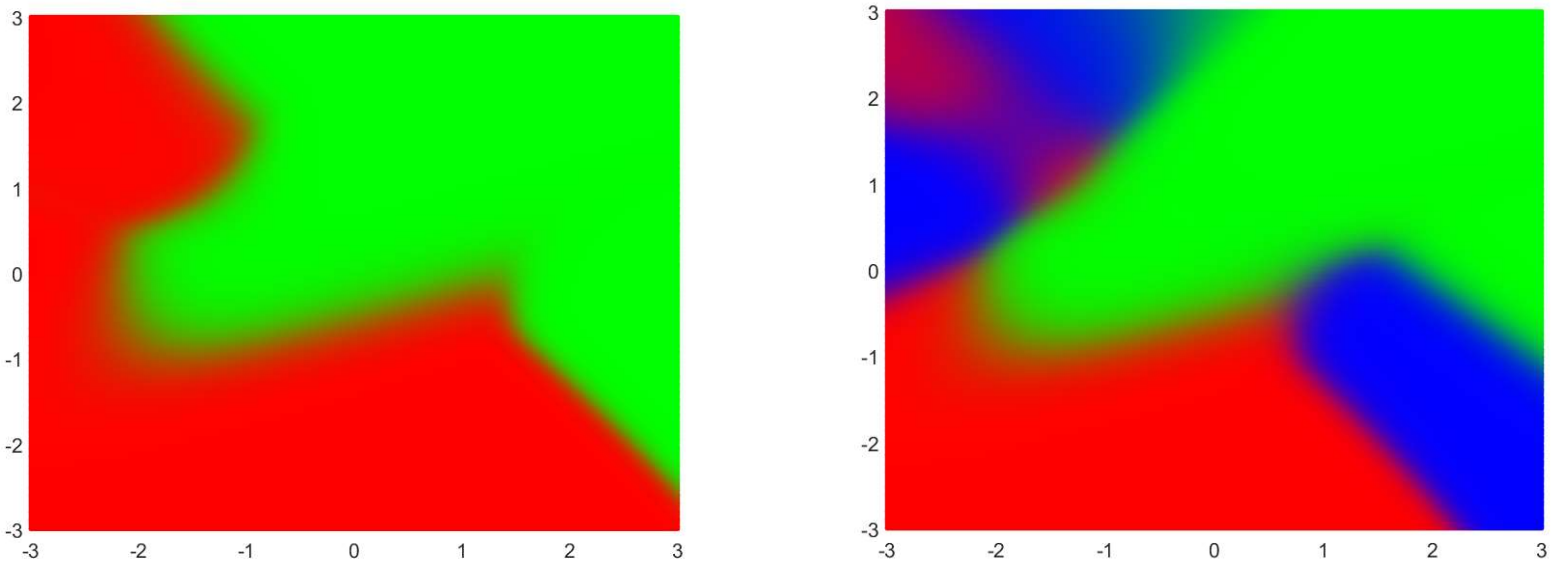
- For each class, train a GMM (with EM)

$$p(\mathbf{x}|C_i) = \sum_{j=1}^{K} \pi_j^{(i)} p(\mathbf{x}|\mu_j^{(i)}, \Sigma_j^{(i)})$$

- Classify according to

$$\arg\max_i p(\mathbf{x}|C_i) \cdot p(C_i)$$

Bayesian Classification with Gaussian Mixtures



- Plotting $P(C_i)p(\mathbf{x}|C_i)$ for all classes
  - Left: Two-class classification, Right: Three-class classification
  - Each class modelled by a mixture of three Gaussians
- Note the complex decision boundary

Estimating P(C)

$$\underset{i}{\operatorname{argmax}}\, P(C_i)p(\mathbf{x}, \mu_i, \Sigma_i)$$

- Have not explained where the class prior $P(C_i)$ comes from

- This can be dependent on the test data
- Typical solutions:
  - Estimate from training data
  - Optimize on development or held-out test data
  - Heuristic guess
  - Conservative estimates
    - Set the prior of classes that have high cost if incorrectly detected to be low
    - Set prior of classes that have high cost if incorrectly missed to be low
    - Etc..

Topics not covered

- Maximum a posteriori estimation
  - When we make assumptions about the parameters (means, covariances) themselves
- MAP *regression* with Gaussians
  - Predicting continuous-valued RVs assuming Gaussian distributions
- MAP regression with Gaussian Mixtures
  - Predicting continuous-valued RVs with Gaussian mixture distributions
- Time-series and other structured data
  - Partially covered in a later lecture