Supervised classification

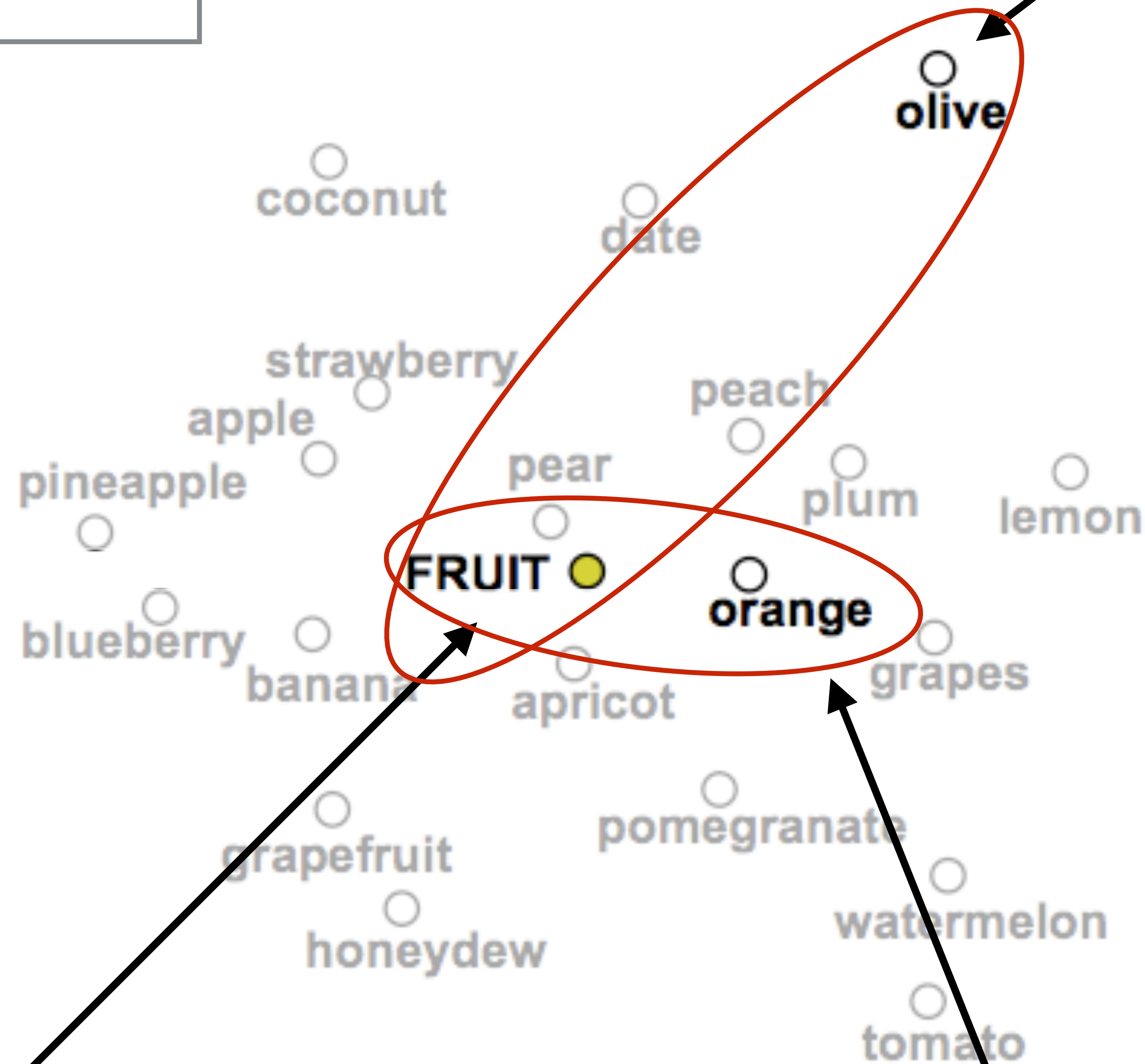
Computational Cognitive Science 2014
Dan Navarro

From last time…

- Ideas from cognitive science
  - The classical view and why it fails
  - Two family resemblance views: prototypes and exemplars
  - Hints about richer structure?
- Ideas from statistical machine learning
  - Supervised, unsupervised and semi-supervised learning
  - A simple Gaussian classifier (linked to prototype models)
- Today…
  - An extension of the Gaussian classifier
  - More classifiers (linked to exemplar models)

The prototype view

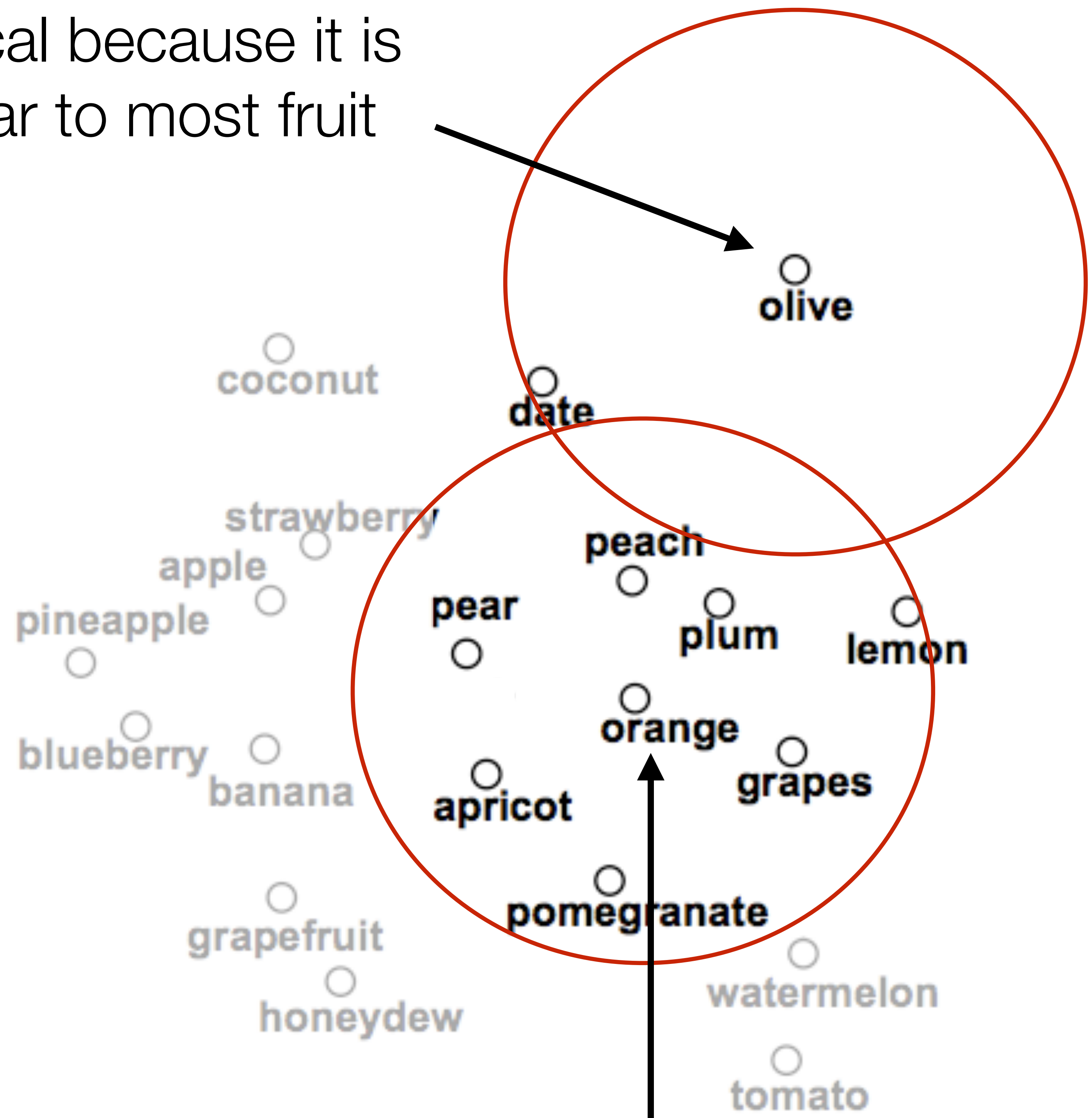
Olive is atypical because it is not similar to the prototype



The prototype

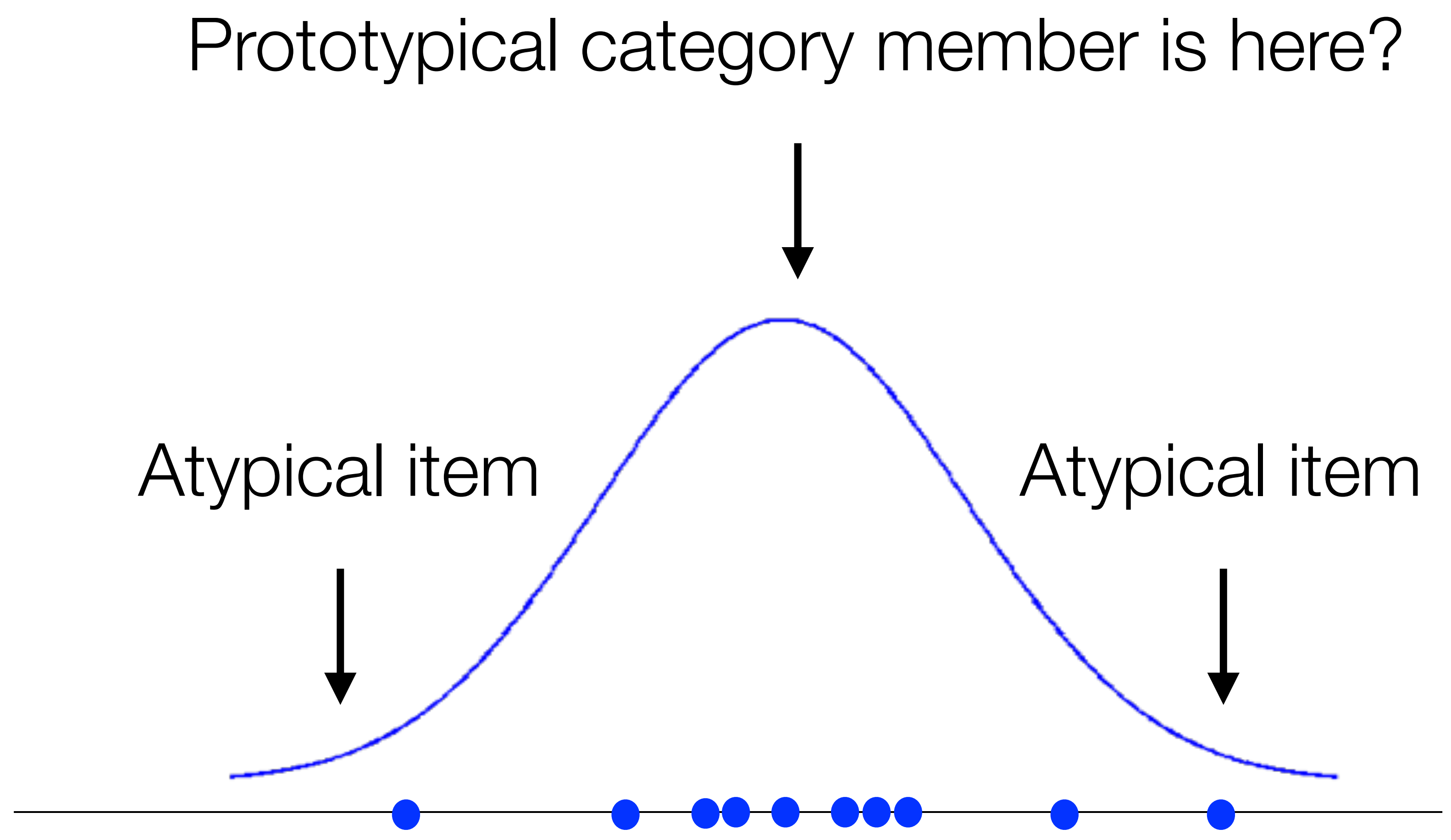
Orange is typical because it is similar to the prototype

Olive is atypical because it is very dissimilar to most fruit



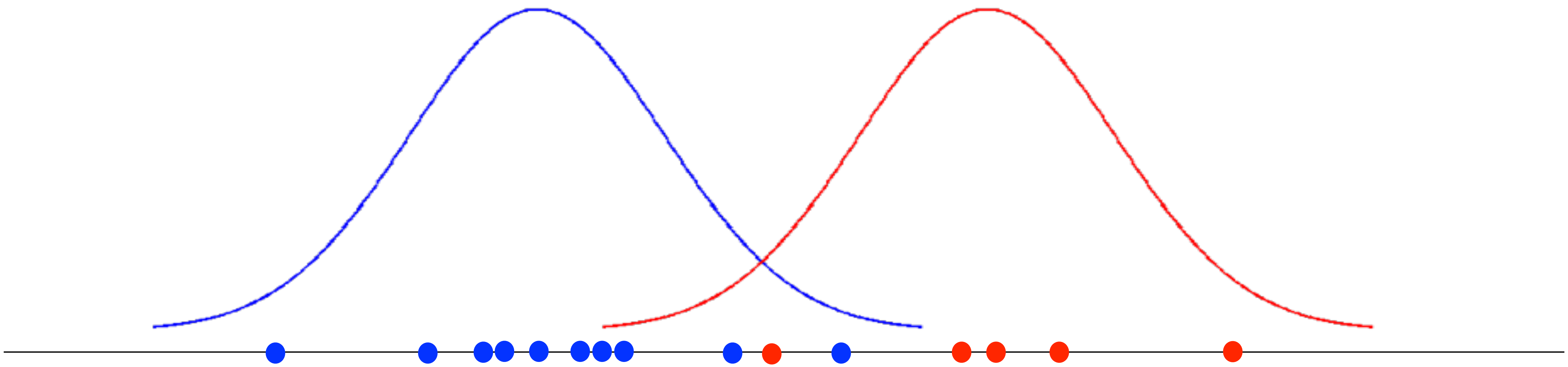
The exemplar view

Orange is typical because it is similar to lots of other fruit



This kind of classifier is very closely linked with **prototype** theory in psychology

The classifier from last time



Inference based on five parameters:

μ_a : Mean of category a

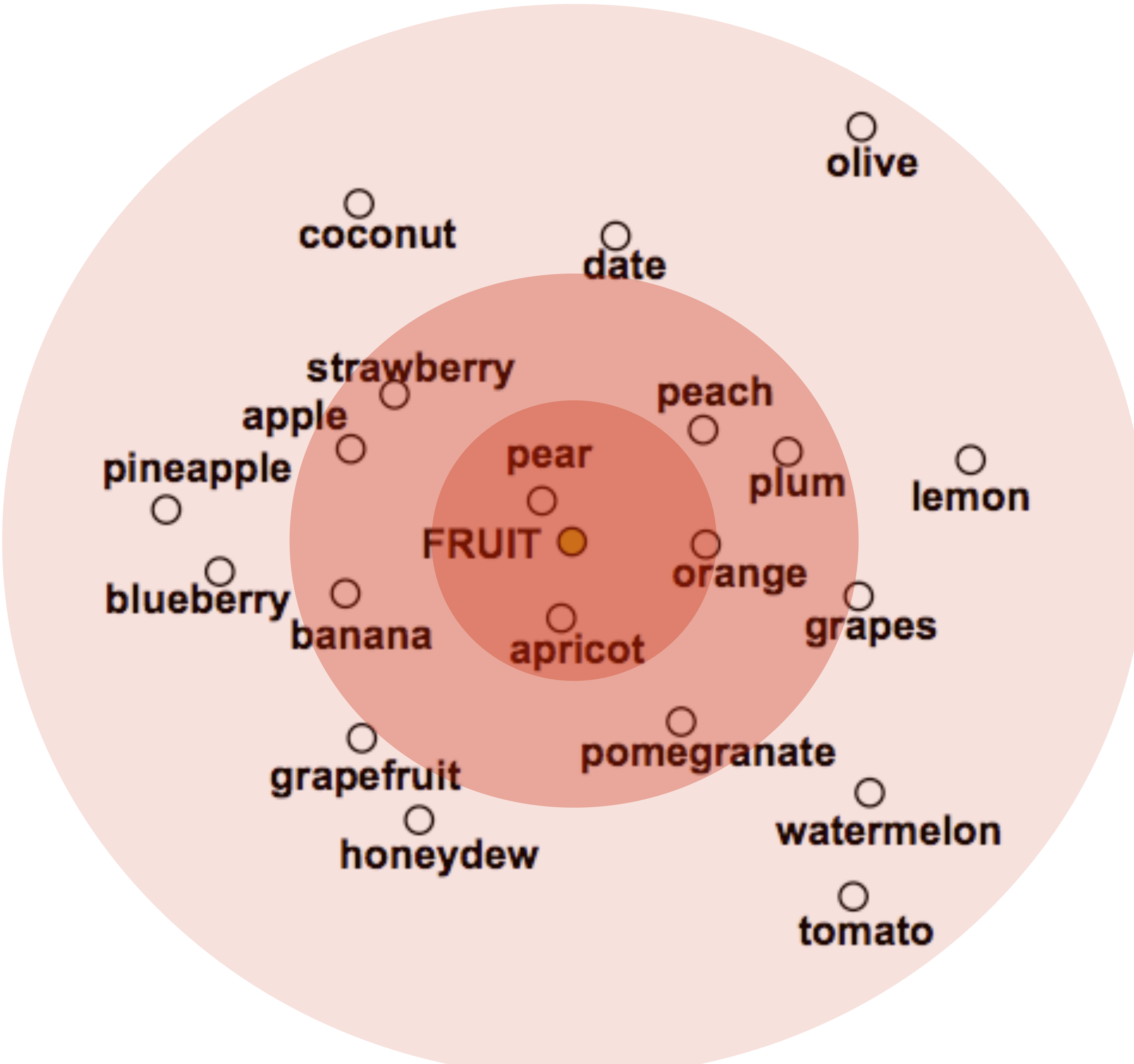
μ_b : Mean of category b

σ_a : Standard deviation of category a

σ_b : Standard deviation of category b

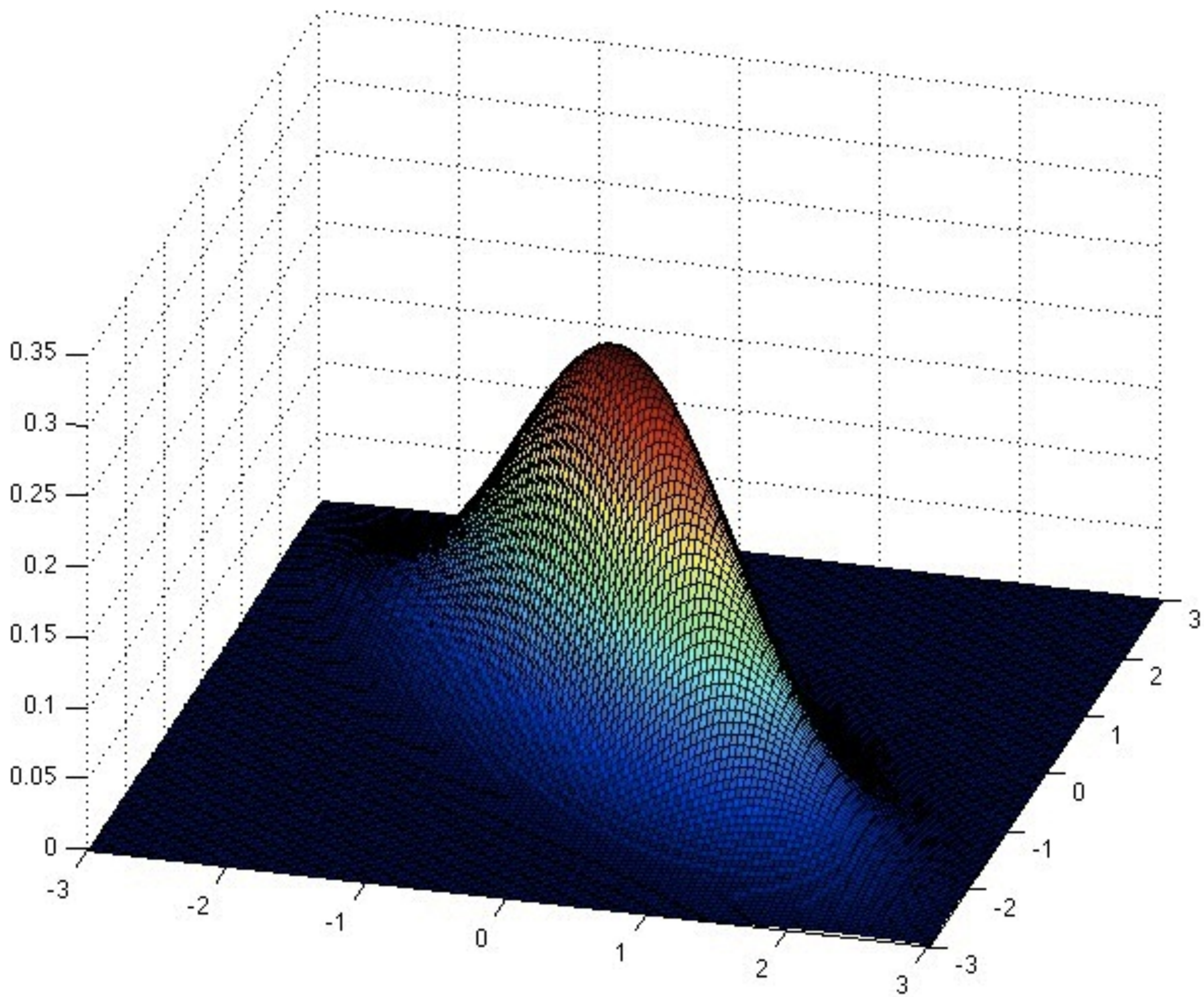
θ : Base rate for category a

Extension to the multidimensional case



Each category is a probability distribution defined over a multidimensional space

Let's say… each category is a multivariate Gaussian



The multivariate Gaussian

Vector describing the observation

$$P(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{k/2}\sqrt{\det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

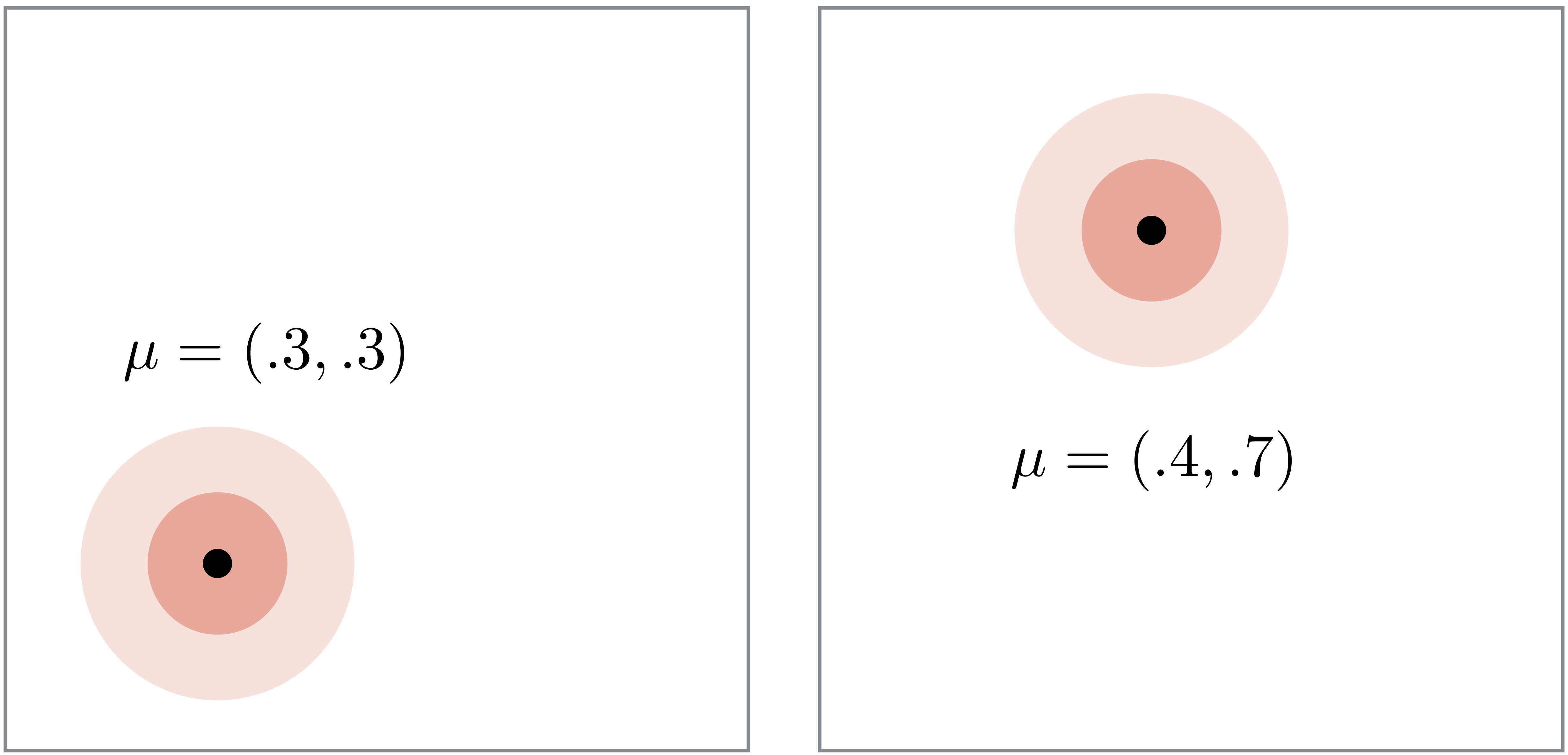
Vector describing the mean

Covariance matrix

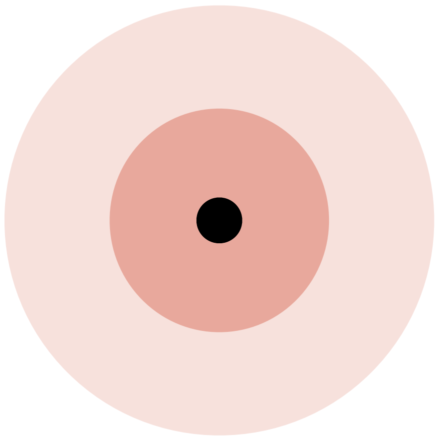
Matrix determinant

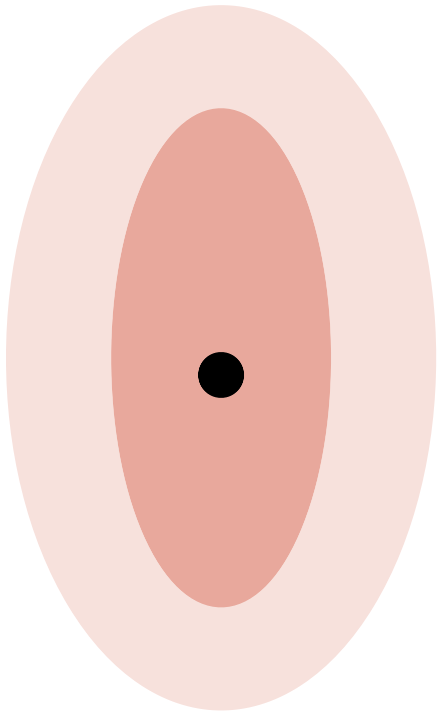
Matrix inverse

The mean vector describes where the distribution is centred



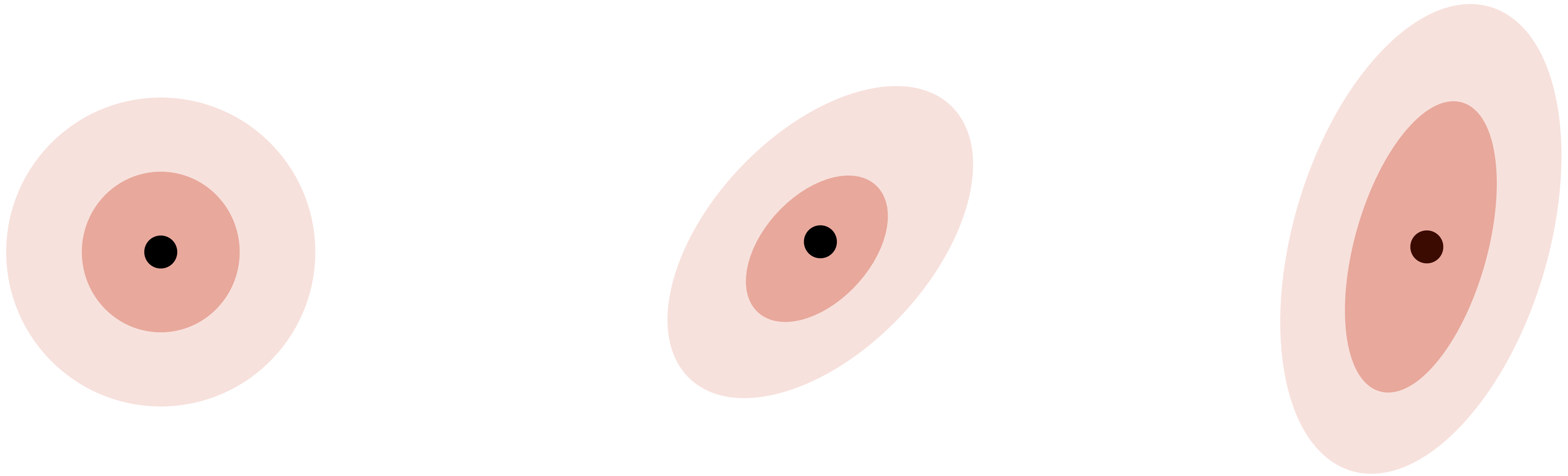
The main diagonal of the covariance matrix describes elongation





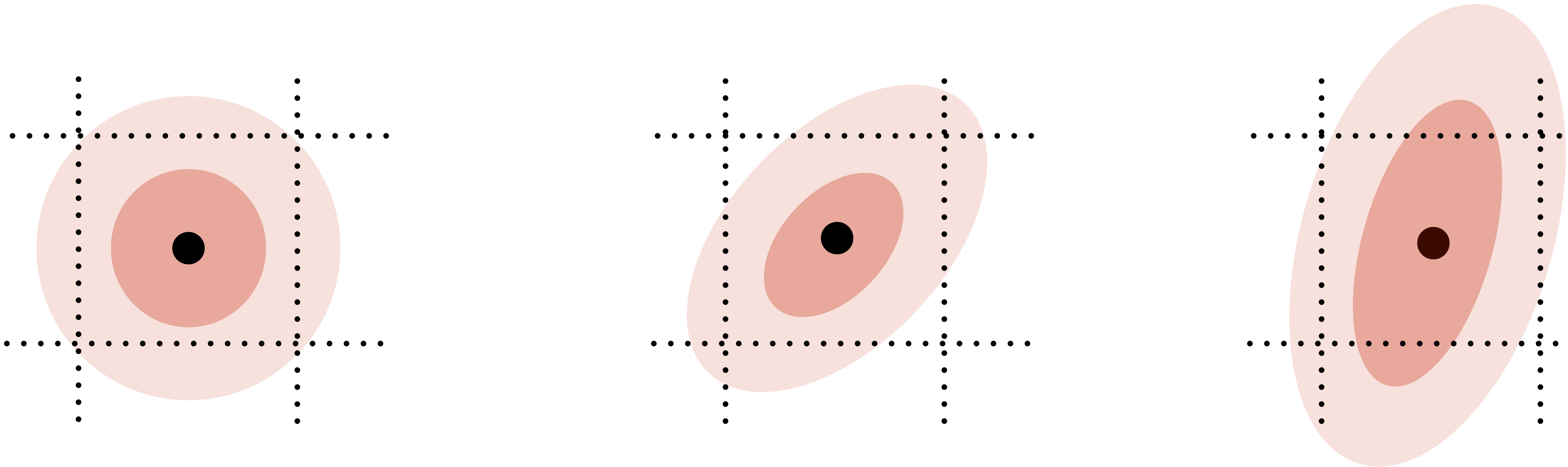
$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

The off diagonal elements describe orientation



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & .5 \\ .5 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & .5 \\ .5 & 2 \end{pmatrix}$$

The off diagonal elements describe orientation



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & .5 \\ .5 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & .5 \\ .5 & 2 \end{pmatrix}$$

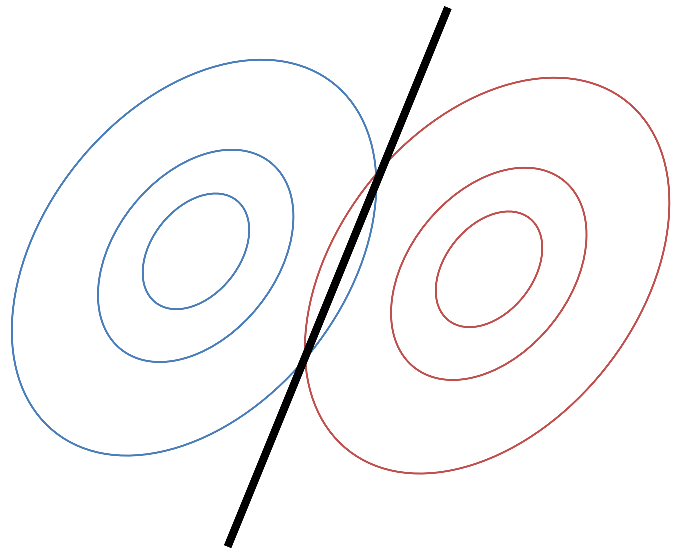
Doing it in R

- We won't spend time working on this at a low level
- See: multivariateGaussian function in classifiers.R
- See: the mvtnorm package
  - rmvnorm - generates samples from a multivariate normal
  - dmvnorm - calculates probability under a multivariate

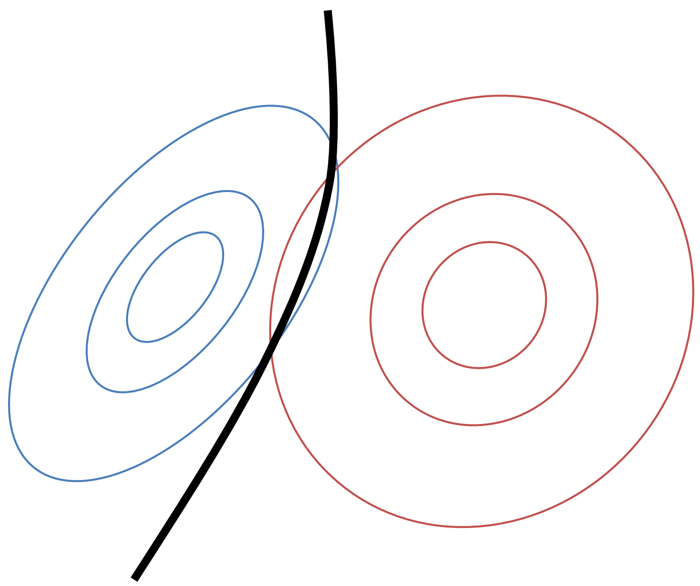
```
install.packages( "mvtnorm" )
library( mvtnorm)
rmvnorm( n=3, mean=c(10,0), sigma=rbind( c(10, 3), c(3, 1) ) )

          [,1]       [,2]
[1,] 11.279109  0.6773104
[2,]  7.777925 -1.4224349
[3,]  5.109438 -1.7661479
```

Linear and quadratic decision bounds

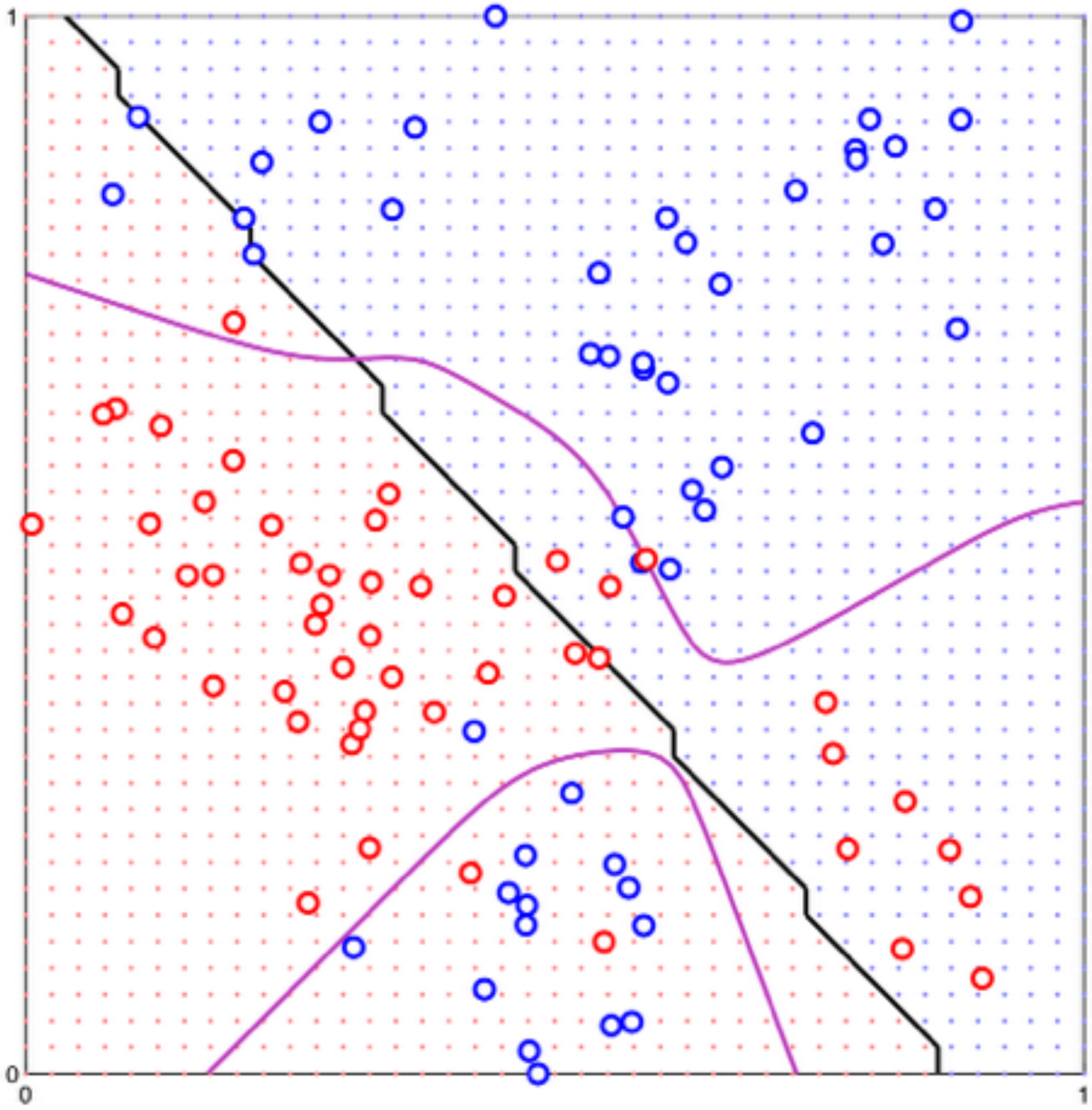


If the covariance matrices are the same for the two categories, the decision rule is a linear function in the space

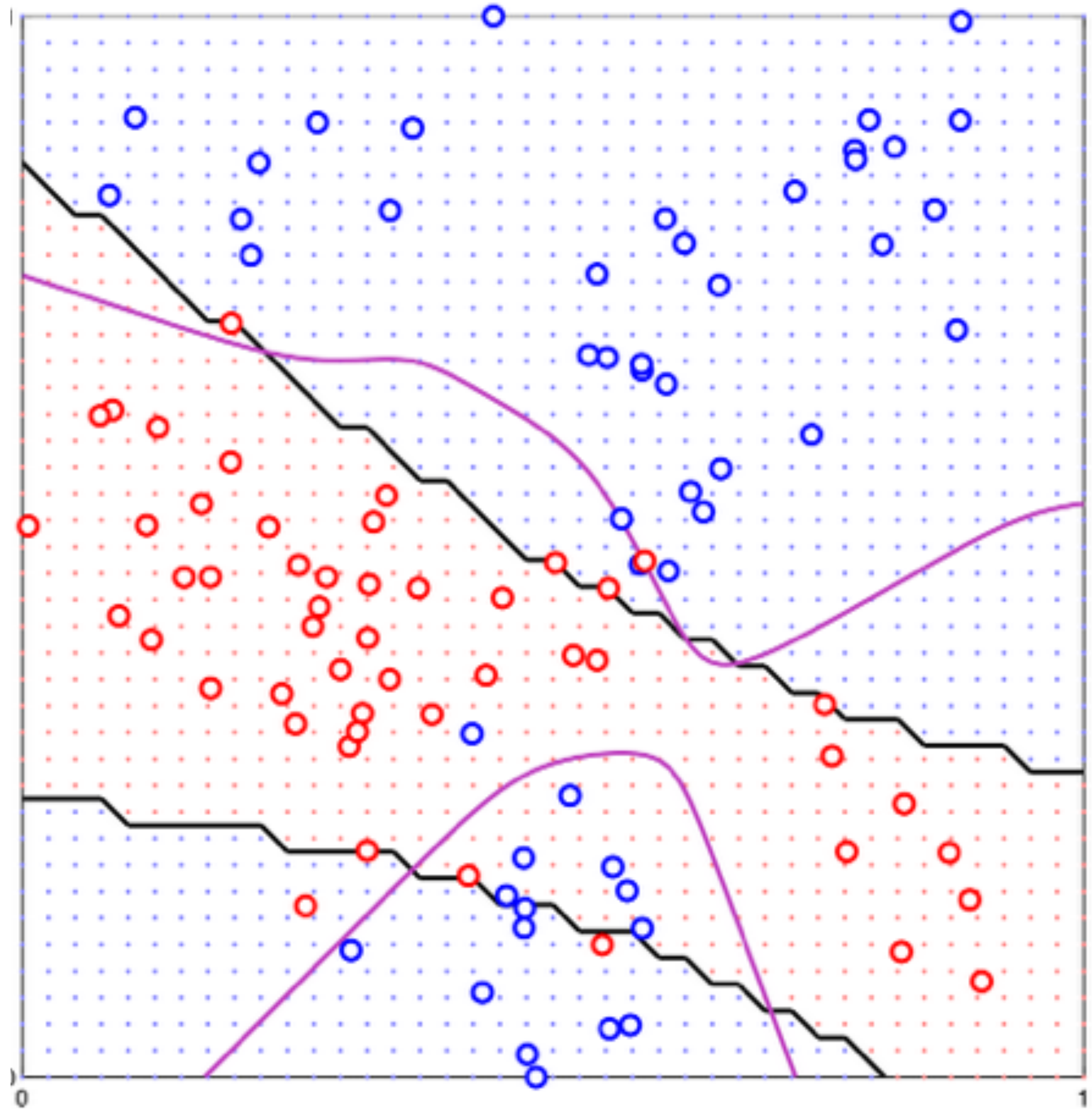


Unequal covariance matrices produce decision boundaries described by quadratic functions

Comparison



Linear



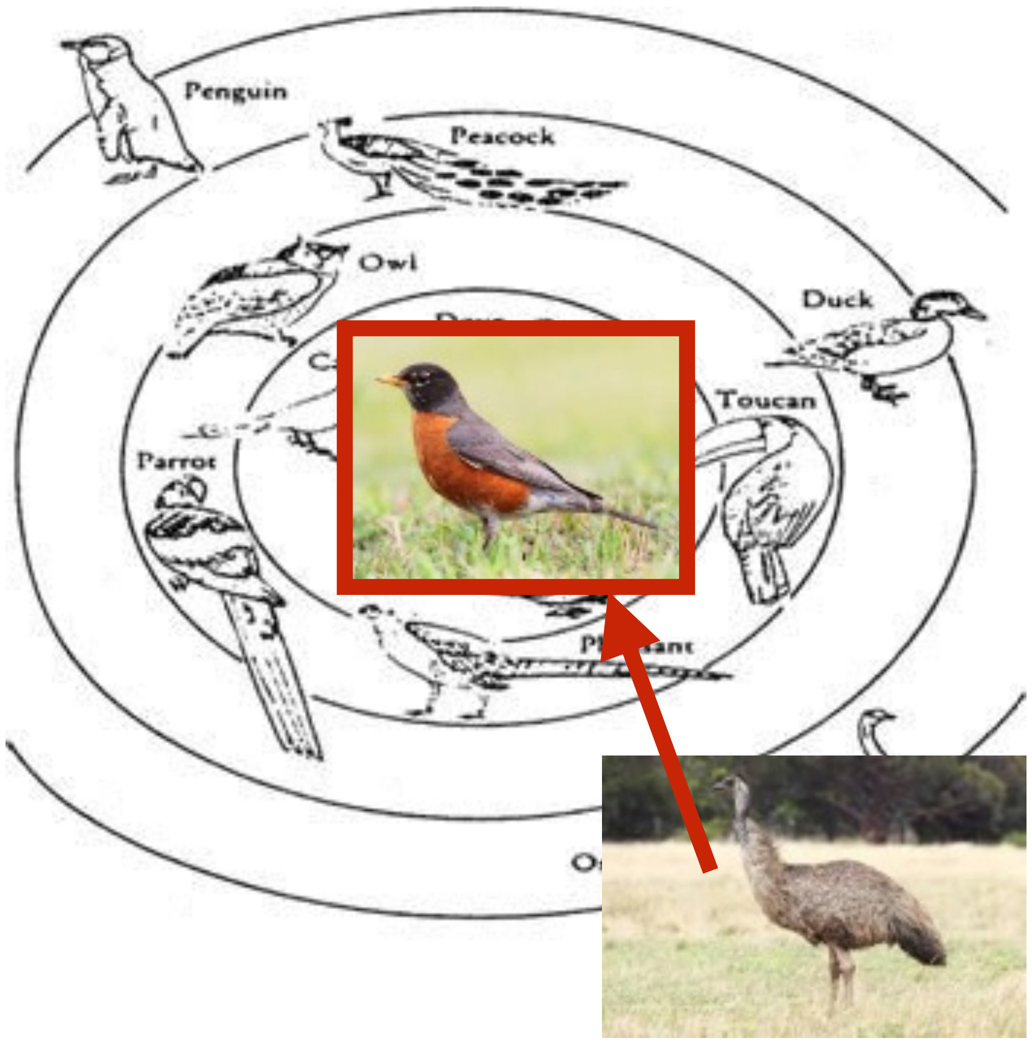
Quadratic

Demonstration code

(classifiers.R, multivariateGaussianClassifier function)

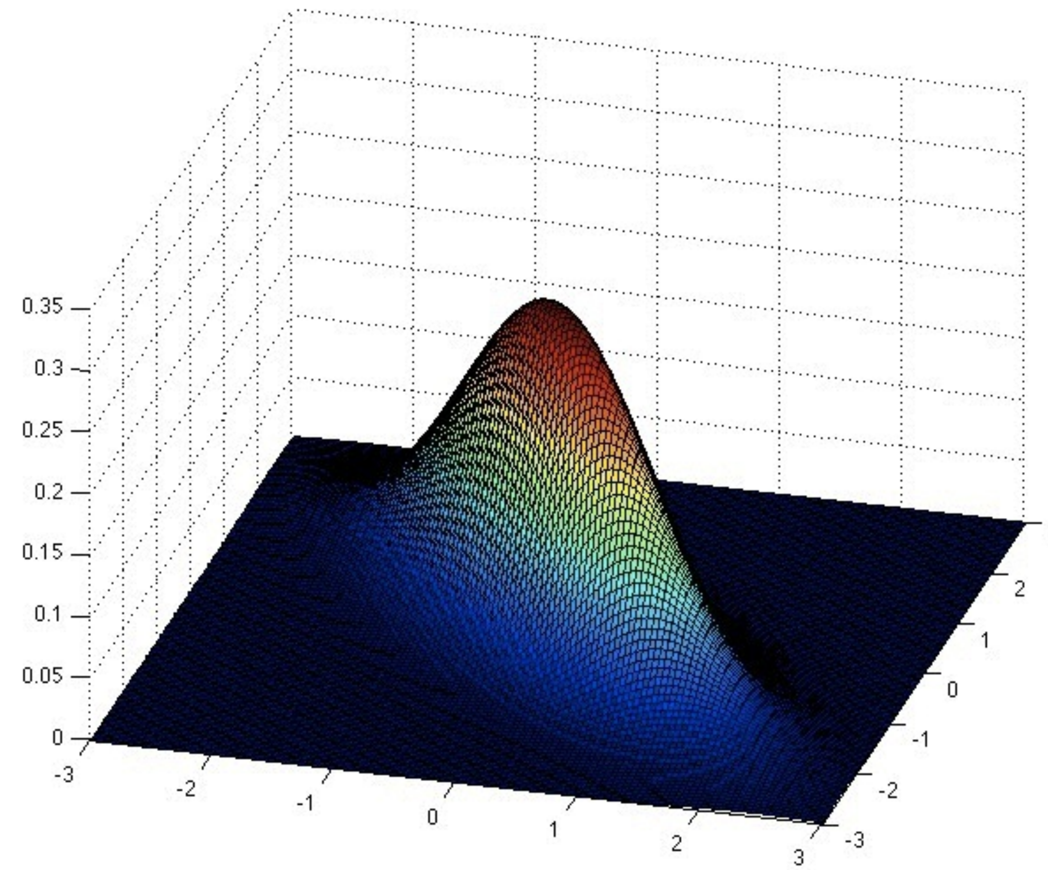
Where are we up to?

Prototype theory



We store a single “prototype”, against which new exemplars are measured

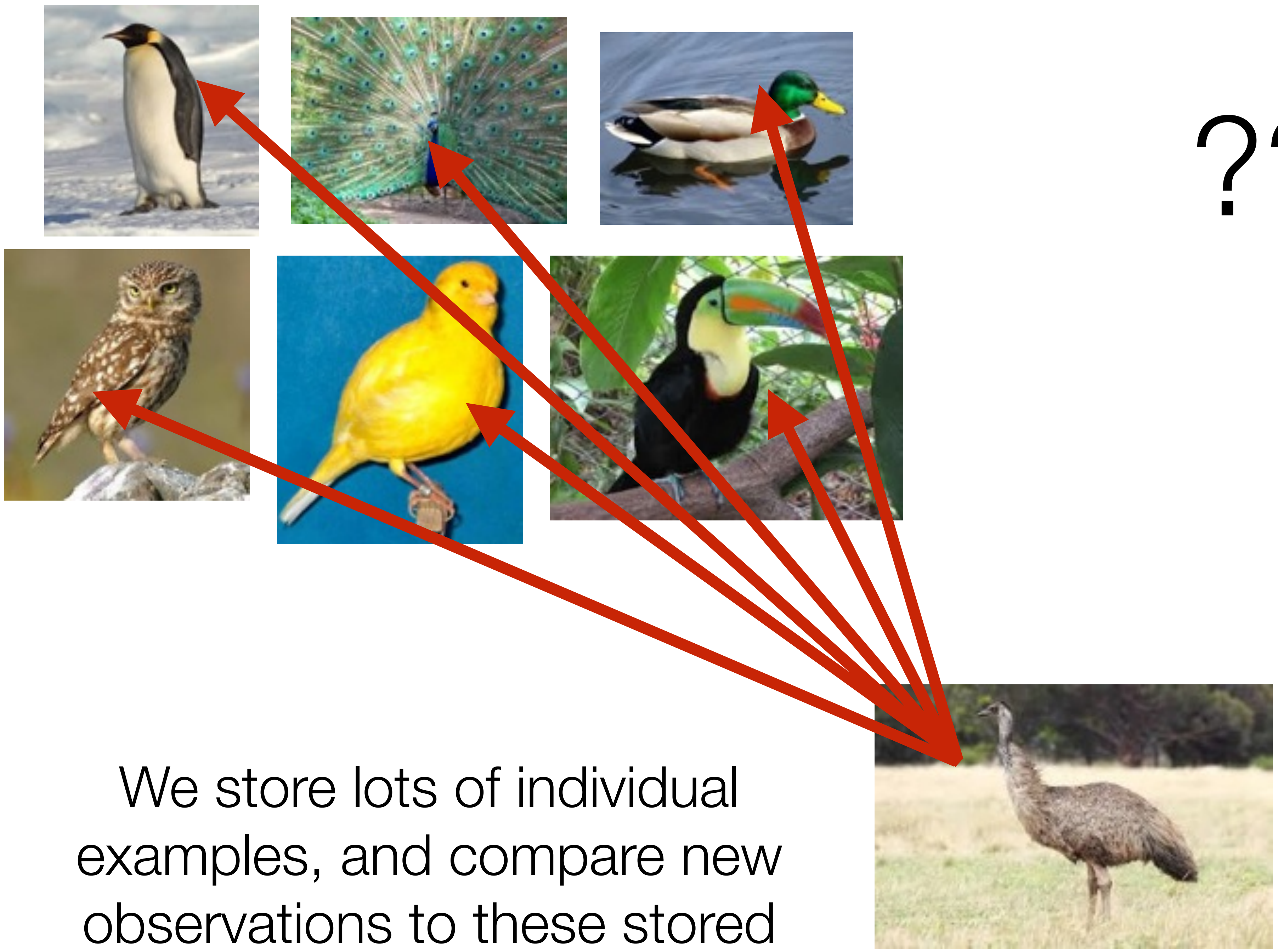
Corresponding classifier…



Categories are represented as probability distributions with a single peak

Prototype theory

Corresponding classifier…



????

We store lots of individual examples, and compare new observations to these stored items separately

Different kinds of classifiers

- Model based, “parametric” classifiers
  - Assume we know the shape of the category distribution (e.g., normal)
  - Learn the parameters (mean, covariance) that describe a category
  - Hold up well even when there’s very little data
  - Perform poorly when the category has a different shape

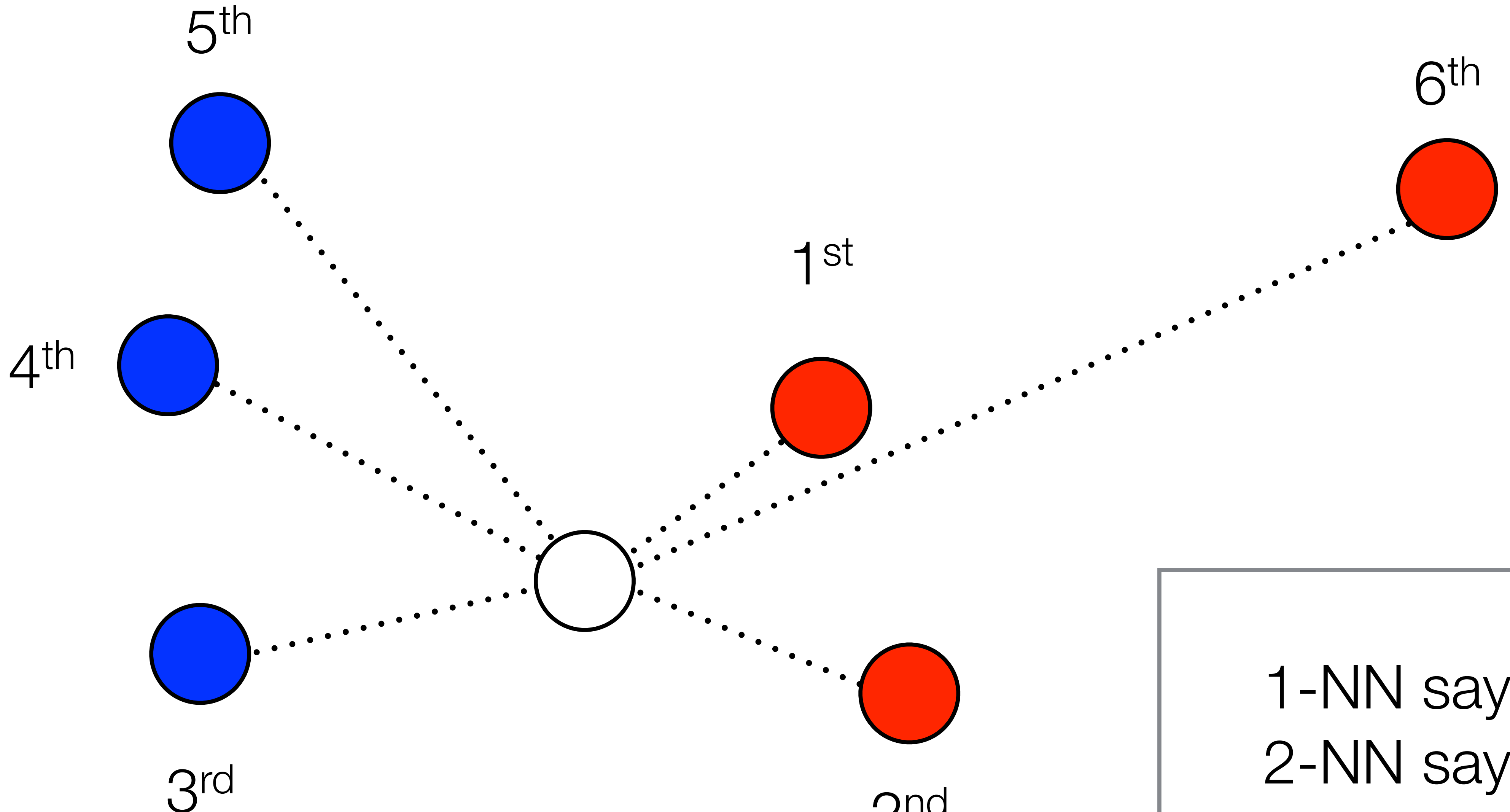
Different kinds of classifiers

- Model based, “parametric” classifiers
  - Assume we know the shape of the category distribution (e.g., normal)
  - Learn the parameters (mean, covariance) that describe a category
  - Hold up well even when there’s very little data
  - Perform poorly when the category has a different shape
- Model free, “non-parametric” classifiers
  - Avoid making any specific assumption about the category distribution
  - Try to let the data itself tell you the shape of the category
  - Very flexible, and perform well no matter what shape the category is
  - Tend to perform worse when you have very little data

The k-nearest neighbours (kNN) classifier

k-NN

- Very simple algorithm for finding label $l(y)$
  - Find $X^{(k\text{-}near)}$, the k observations that are closest to y
  - Look up the labels $l(X^{(k\text{-}near)})$ of those k items
  - Use a "majority vote" to predict $l(y)$.
- In cognitive science terms:
  - Stores all items, and uses retrieval from memory to do all the work
  - It's an exemplar model

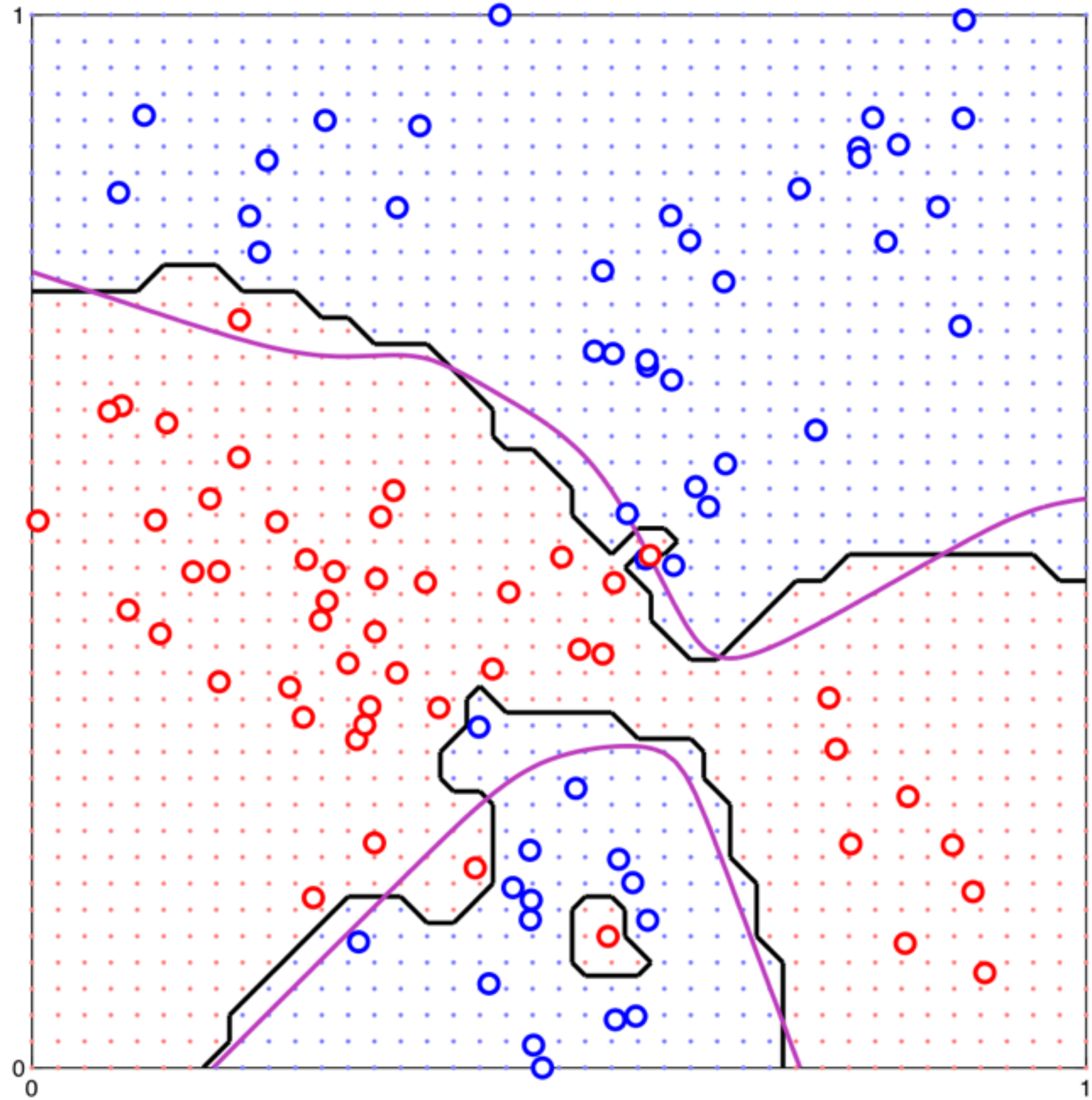


1-NN says: **RED**
2-NN says: **RED**
3-NN says: **RED**
4-NN says: either
5-NN says: **BLUE**
6-NN says: either

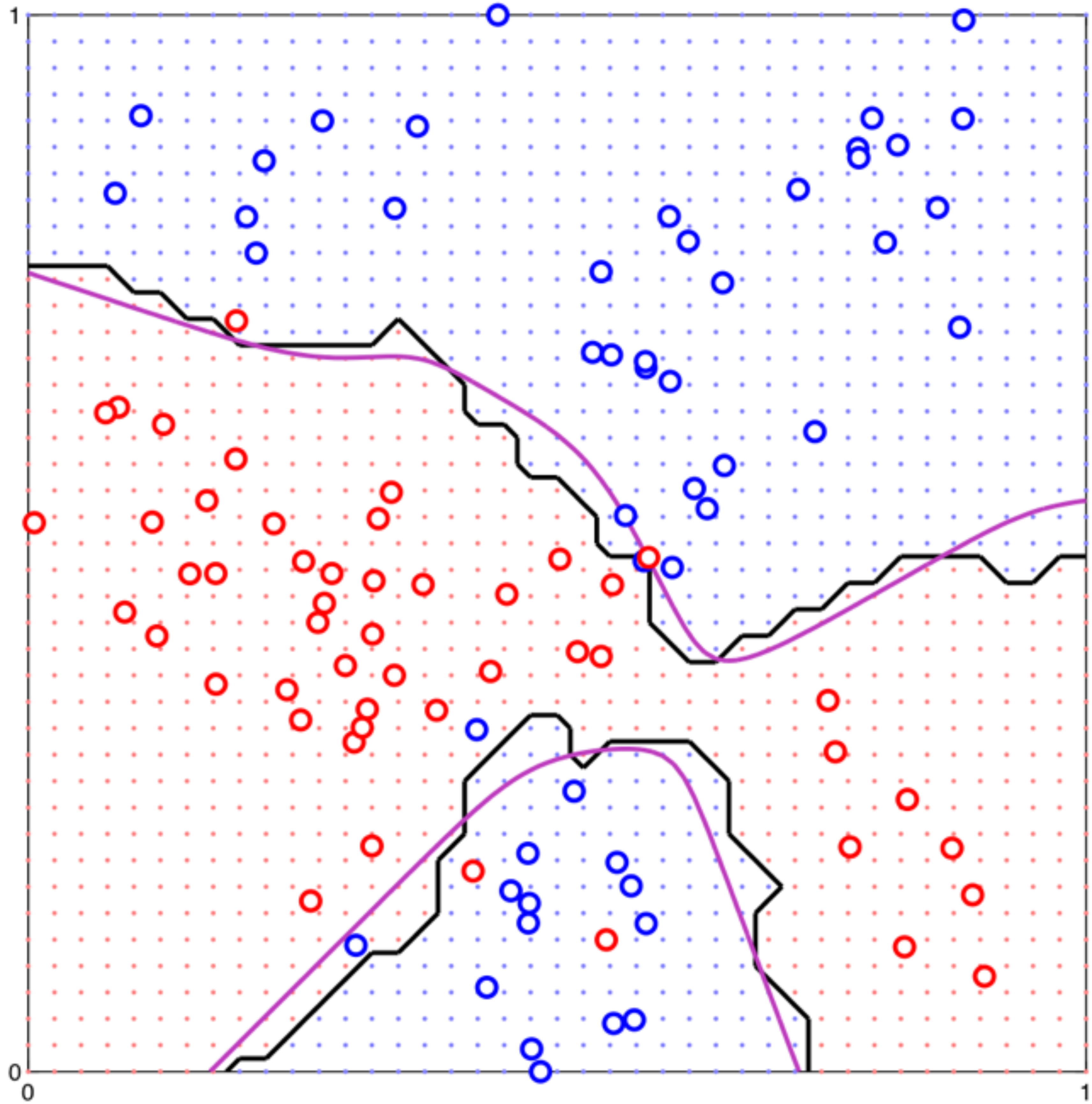
Demonstration code

(classifiers.R, kNN function)

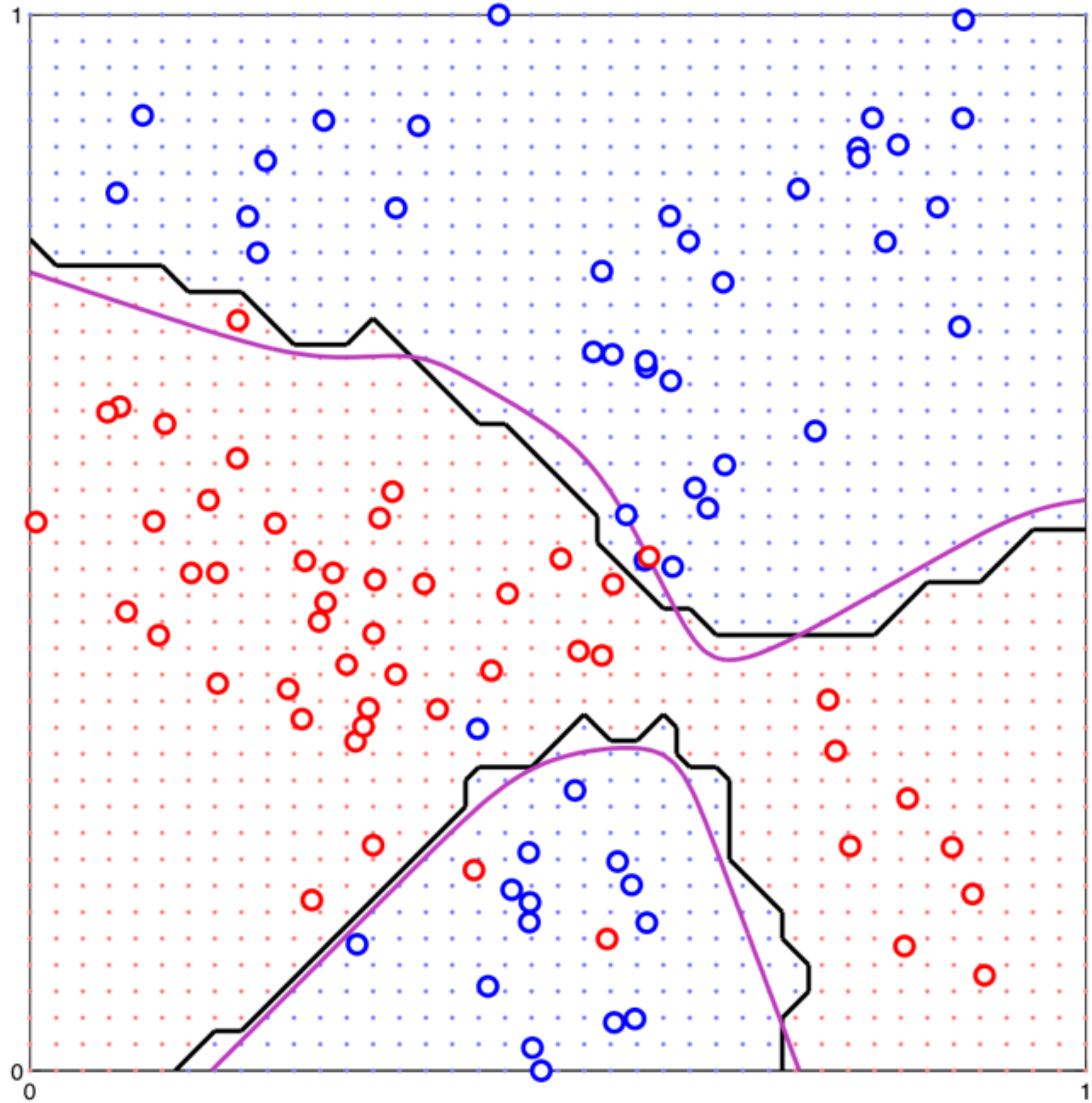
1 nearest neighbour



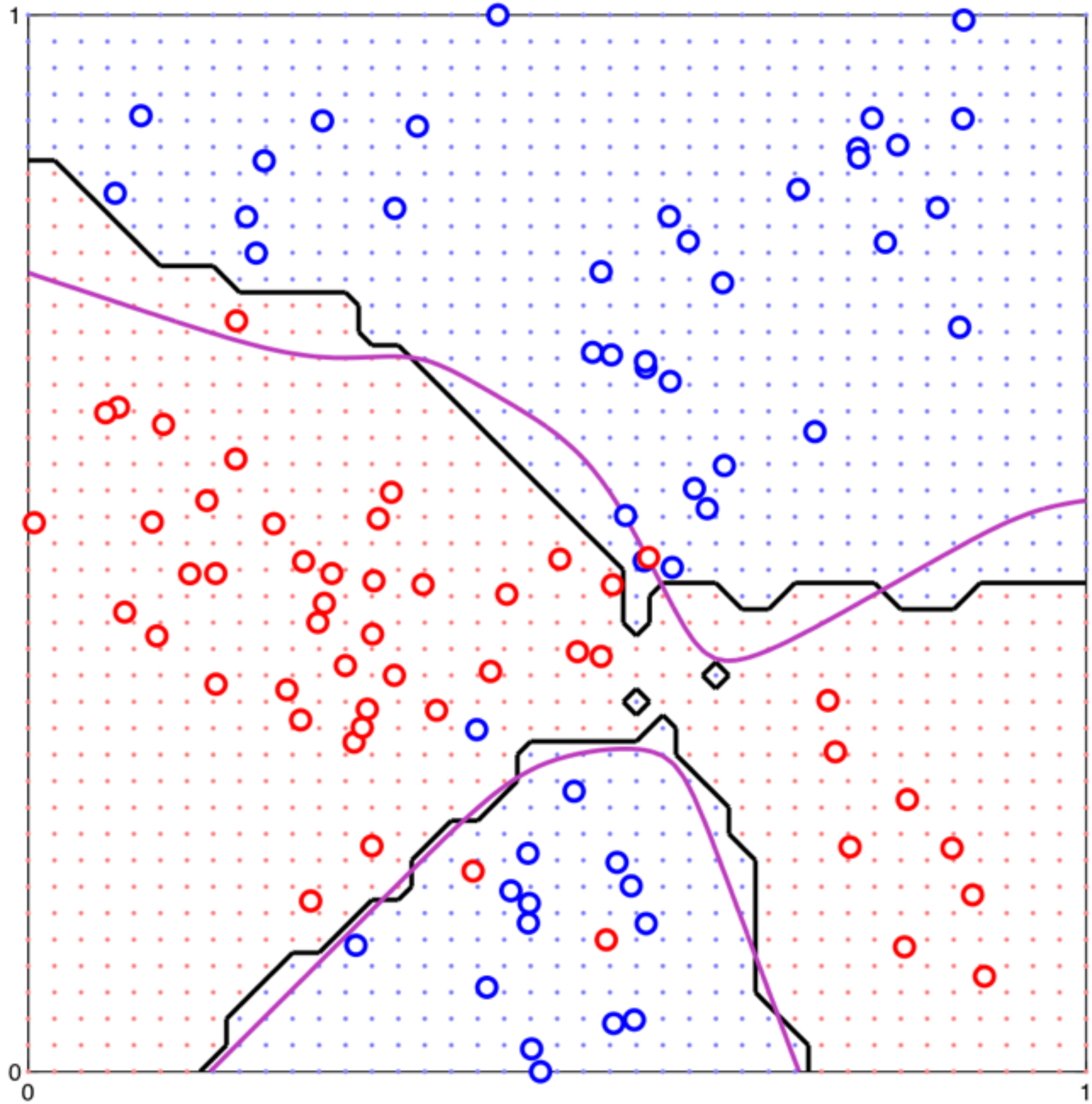
3 nearest neighbours



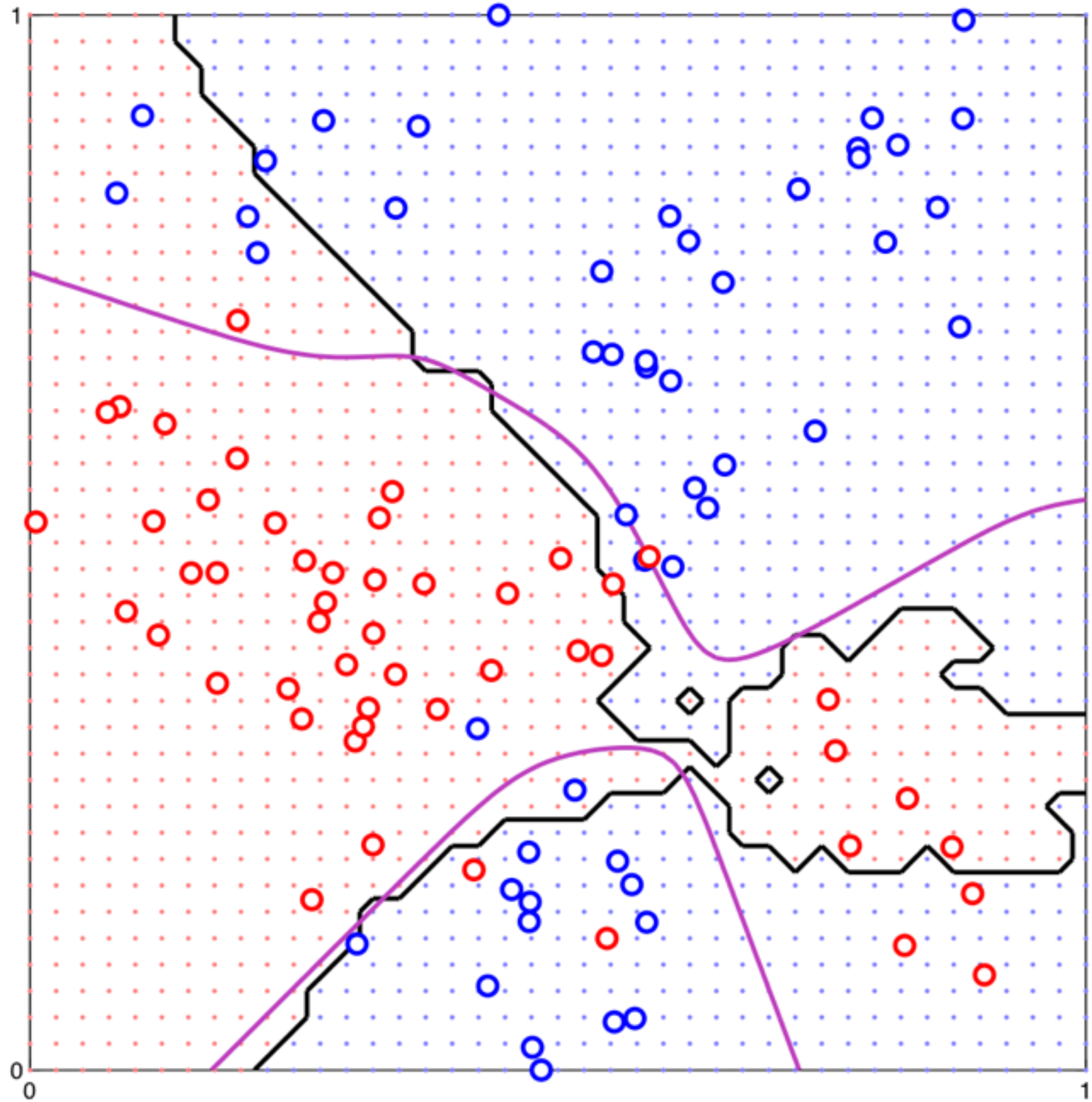
7 nearest neighbours



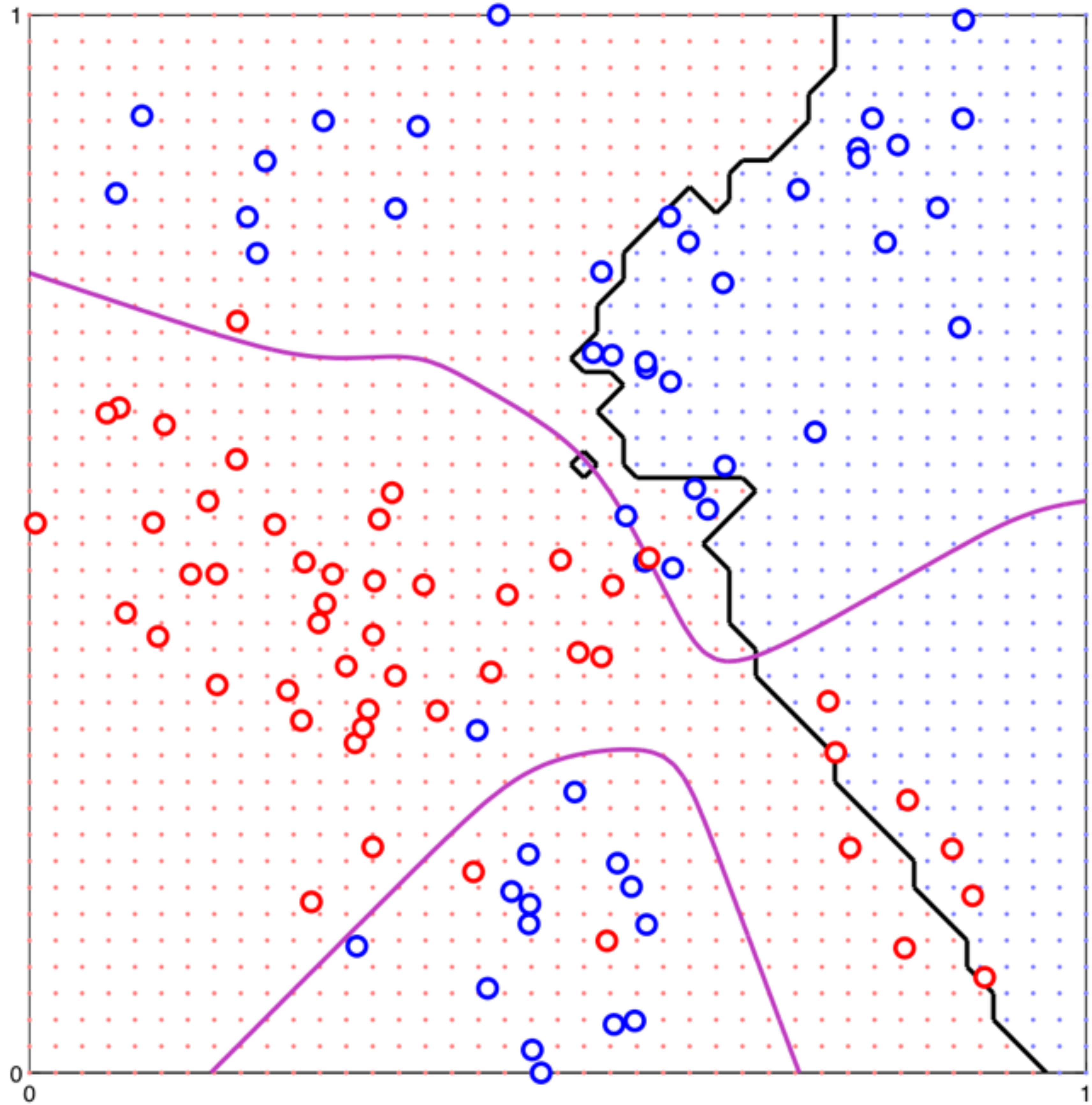
15 nearest neighbours



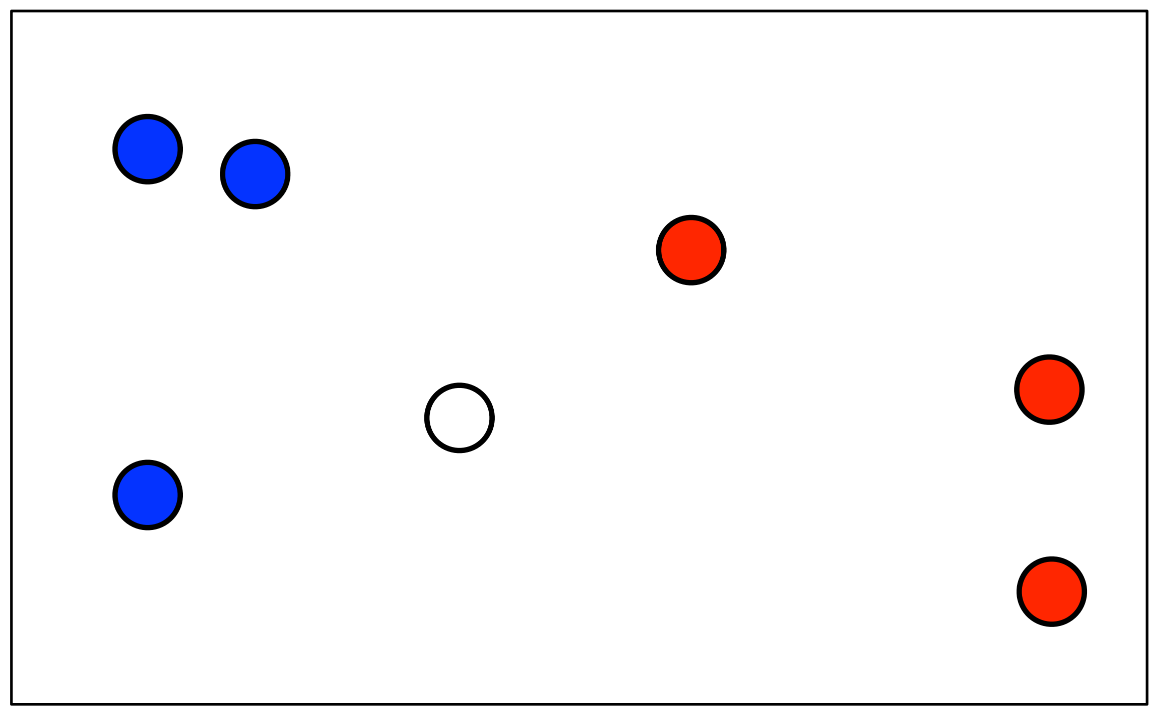
25 nearest neighbours



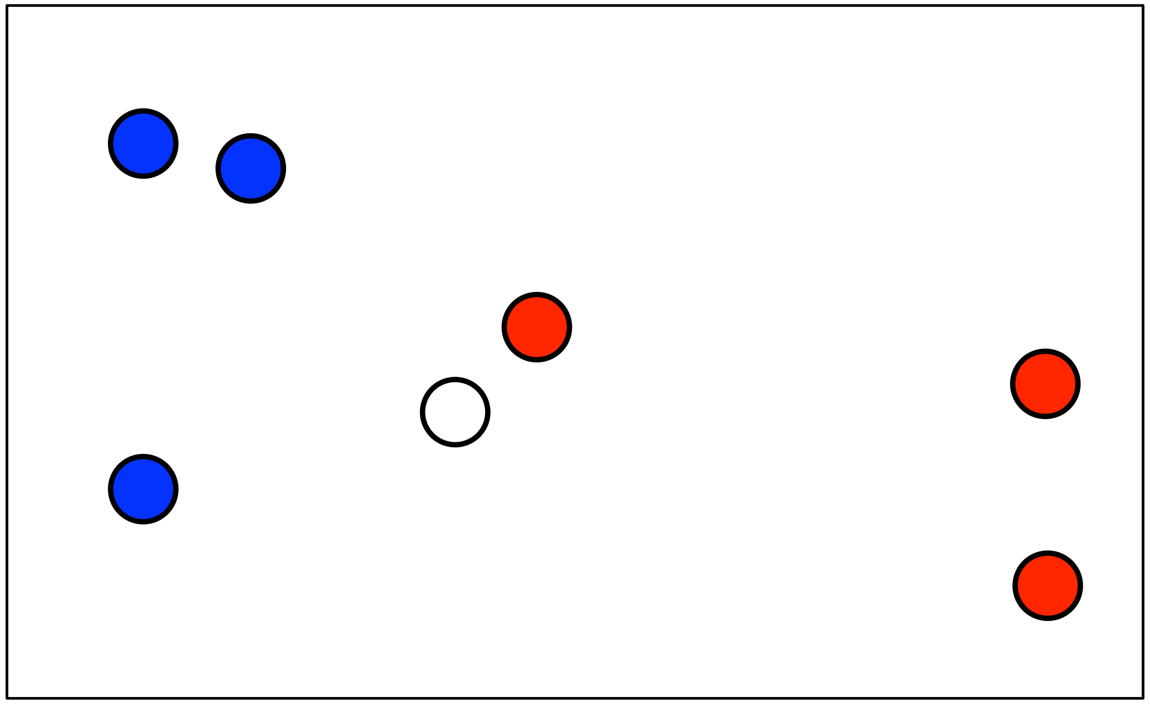
75 nearest neighbours



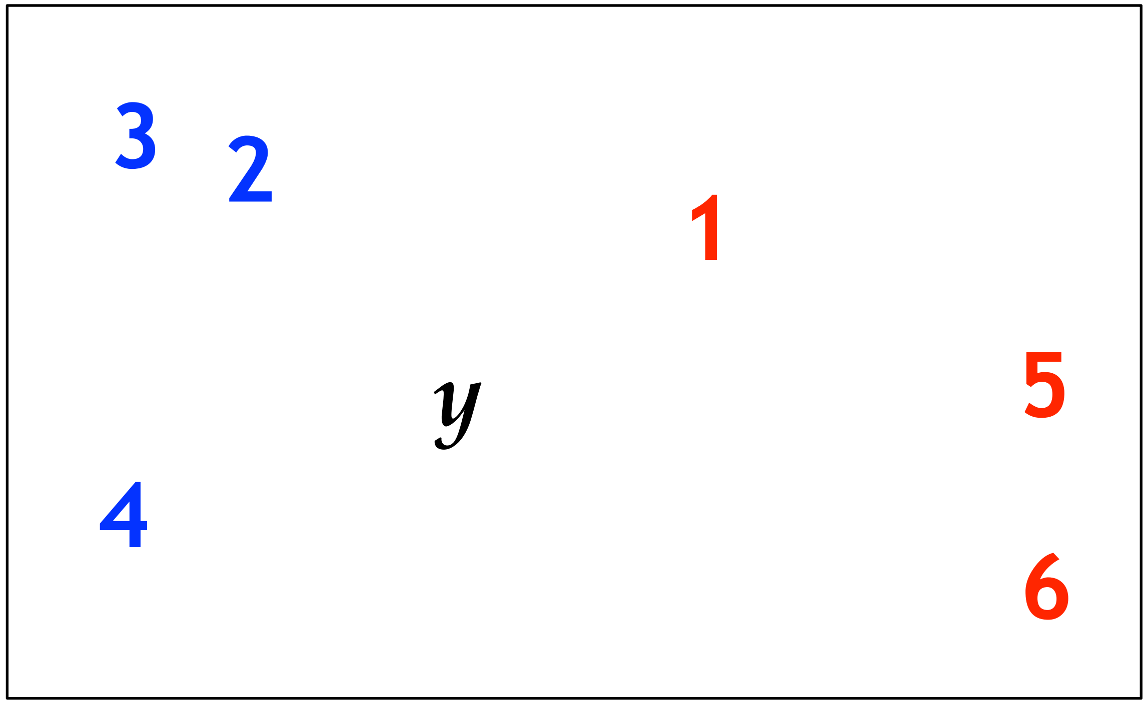
Is ○ a blue or red?

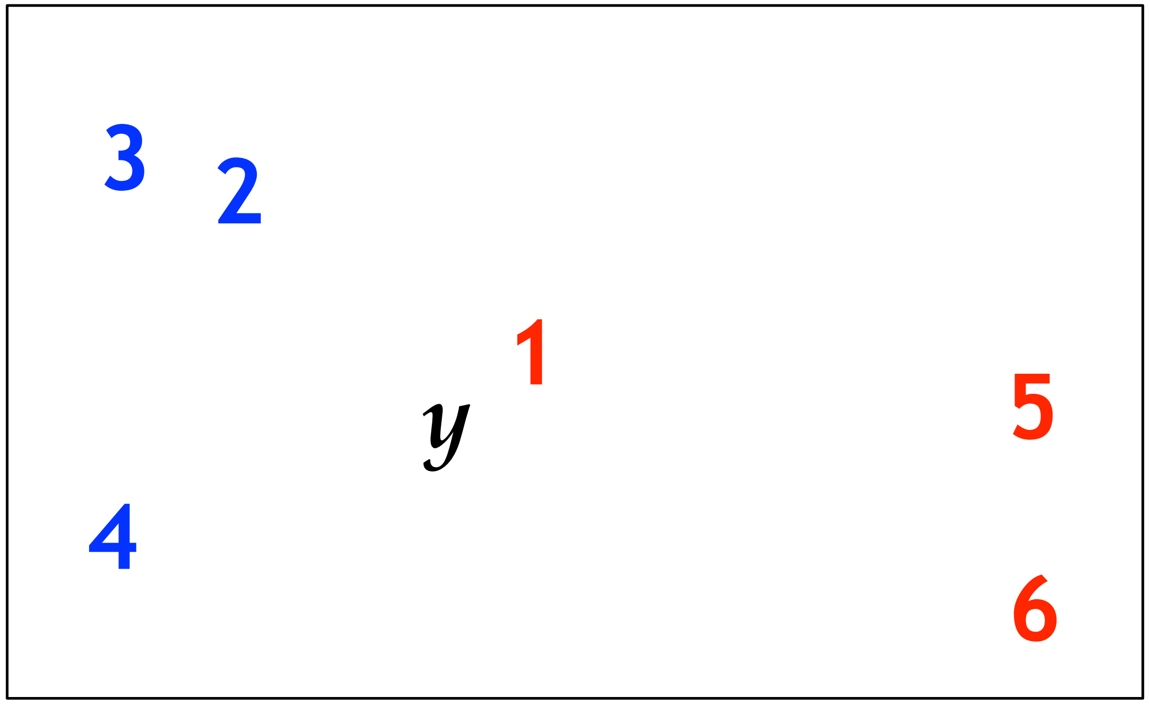


How about now?



The neighbour ranks are the same

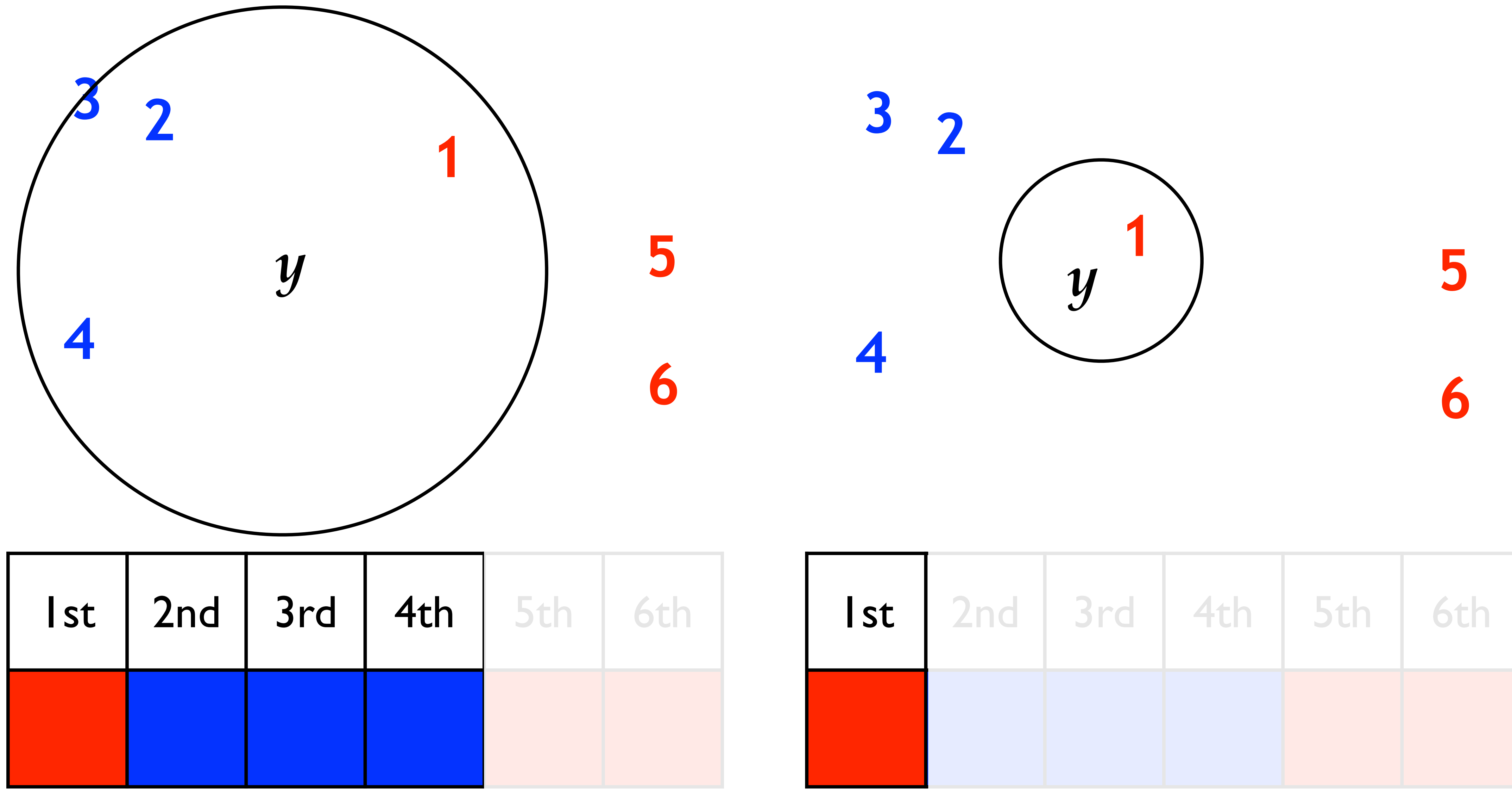




1st	2nd	3rd	4th	5th	6th

1st	2nd	3rd	4th	5th	6th

We don’t treat them the same



People pay attention to the distances

- It’s not just the “rank order”
  - A very close 1st NN is more convincing than a distant 1st NN
  - One very similar item can “swamp” everything else

- If we want a psychologically plausible exemplar model…
- … we’re going to have to make use of the actual distances

Non-parametric classifiers using kernel density estimators

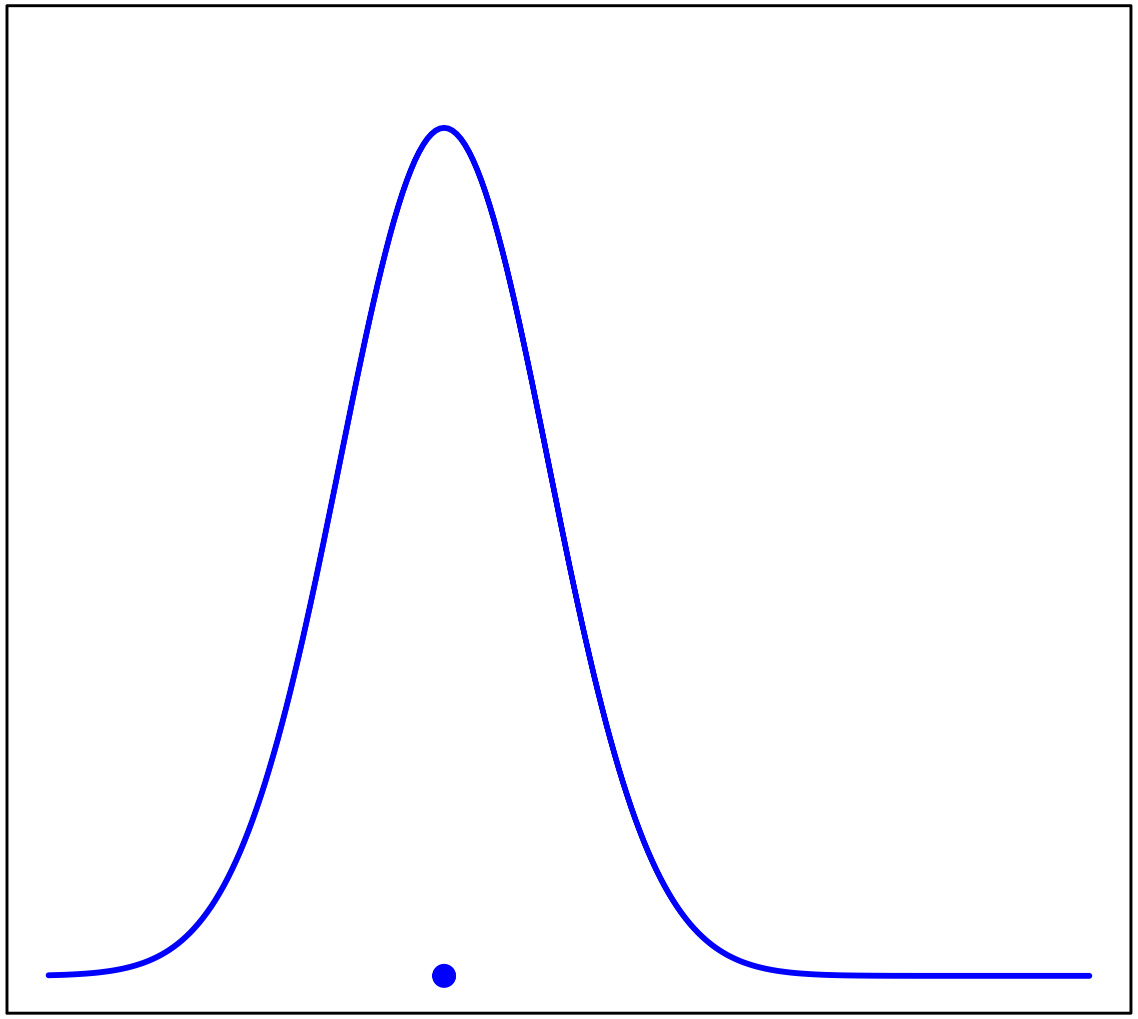
Kernel density estimators

- Flexible, model-based classifier:
  - Place a **kernel** *K* around every training exemplar
  - The kernel is a function (e.g., Gaussian distribution)

$$P(y|x_1, \ldots, x_n) \propto \sum_{i=1}^{n} K(y - x_i)$$

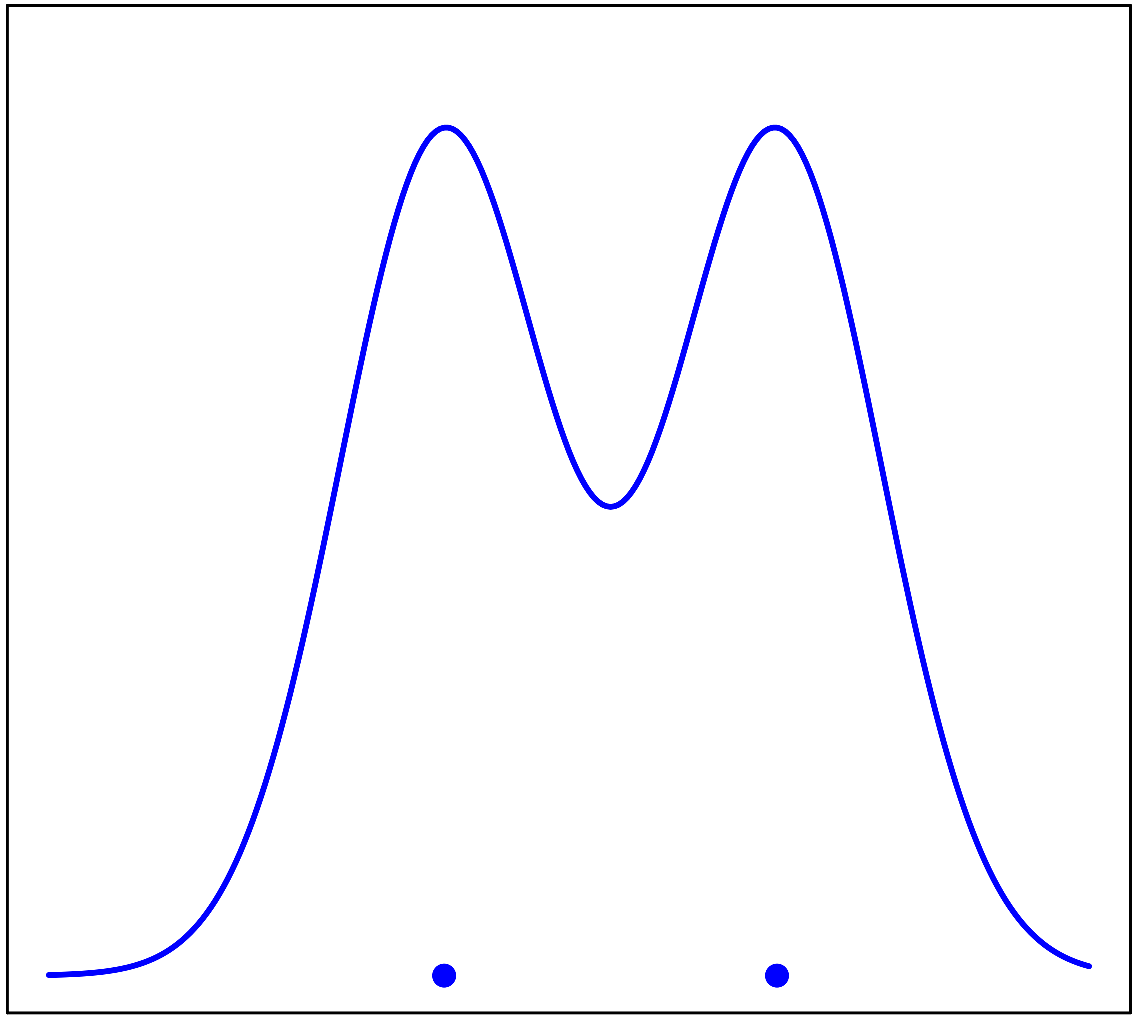
- In psychology:
  - The exponential kernel is special (Shepard's law of generalisation)
  - It produces a model known as the "generalised context model"
  - (Nosofsky, 1984, 1986)

Kernel density estimators



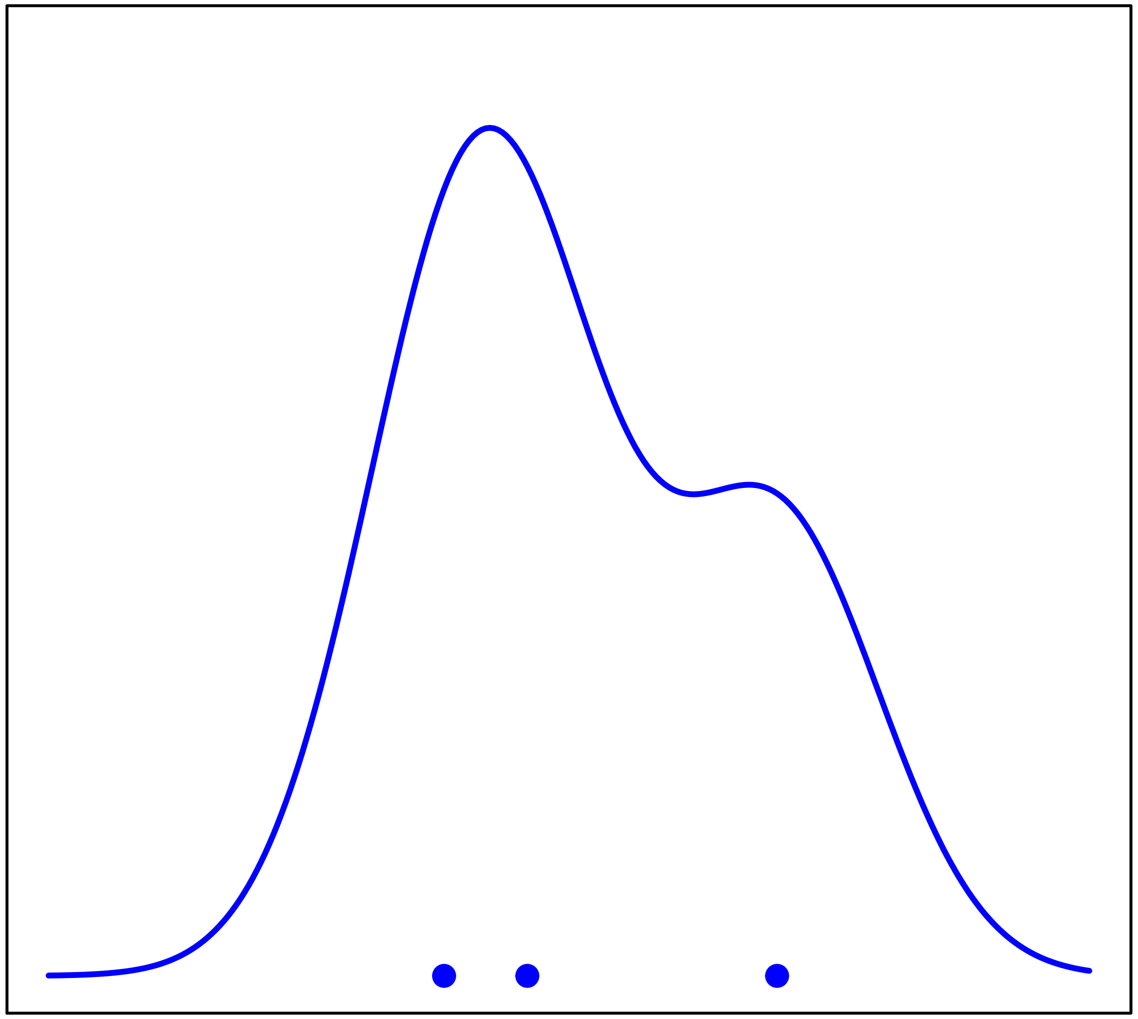
Gaussian kernel around a single observation

Kernel density estimators



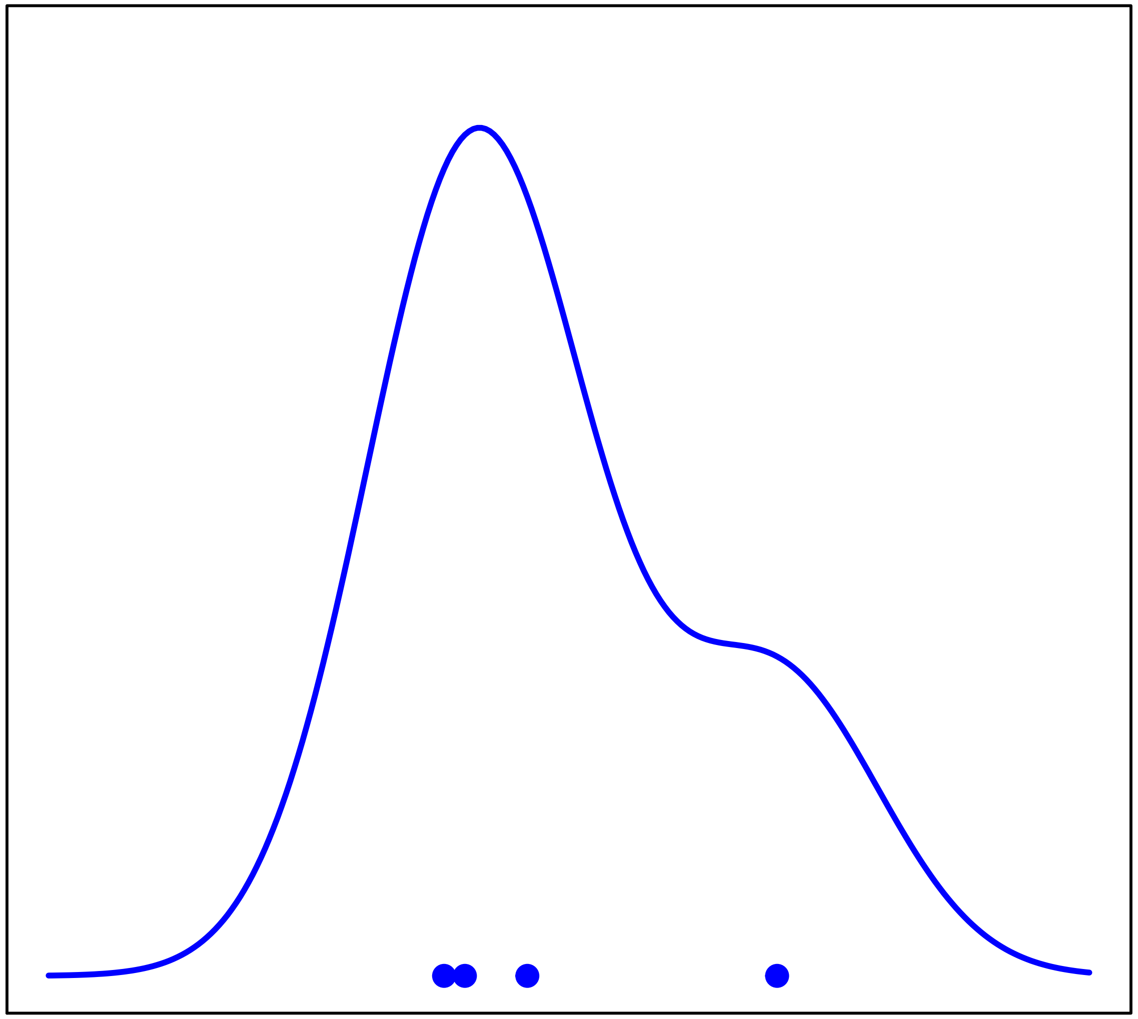
Gaussian kernels around
two observations

Kernel density estimators



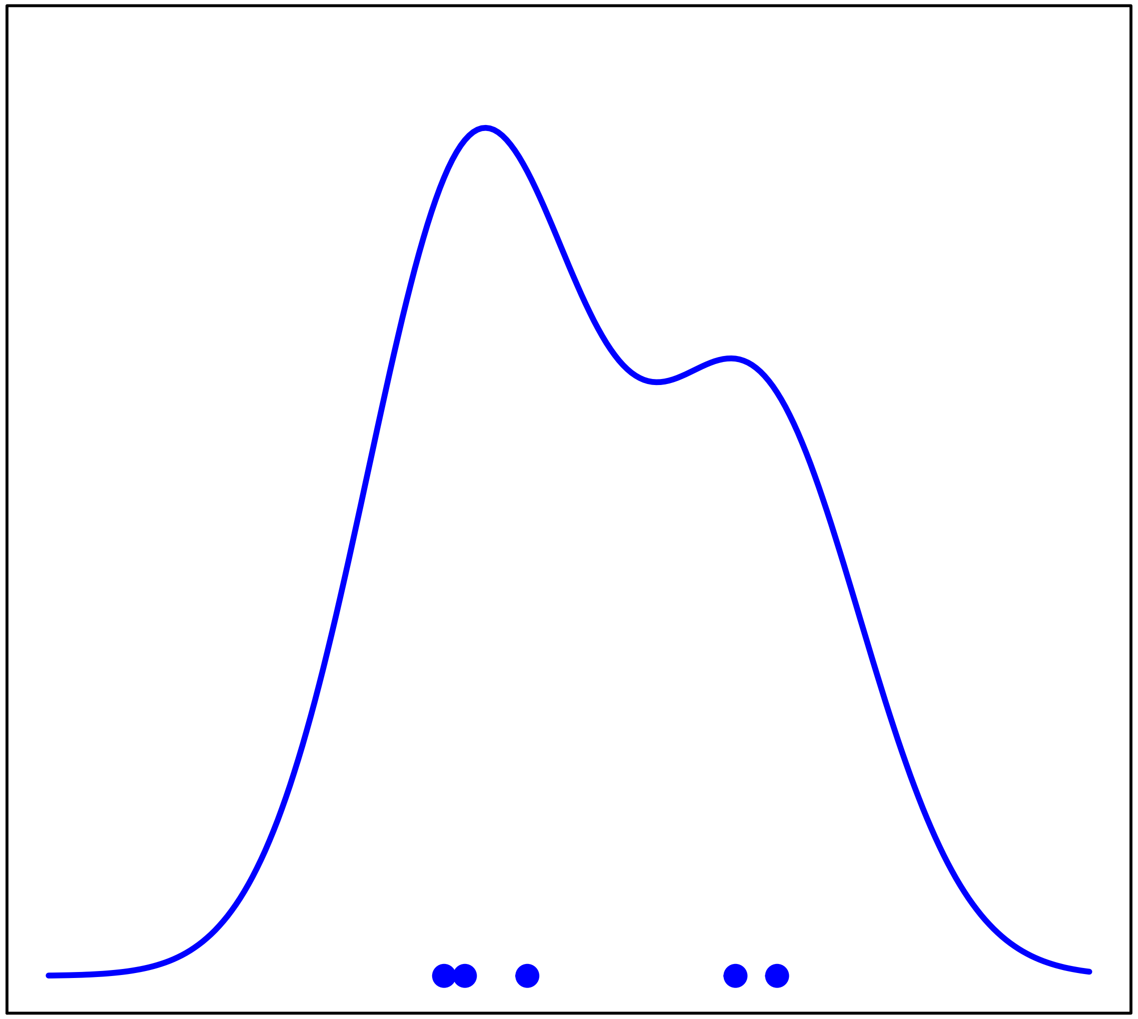
Gaussian kernels around three observations

Kernel density estimators



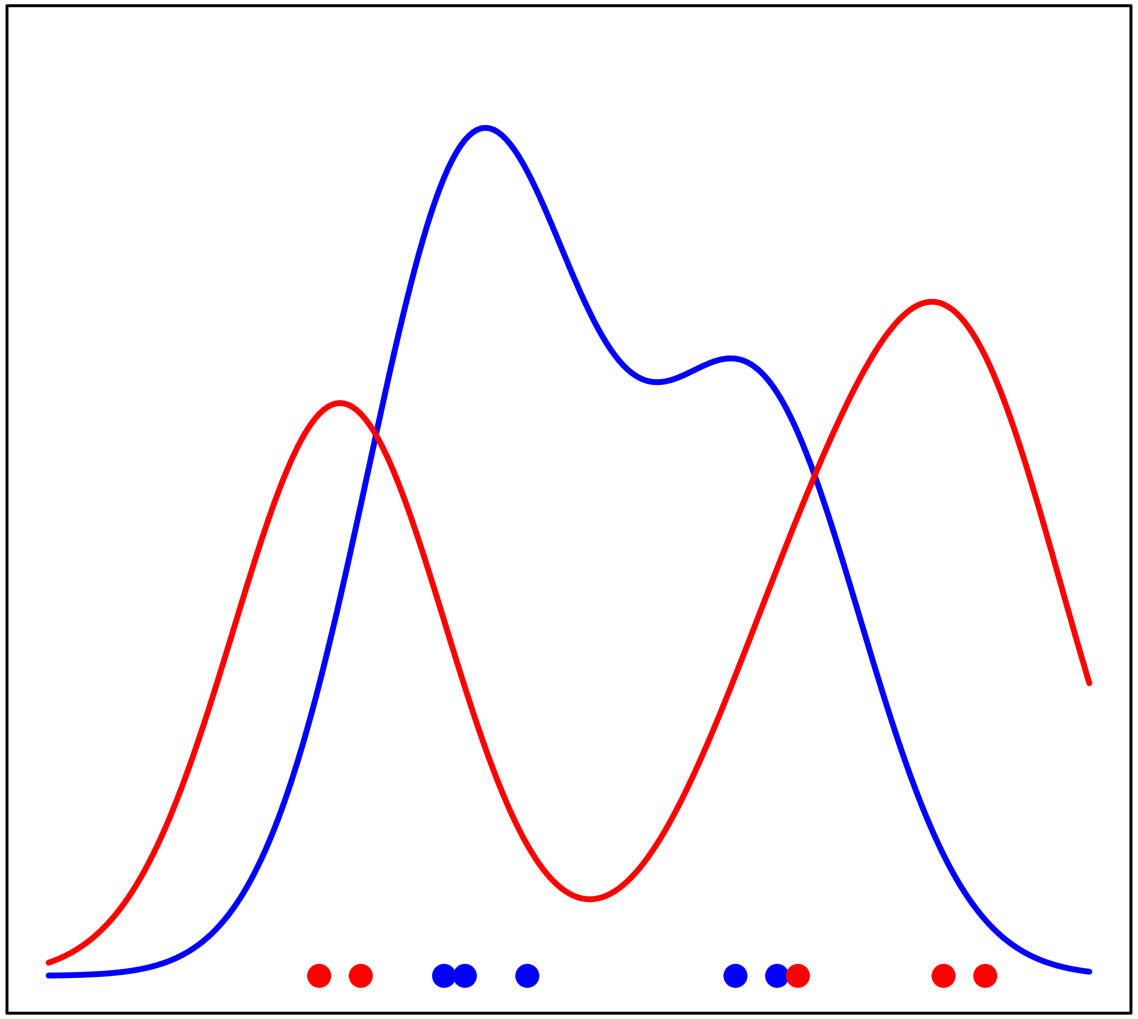
Gaussian kernels around
four observations

Kernel density estimators



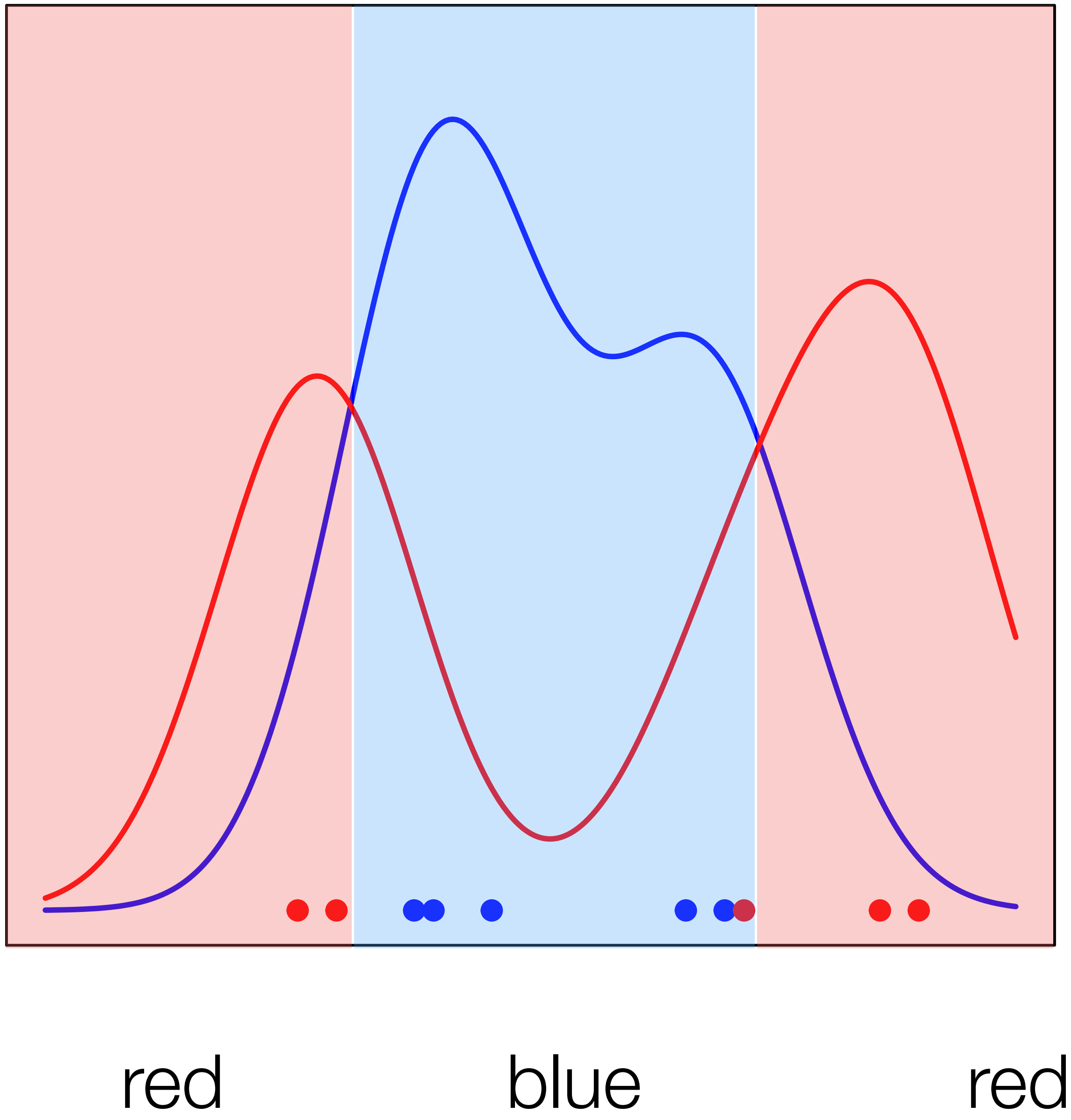
Gaussian kernels around five observations

Kernel density estimators



Gaussian kernels around five observations each from two categories

Kernel density estimators



Classification boundaries?

The cognitive science perspective

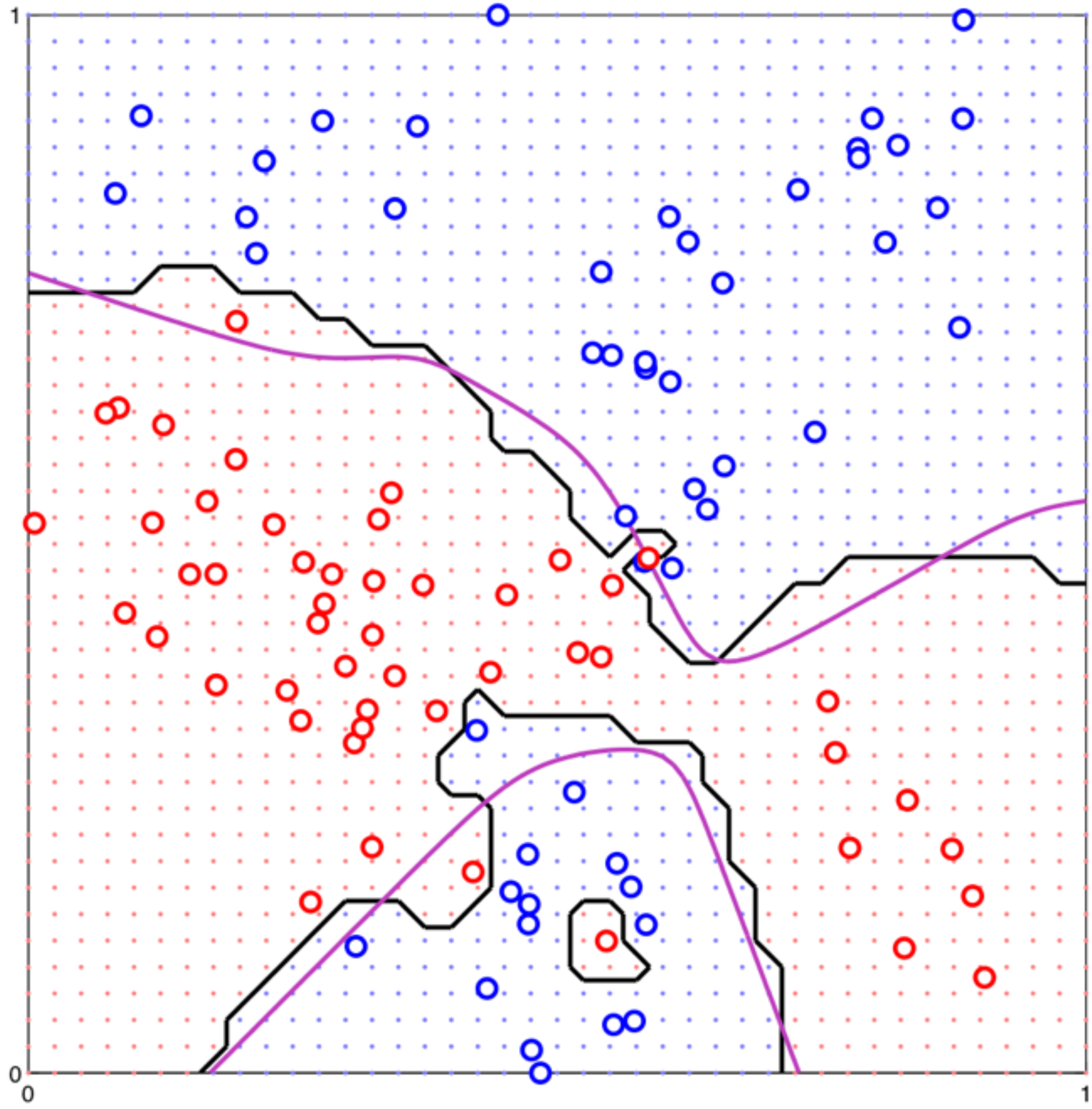
- Kernel density estimates are exemplar models
  - Each observation x is stored
  - The kernel describes the probability of generalising from a single stored x to a new item y
- In particular…
  - The exponential kernel is special (Shepard's law of generalisation)
  - It produces a model known as the "generalised context model"
  - (Nosofsky, 1984, 1986)

$$P(y|x_1, \ldots, x_n) \propto \sum_{i=1}^{n} \exp(-\lambda \; d(y, x_i))$$

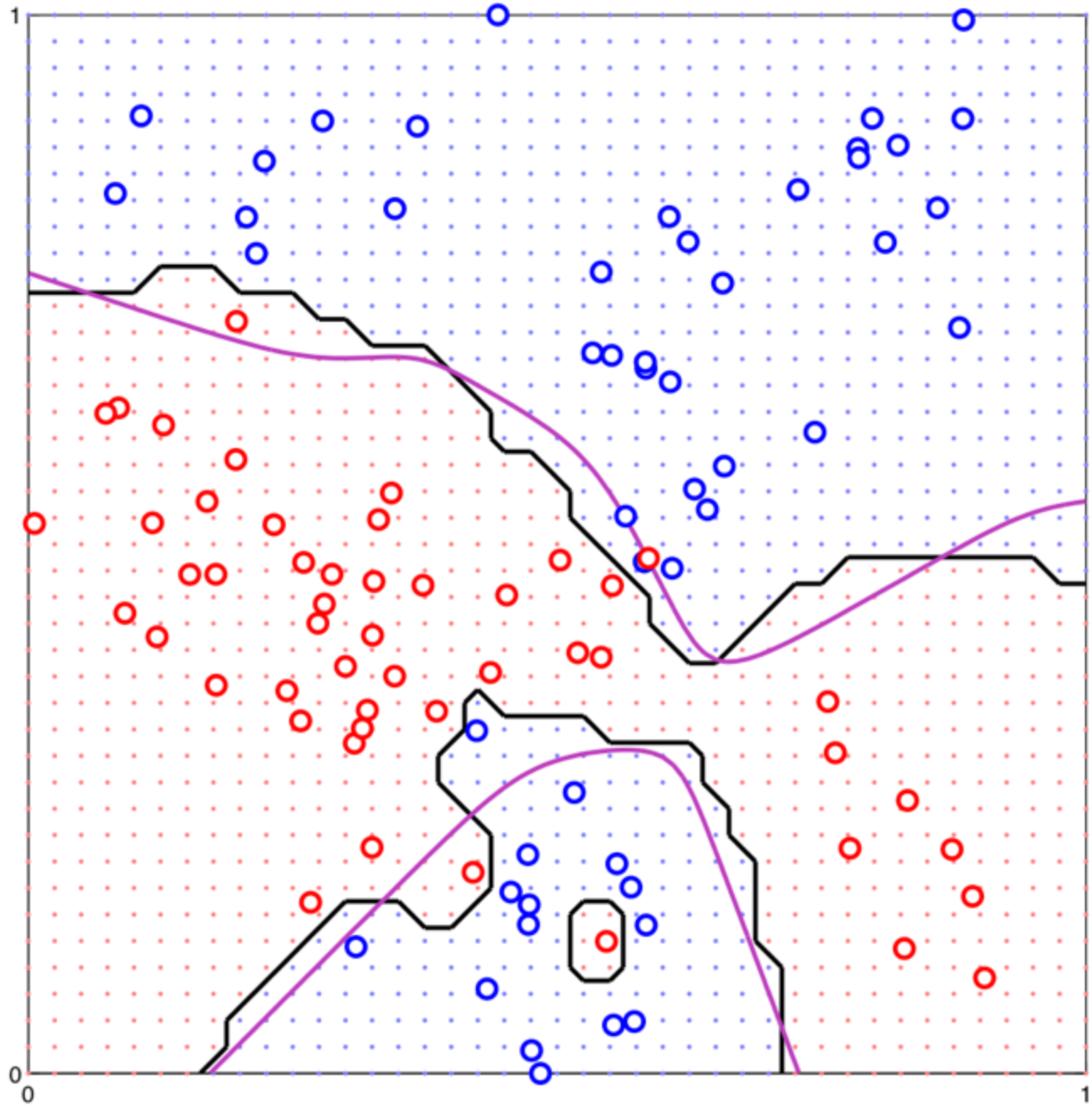
Demonstration code

(classifiers.R, kernelClass function)

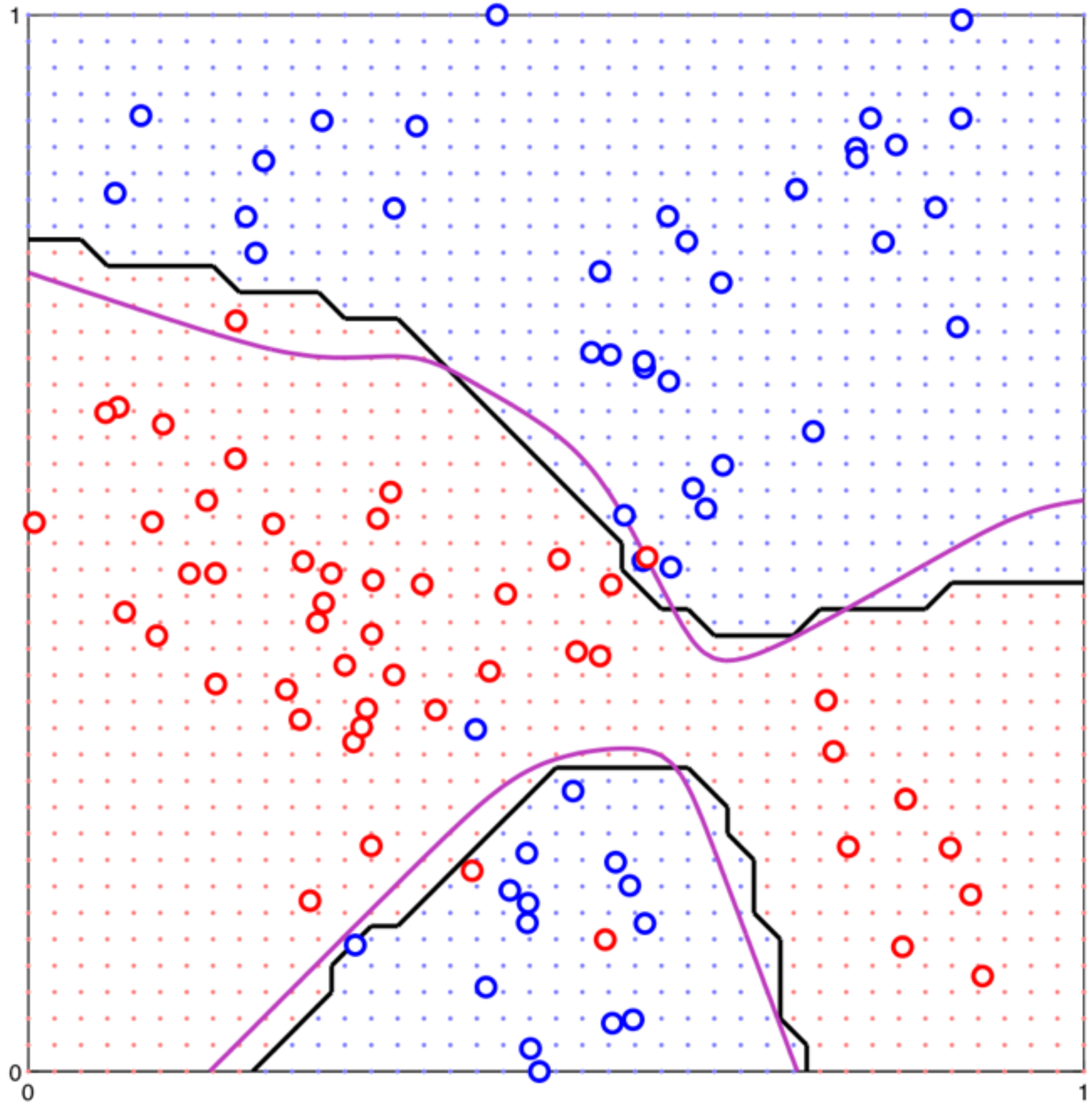
$\lambda = 1000$



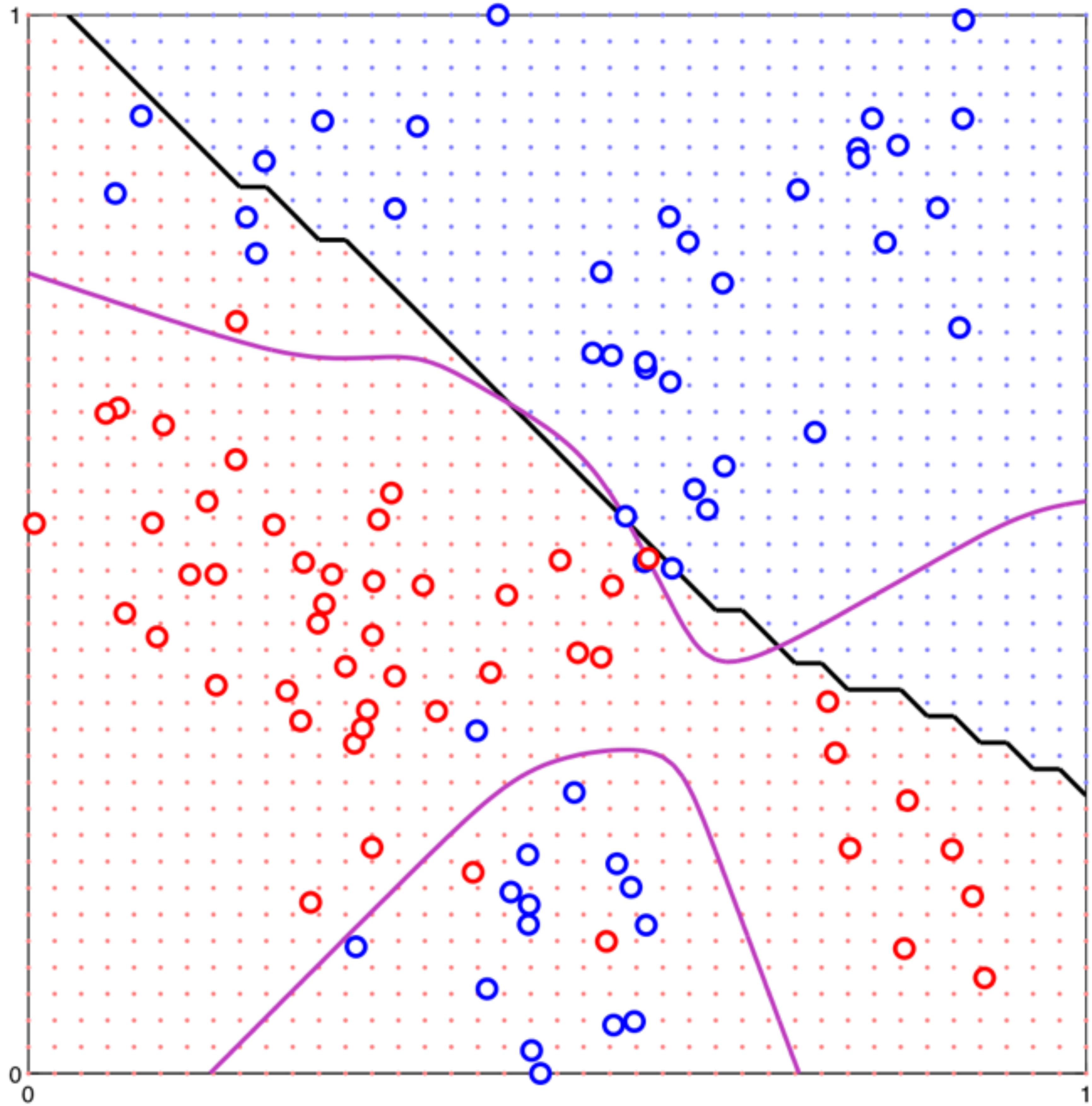
λ = 100



$\lambda = 10$

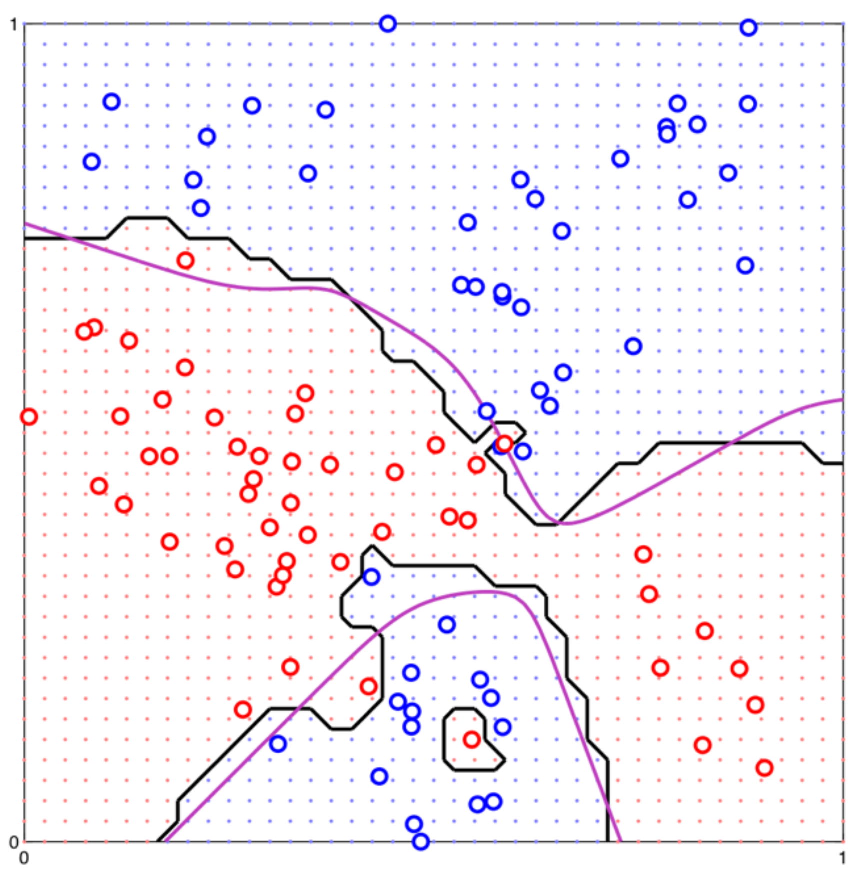


$\lambda = 1$

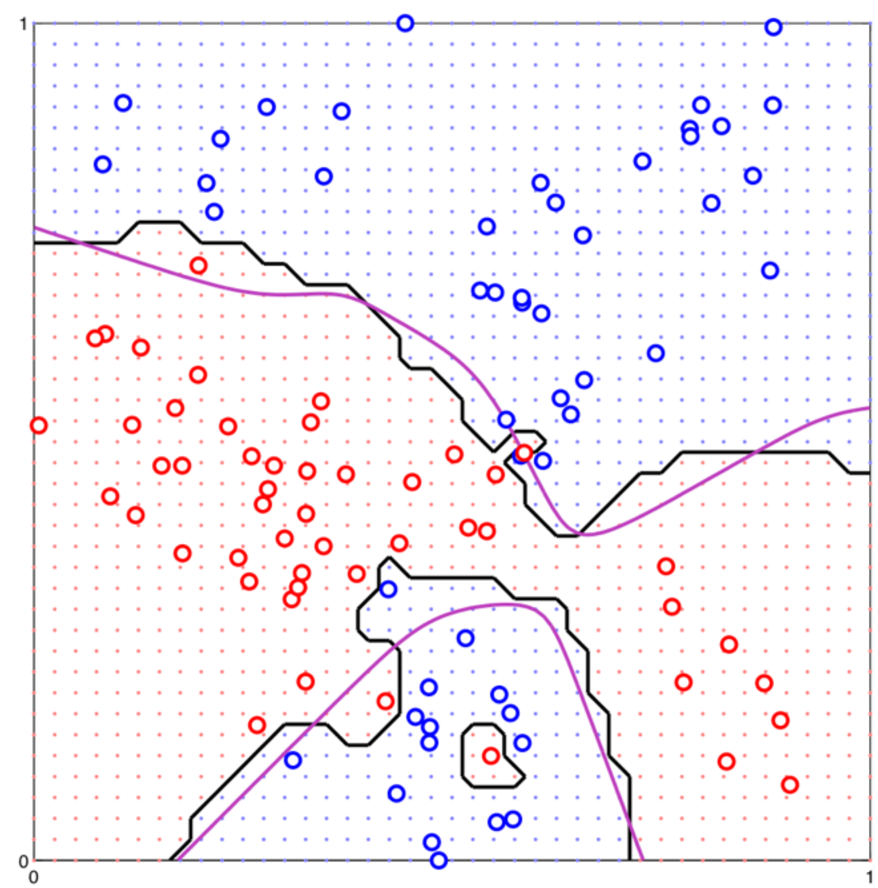


Very large λ behaves like 1NN

1 NN



λ = 1000



The cognitive science perspective

- The “generalised context model” (GCM)
  - Proposed by Nosofsky (1984, 1986)
  - Independent of the statistics literature on the topic!
  - Equivalent to an exponential kernel classifier

- Very successful model.
  - It’s very hard (not impossible) to beat the GCM as a cognitive model
  - It’s a very good predictor of human behaviour
  - It’s also simple, effective classifier

Training your classifier using cross-validation

(no demonstration code :-( )

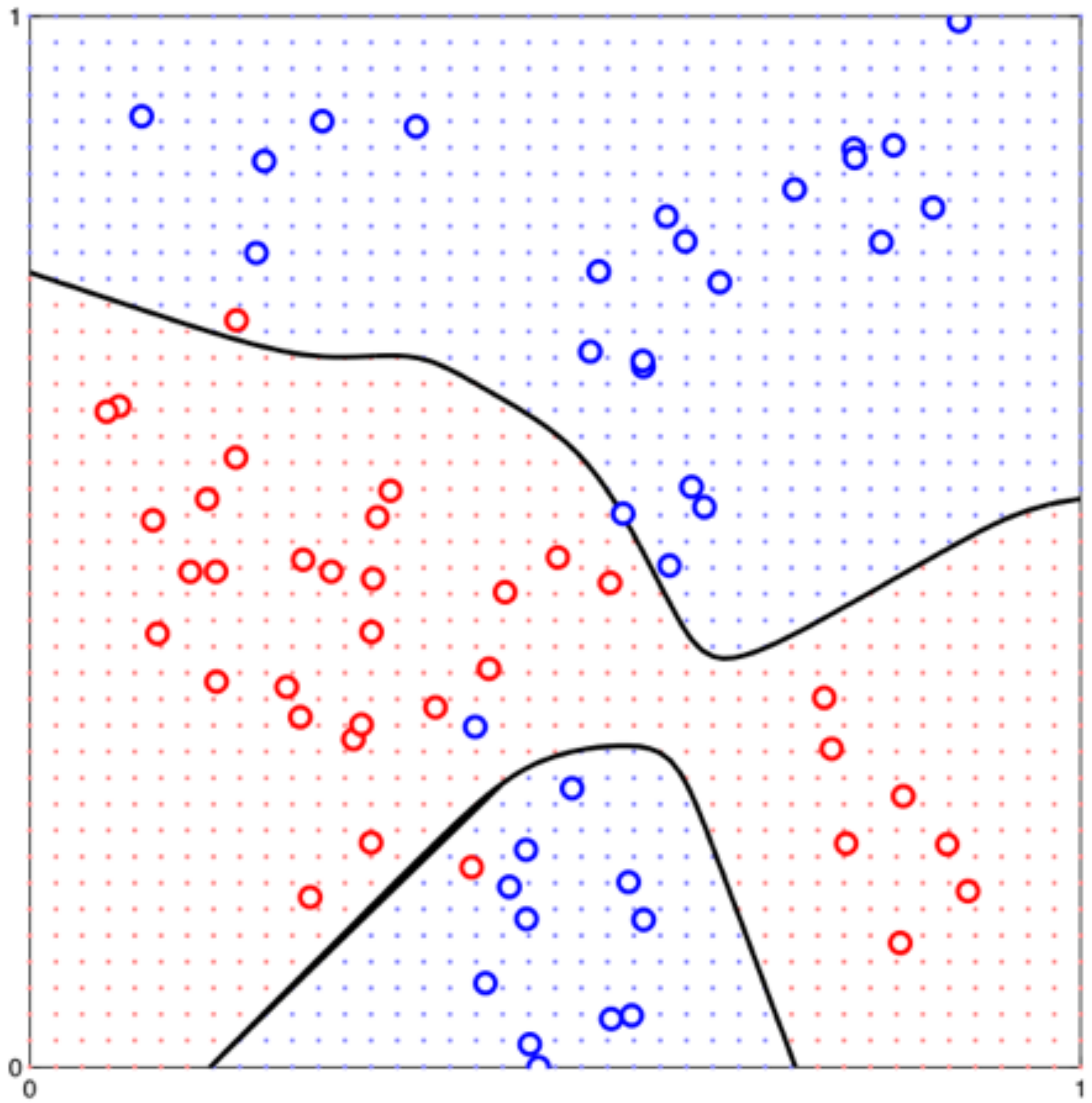
The issue

- Choosing good parameters
  - A lot of our algorithms have free parameters
  - k in k-NN, λ in the kernel method
- Model selection:
  - We have lots of competing classifiers
  - We want to know which is best
- Goal:
  - The goal isn't to select the classifier that best fits the training data
  - The goal is to select the one that best fits <u>future</u> data

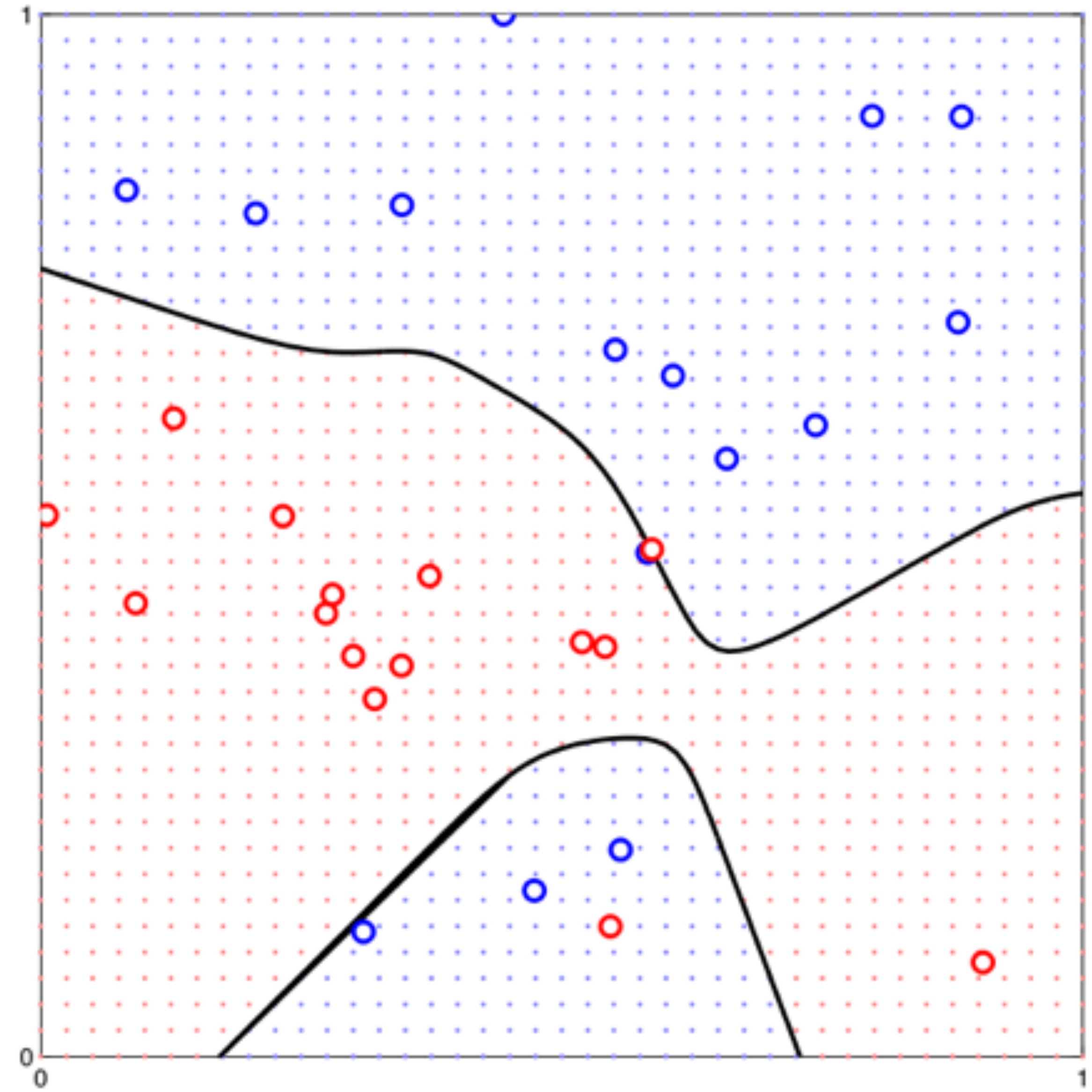
Cross validation

- What are we trying to do?
  - Select a model that is trained on X, and generalises well to Y
  - If we have several models, which will generalise best?
  - Which should we select?
- A simple suggestion:
  - Divide the training data X into two subsets, X1 and X2.
  - Train each model on X1 and test it on its predictions about X2.
  - Choose the model that makes the best predictions.

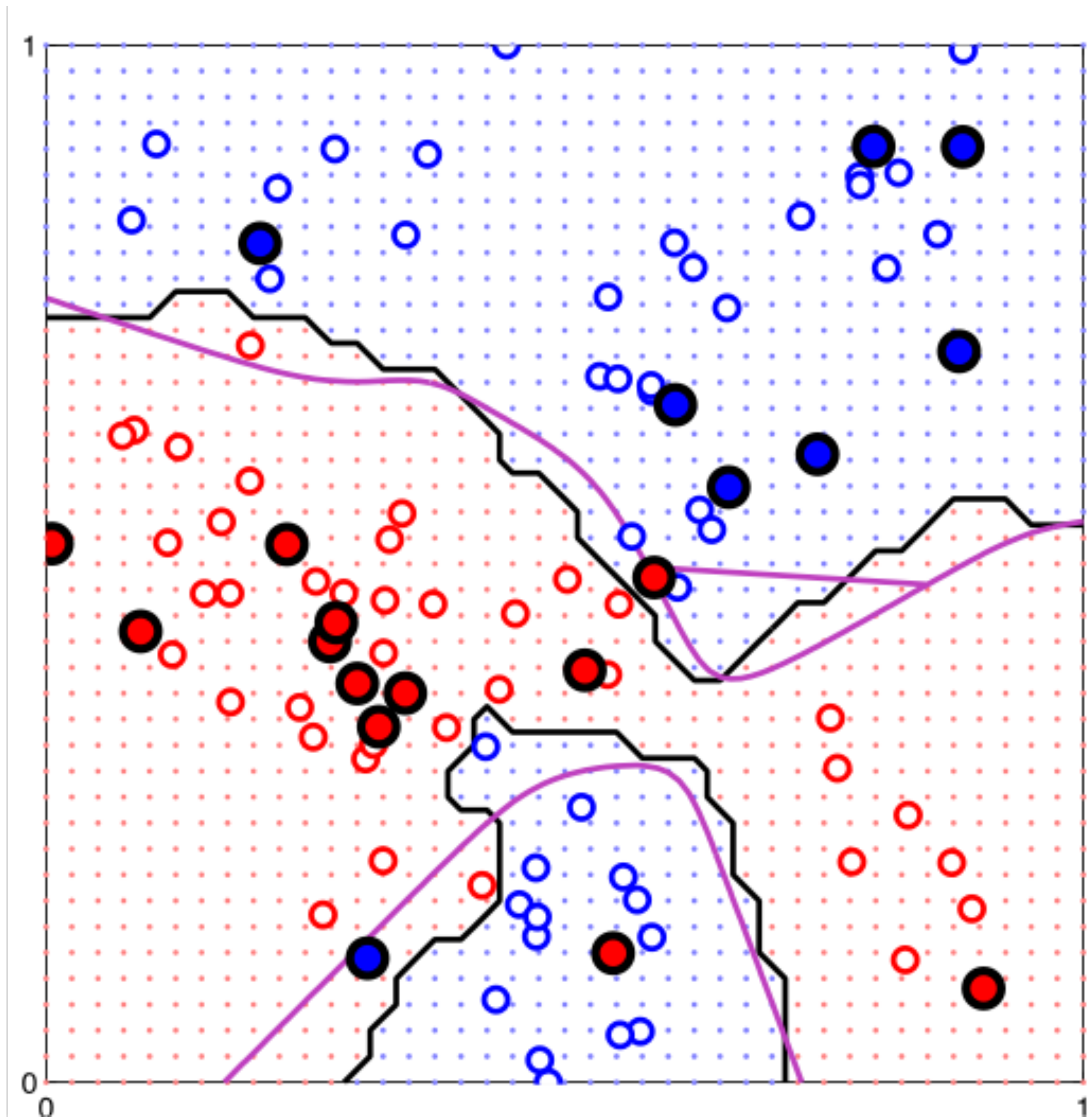
Train the model on this



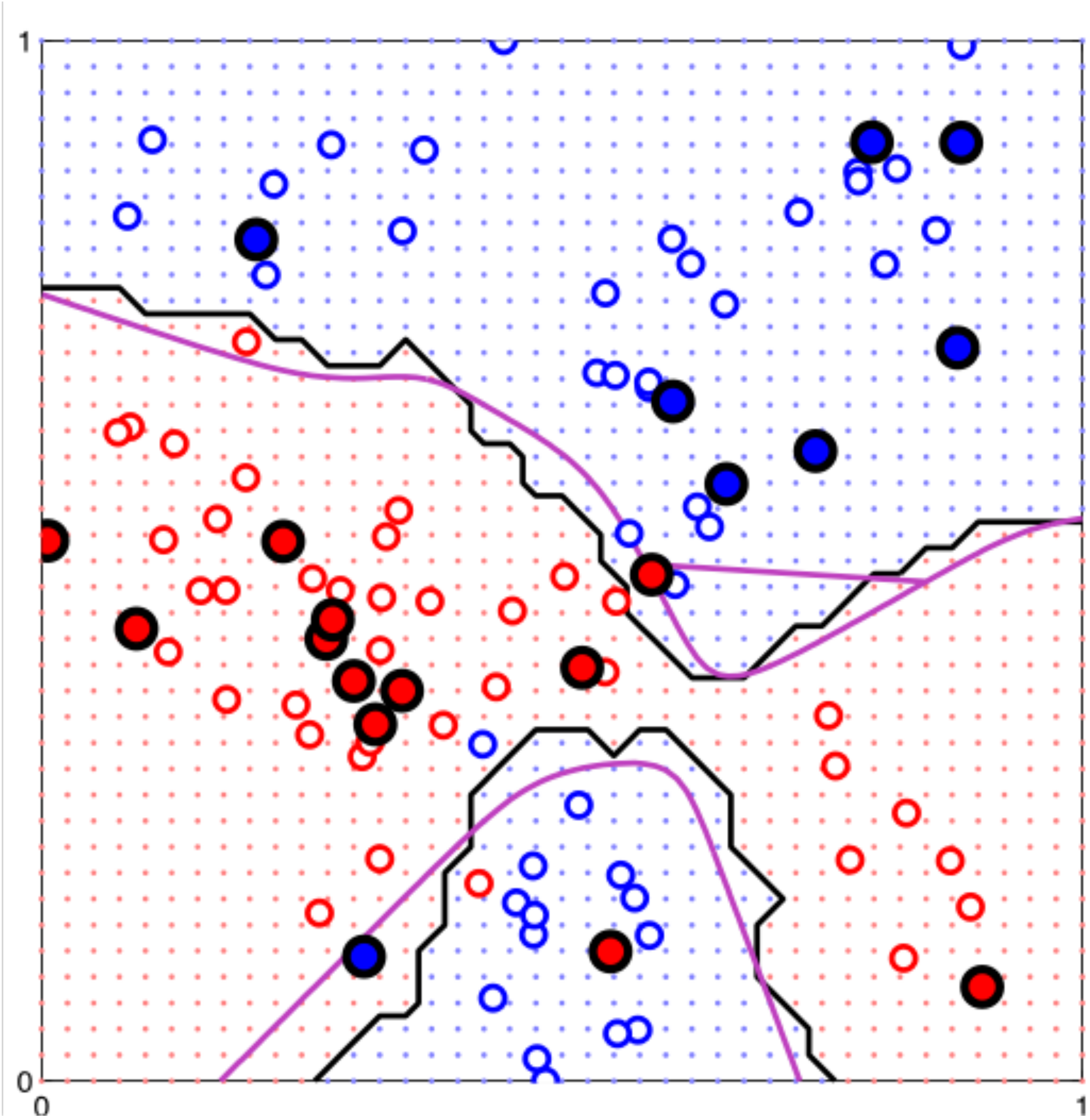
Test the model on this



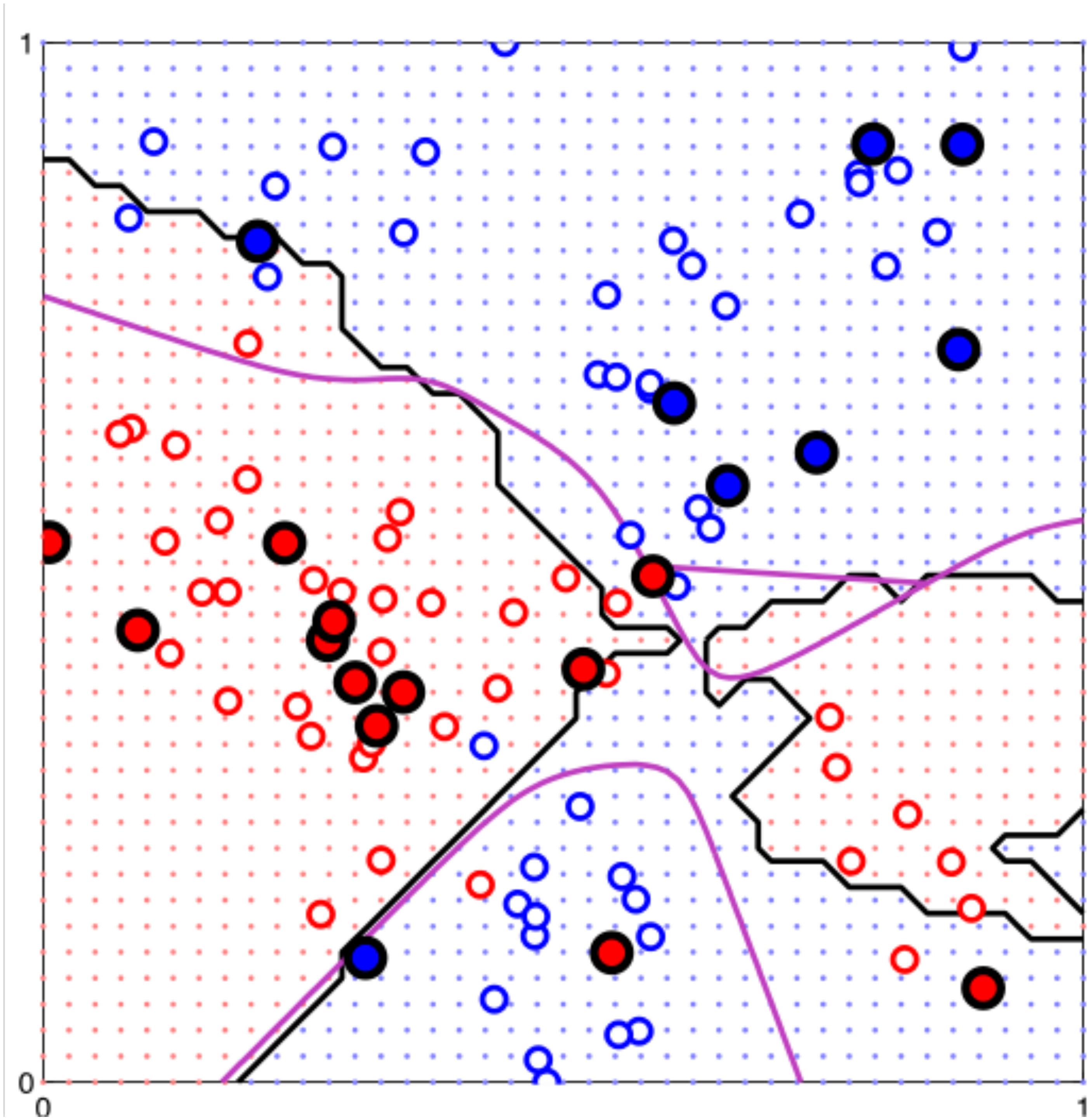
1-NN scores 17/20



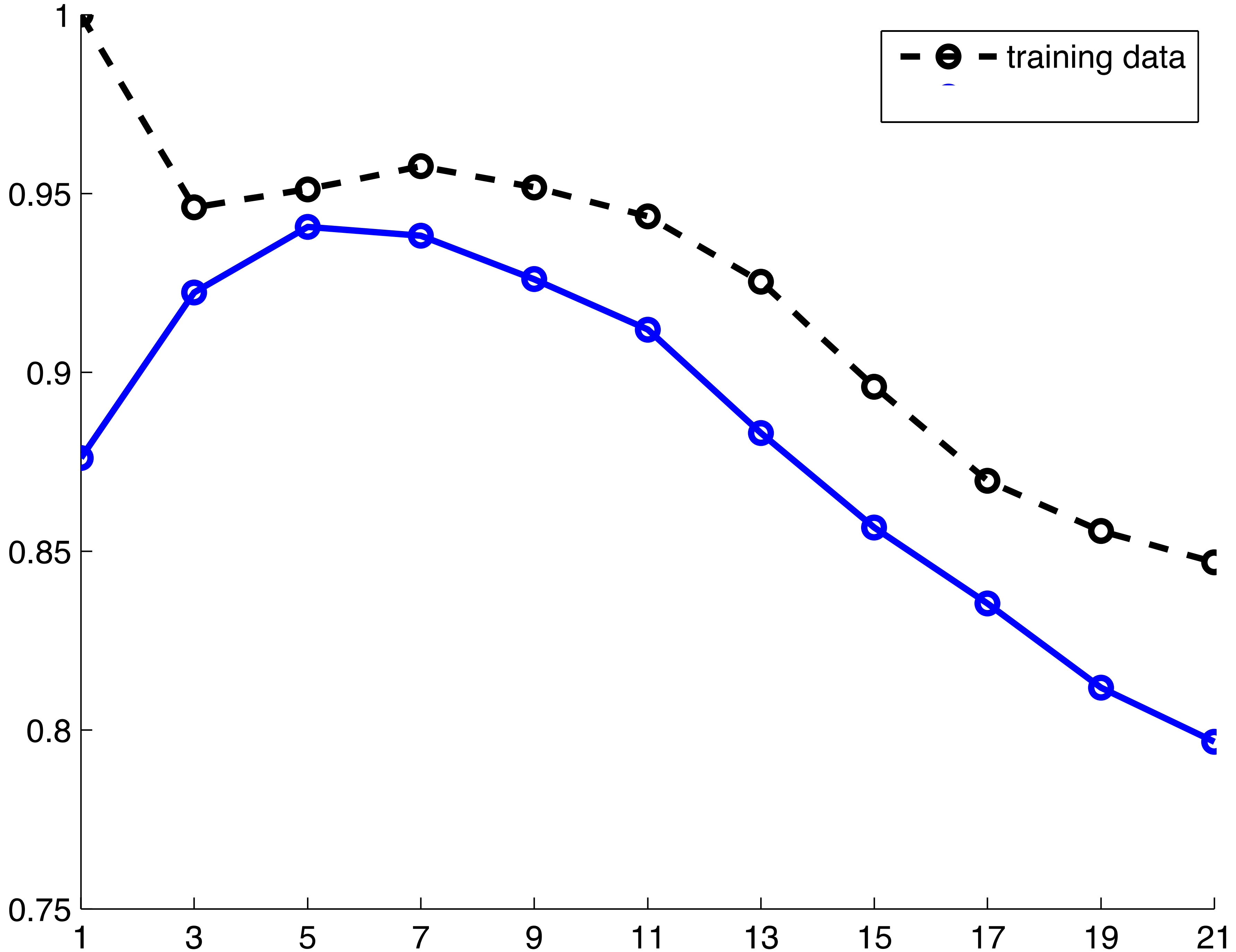
3-NN scores 17/20



15-NN scores 16/20



On average, across many splits



Summary

- Prototype-like classifiers
  - simple Gaussian classifier
  - multivariate Gaussian classifier
- Exemplar-like classifiers
  - k nearest neighbours
  - kernel classifiers
- Cross-validation
- Next time: unsupervised classification…