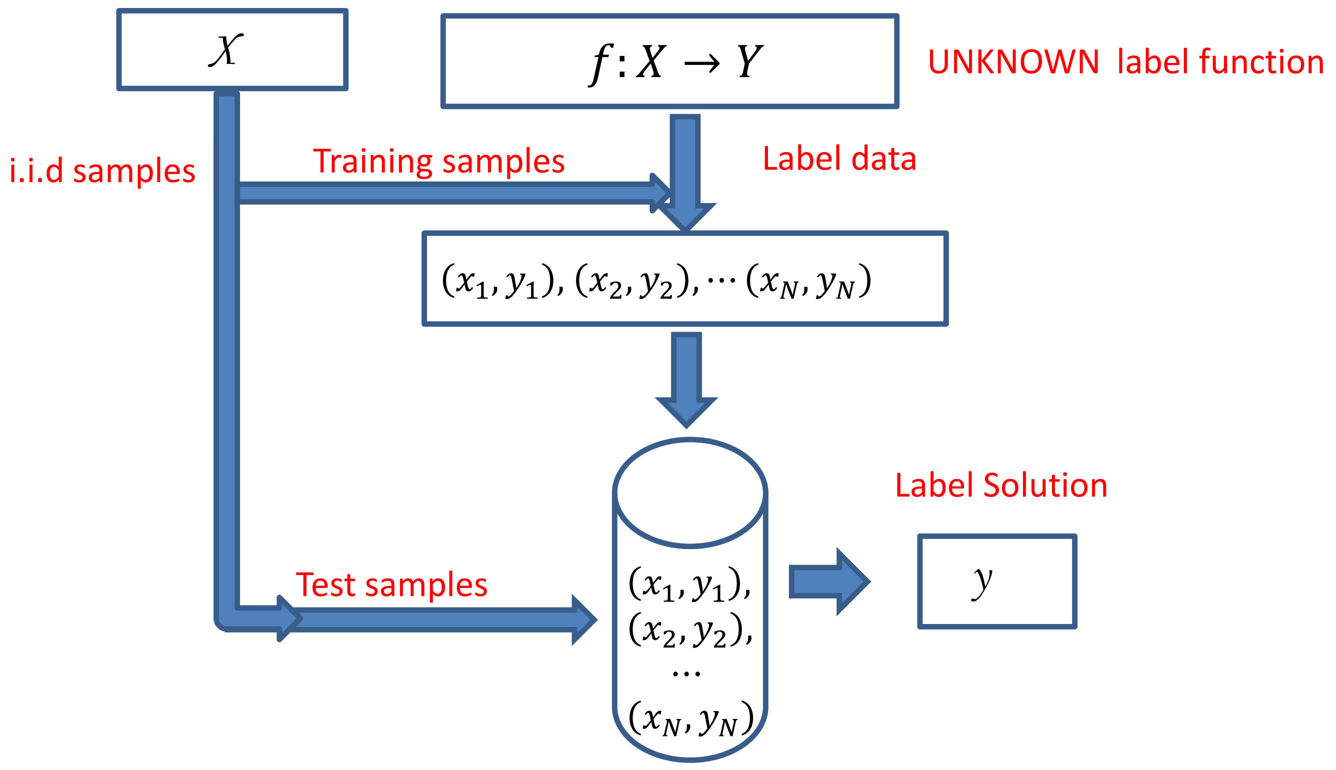
Learning by Similarity: the nearest neighbour rule (NN)

Nicolás Pérez de la Blanca
DECSAI-UGR

Learning by Similarity: diagramme



- **Data are used without any processing**
- **We do NOT fit any function on the training sample data**

- A test time labels are assigned by similarity
- This approach emphasizes memory instead of learning

Common disimilarity measures

- For $x, x' \in \mathbb{R}^d$
  - $d(x,x') = \|x - x'\|$ L2 or L1 norm
  - $d(x,x') = x^T Q x'$ Mahalanobis distance (account for the covariances)
  - $d(x,x') = \text{cosSim} = \frac{x^T x\prime}{\|x\| \|x\prime\|} \in [-1,1]$
- For A , B set of element
  - Jaccard coeficient: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$
- There exist many disimilarity (similarity) functions according to the properties we want to measure.

The Nearest Neighbourd Model

Two basic NN models

Elements of the Approach:

- A set of sample from $\mathcal{X}$ labeled by an unknown function f
- A distance/disimilarity function between items: $d(x, y): \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$

Basic Classification Algorithm **1-NN:**

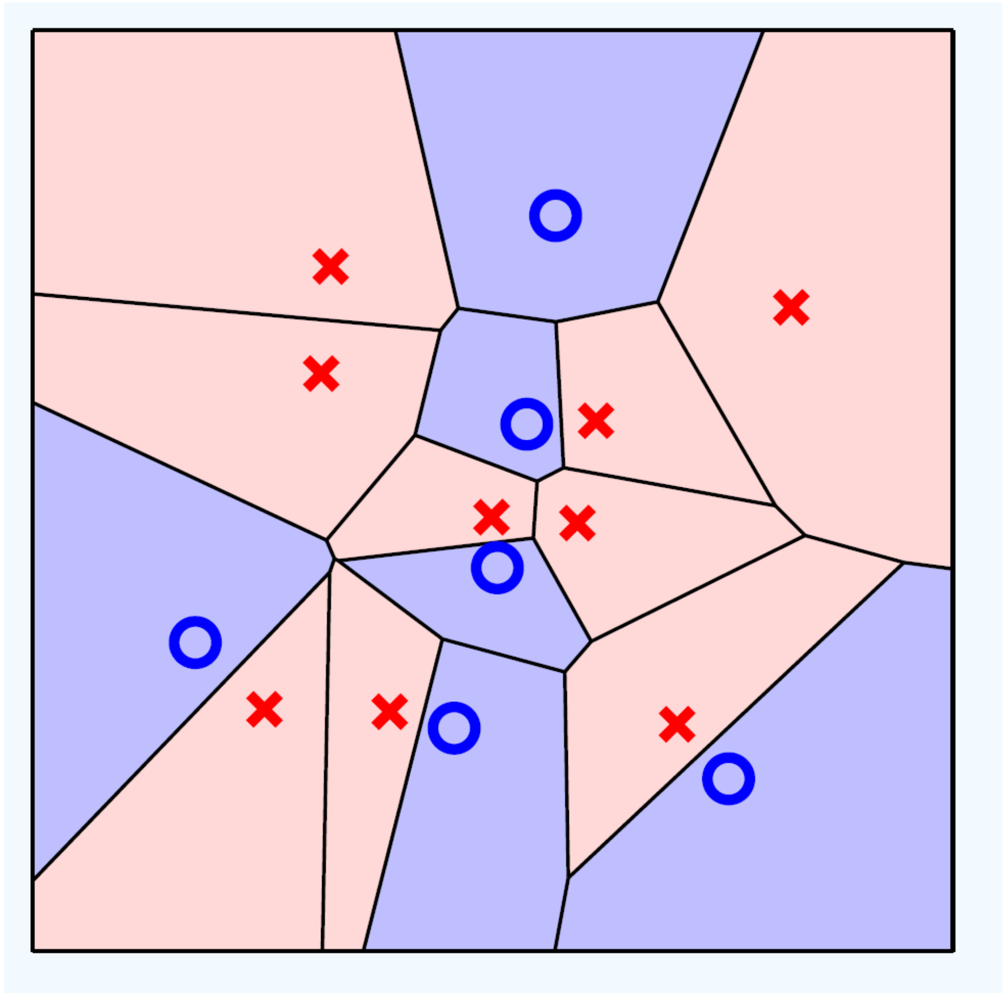
1. Save the sample data
2. For each new sample:
   1. Search the saved ítem with the mínimum distance to the sample
   2. Assign to the new sample the label of its minimun distance item

Classification Algorithm **k-NN:**

1. Save the sample data
2. For each new sample:
   1. Search the k saved items with the minimum distance to the sample.
   2. Assign to the new sample the majority label of its k minimum distance ítems.

- The distance/disimilarity function to use depend on the feature vectors
  - Example: Euclidean distance, Hamming distance, etc

2D Voronoi tessellation



- Partition of the 2D space using the 1-NN rule (two classes)
- ¿How are the border between each two regions?

Pros and Cons of the 1-NN rule

- Pros:
  - We do NOT need to learn any function: Only memorize data
  - Very simple classification rule
  - It works very well in many practical scenarios
- Cons:
  - Small error is only possible in points very near to the sample
  - Unknown tradeoff between distance and feature vector dimension

its simplicity is misleading !!

- Relevant questions:
  - How many samples do we need to guarantee a small test error ?
  - How relevant is the size of the feature vector?
  - What if instead of using the 1-NN we use the k-NN ?

VC dimension

- 1-NN rule: infinity VC dimension
  - Why ?

- Then, how to estimate the prediction error?
  - There exist a very strong connection between the Bayes error and the NN-error

- Bayes Rule: Given any probability distribution $\mathcal{P}$ over $\mathcal{X}\times\{0,1\}$, the best label predicting function from $\mathcal{X}$ to $\{0,1\}$ will be

$$f_{\mathcal{P}}(x) = \begin{cases} 1 & if\ \ \mathbb{P}[y=1|x] \geq \mathbb{P}[y=0|x] \\ 0 & otherwise \end{cases}$$

  - Bayes Error is the probability of misclasification
  - An error bound for the 1-NN rule can be established using the Bayes error

Generalization bound for the 1-NN rule

- Let $\mathcal{X} = [0,1]^d$, $\mathcal{Y} = \{0,1\}$, $\mathbb{P}$ a probability distribution on $\mathcal{X} \times \mathcal{Y}$ and loss function 0-1, this is $\ell\big(h, (\boldsymbol{x}, y)\big) = \mathbb{1}[h(\boldsymbol{x}) \neq y]$.

- Let $\eta(\mathbf{x}) = \mathbb{P}[y = 1 \mid \mathbf{x}]$ the conditional probability function on the labels

- We assume that this probability function is c-Lipschitz for some c > 0

$$\text{for all } \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X},\ |\eta(\boldsymbol{x}) - \eta(\boldsymbol{x}')| < c\|\boldsymbol{x} - \boldsymbol{x}'\|$$

- This says that if two vectors are closed to each other their labels are likely to be the same

- The constant *c* talk us about the complexity of the distribution $\eta(\boldsymbol{x})$
  - *c* small implies soft local behaviour
  - *c* large implies strong local changes

Main result

- This result shows us how is the dependency of the generalization error from the Bayes rule, the **number of samples** and the **dimension of the feature** vector

- **NN-Main Result**: Let $\mathcal{X}$= $[0,1]^d$ and $\mathcal{P}$ be the distribution over $\mathcal{X}\times\mathcal{Y}$ for which the conditional distribution η is a c-Lipschitz function. Let h_S denote the result of applying the 1-NN rule to a sample $S \sim \mathcal{P}^N$. Then,

$$\mathbb{E}_{S\sim\mathcal{P}^N}[L_{\mathcal{P}}(h_S)] \leq 2L_{\mathcal{P}}(h^*) + 4cd^{1/2}N^{-(\frac{1}{d+1})}$$

  where h^* is the Bayes rule (minimum error)

- What is relevant here is to analyze the implication of *c, d* and *N* in order to get

$$4cd^{1/2}N^{-(\frac{1}{d+1})} < \epsilon$$

The curse of dimensionality

- From the above result the number of samples needed to get an added error $< \varepsilon$ is

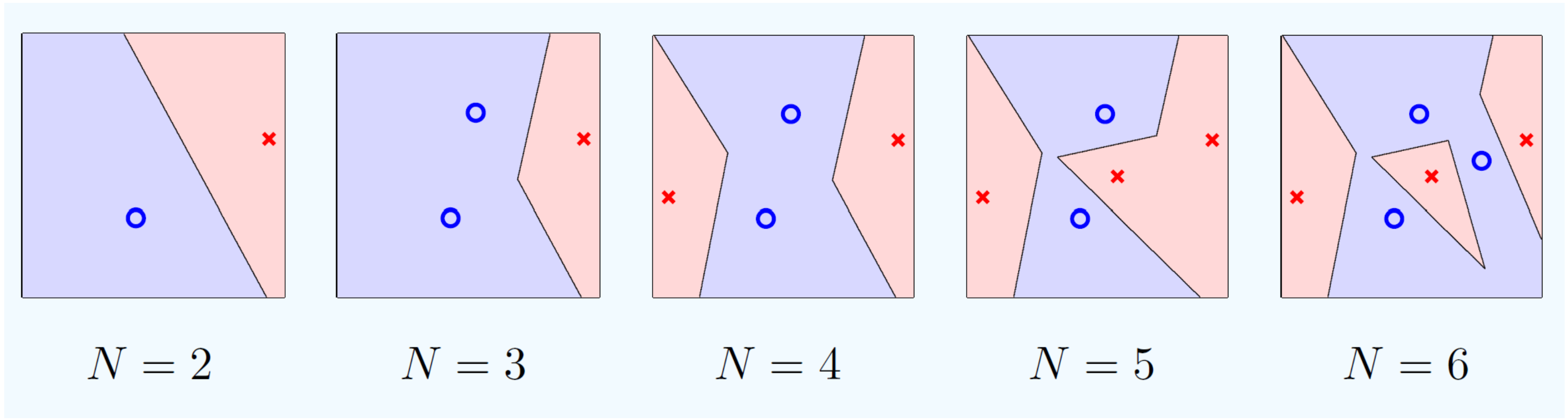
$$N > \left(\frac{4cd^{1/2}}{\epsilon}\right)^{d+1} = \mathcal{O}(d^d)$$

- This shows an exponential increasing with the vector dimension !!
- Conclusion: NN is only applicable with low dimensionality vectors.

- **Lower bound result:** The 1-NN rule needs at least $\frac{(c+1)^d}{2}$ examples to succeed (bounded error).
  - This shows that complex label function requires a large number of samples

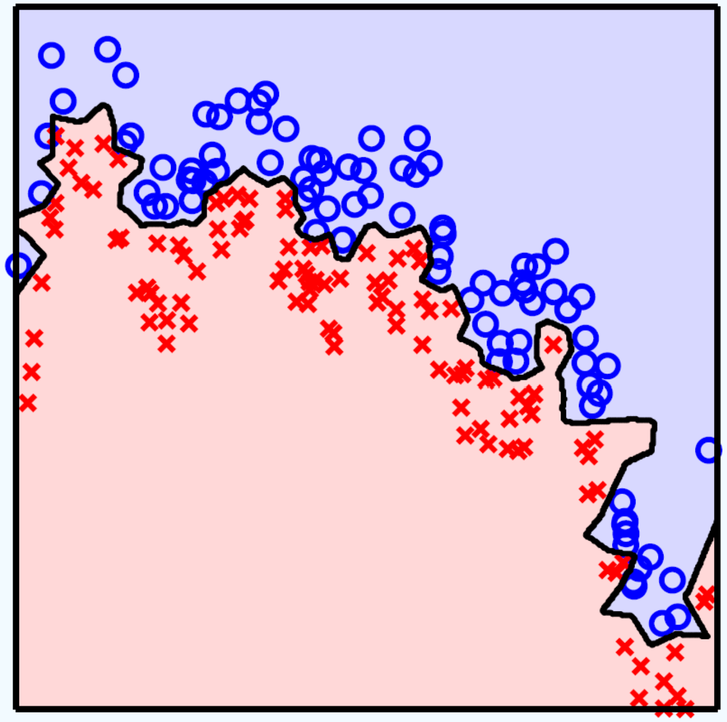
- What to do when we have large dimensionality vectors?
  - Dimensionality reduction
  - Approximate search: kd-tree, LSH, etc

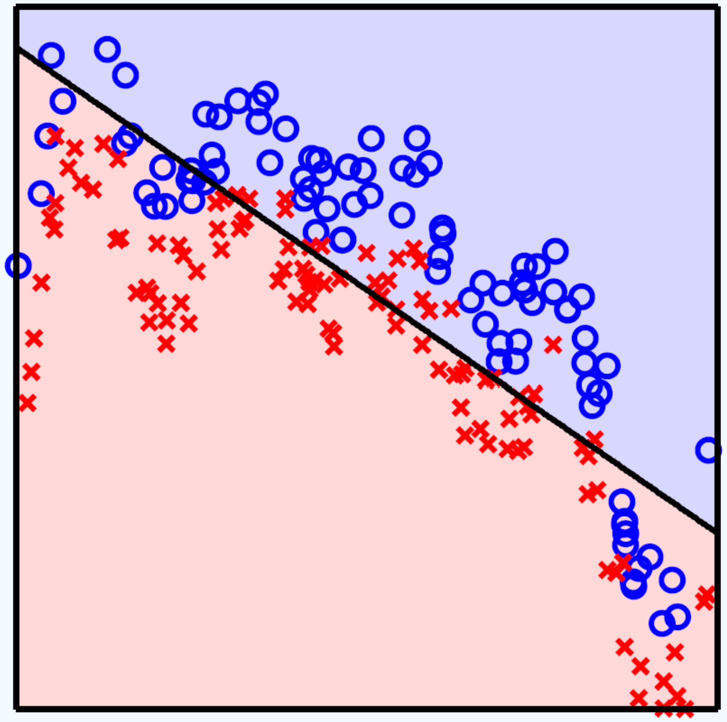
Function complexity increases with N



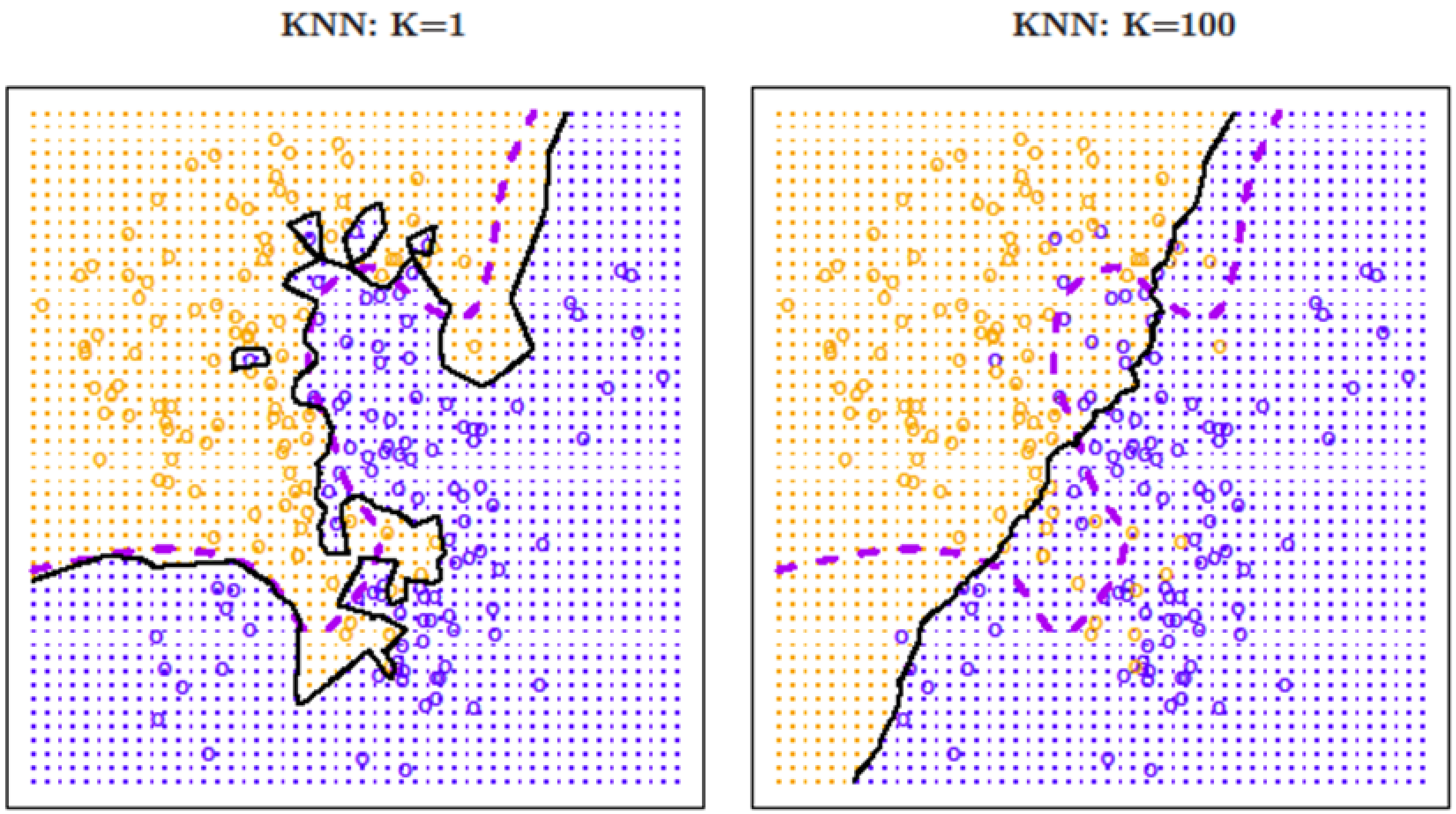
- This example shows how the number of samples influence the complexity of the NN solution
- This means the complexity of the function solution depends on the data and NOT of a set of parameters.

NN is nonparametric


(a) Nonparametric NN

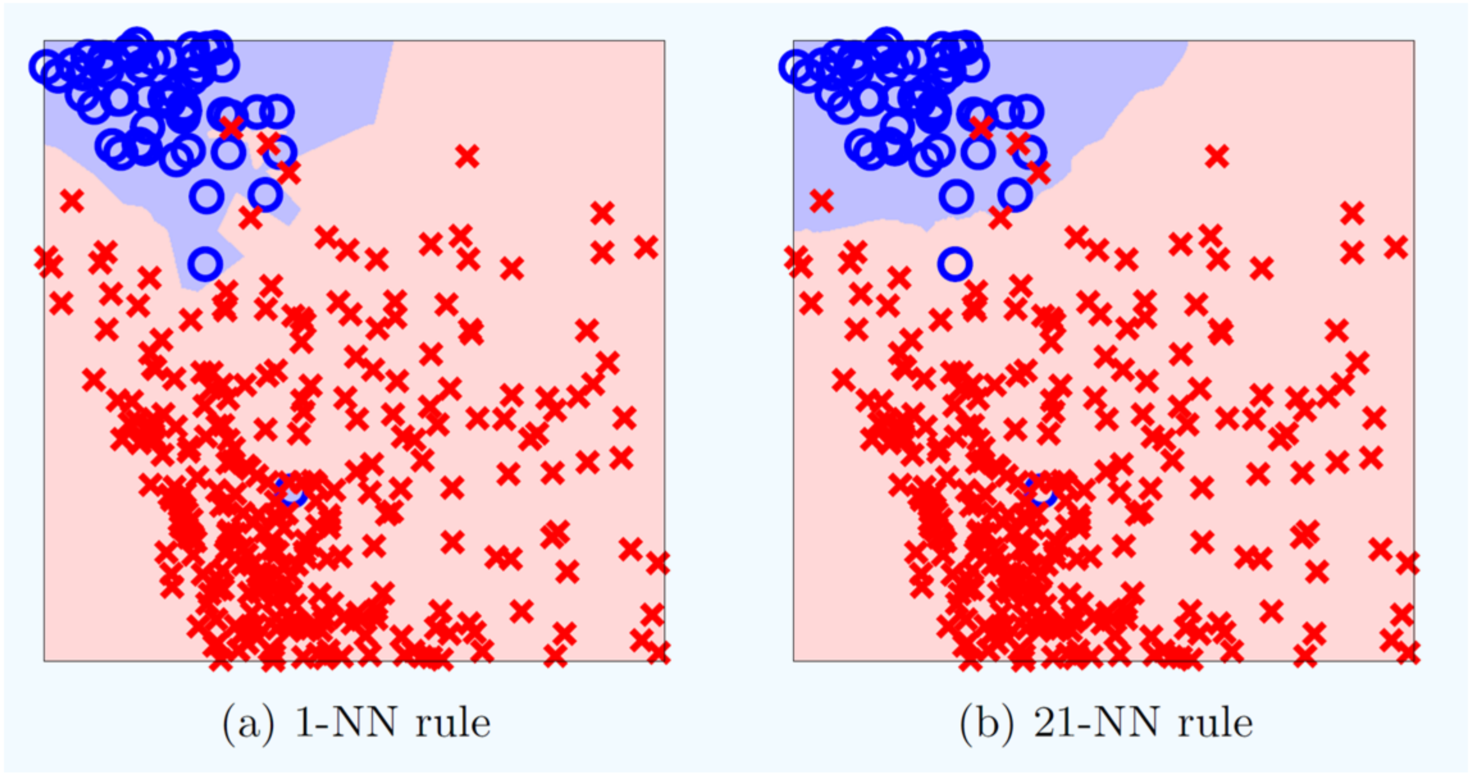

(b) Parametric linear

Solution complexity decrease with K



- The blue line represents Bayes decision function, this is the curve of minimum error separating the classes

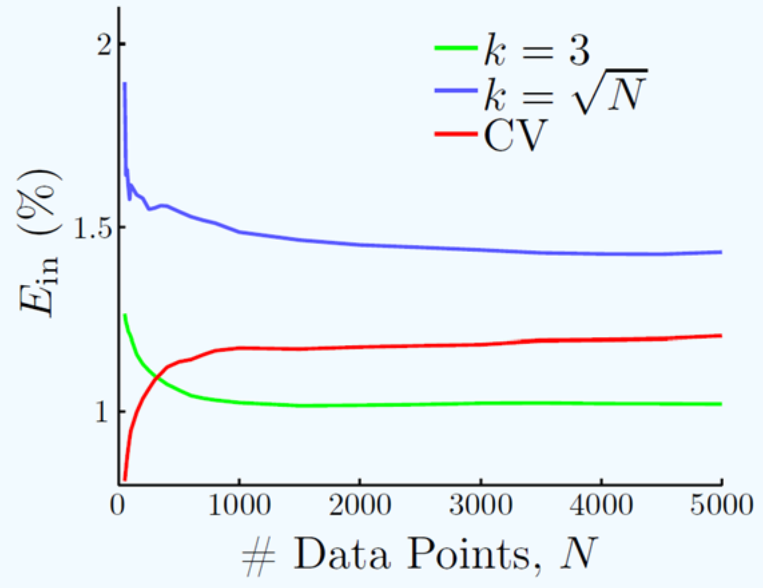
Aproximation vs Generalization



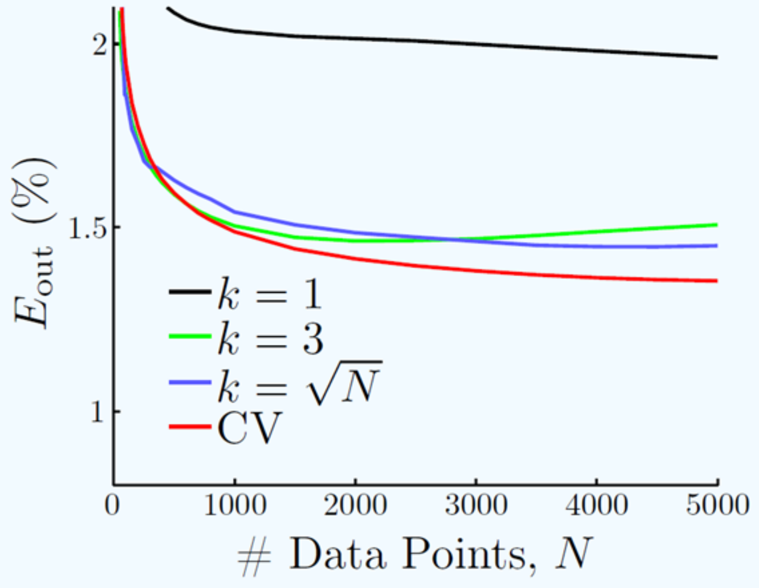
- Digit classification using NN : What is the best value for k?

- **Result**: For $N \to \infty$ if $k(N) \to \infty$ and $k(N)/N \to 0$ then
  $E_{in}(g) \to E_{out}(g)$ and $E_{out}(g) \to E^{*}_{out}$ (Bayes error)

Effective size for k



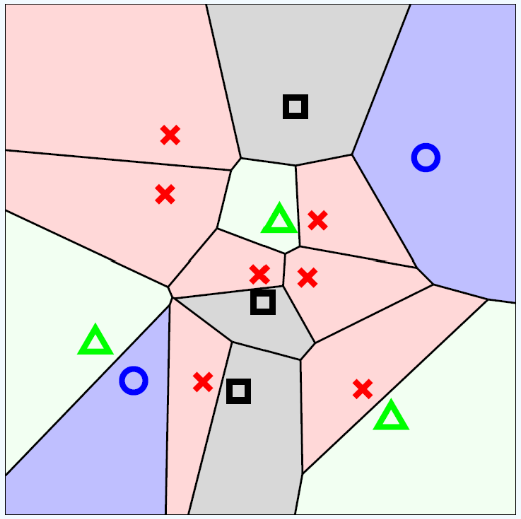
(a) In-sample error of k-NN rule

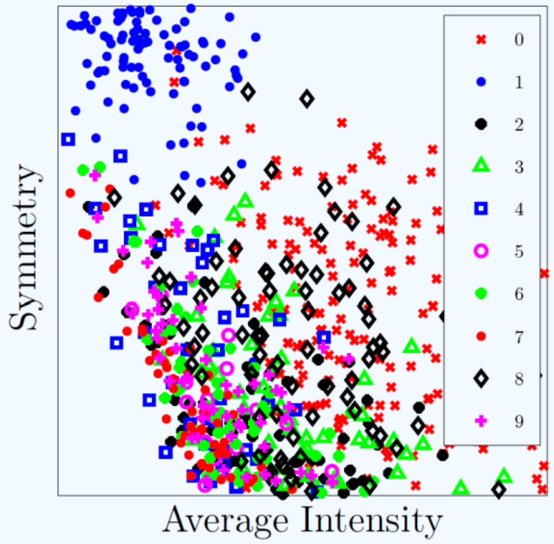


(b) Out-of-sample error of k-NN rule

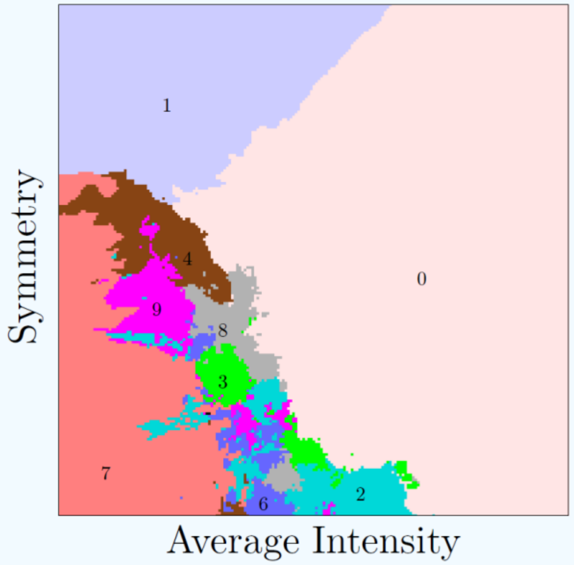
- These figures show the error as a function of the neighbourd size
- A very good compromise between between k-size and error is taken by k=3

NN multiclass





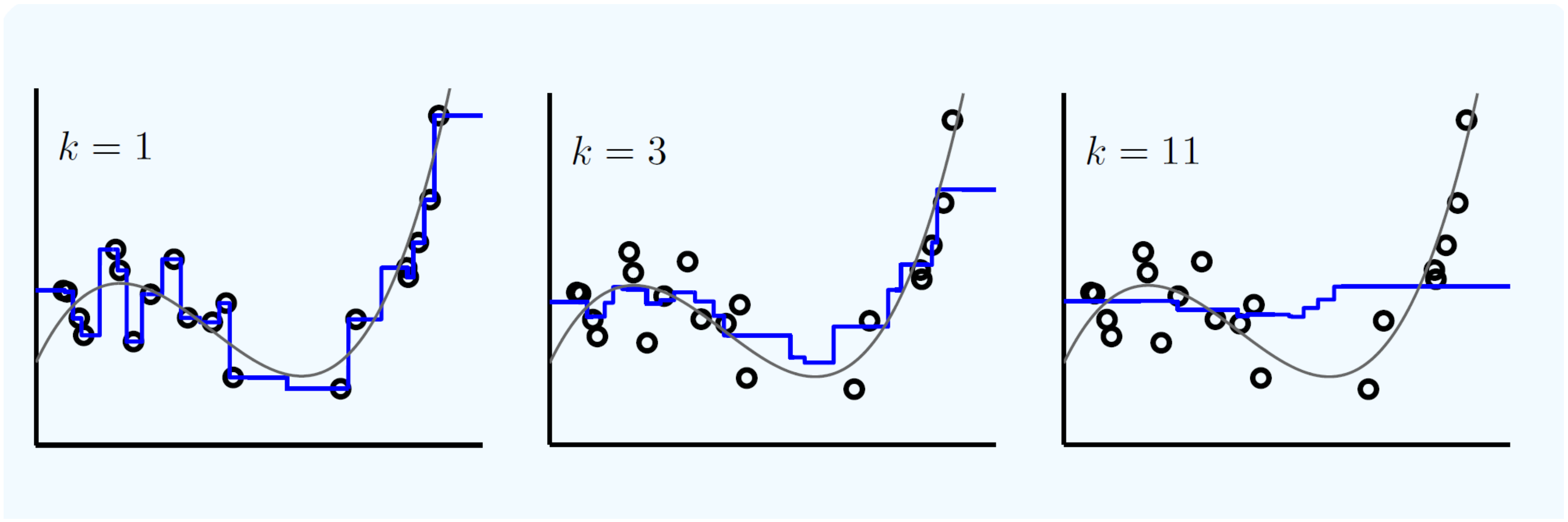
(a) Multiclass digits data



(b) 21-NN decision regions

- Very natural classifier: each pixel assigned to the more frequent class

NN: Regression and Probability estimation



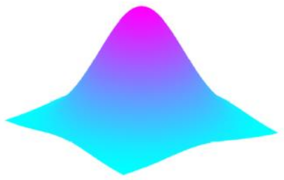
$$g(\boldsymbol{x}) = \frac{1}{k}\sum_{i=1}^{k} y_{[i]}(\boldsymbol{x})$$ k-NN Regression

$$g(\boldsymbol{x}) = \frac{1}{k}\sum_{i=1}^{k} [\![y_{[i]} = +1]\!]$$ k-NN Probability estimation

Radial Basis Functions

- The Radial Basis Functions technique generalizes the prediction model of the NN rule.
- Instead of using only k points to define the prediction, it allows all points contribute, but with diferent weight
- A radial basis function (or kernel) ϕ quantifies how the contribution decays as the distance increases

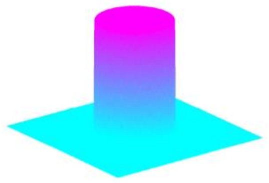
$$\phi(z) = e^{-\frac{1}{2}z^2}$$



$$\phi(\|x - x_n\|$$

defines the influence of x_n at x

$$\phi(z) = \begin{cases} 1 \ \ if \ z \leq 1 \\ 0 \ \ if \ z > 1 \end{cases}$$



For this model the kernel does not need to be normalized

General Predictor

- The kernel function can be generalized to depend on a scale parameter

$$\alpha_n(x) = \phi\left(\frac{\|x - x_n\|}{r}\right)$$

The scale r determine the "unit" of length with which the distance are measured.

$\alpha_n(x)$ measure the influence of x_n at x in units of r

The name "Radial" is becouse the influence only depends on the radial distance $\|x - x_n\|$

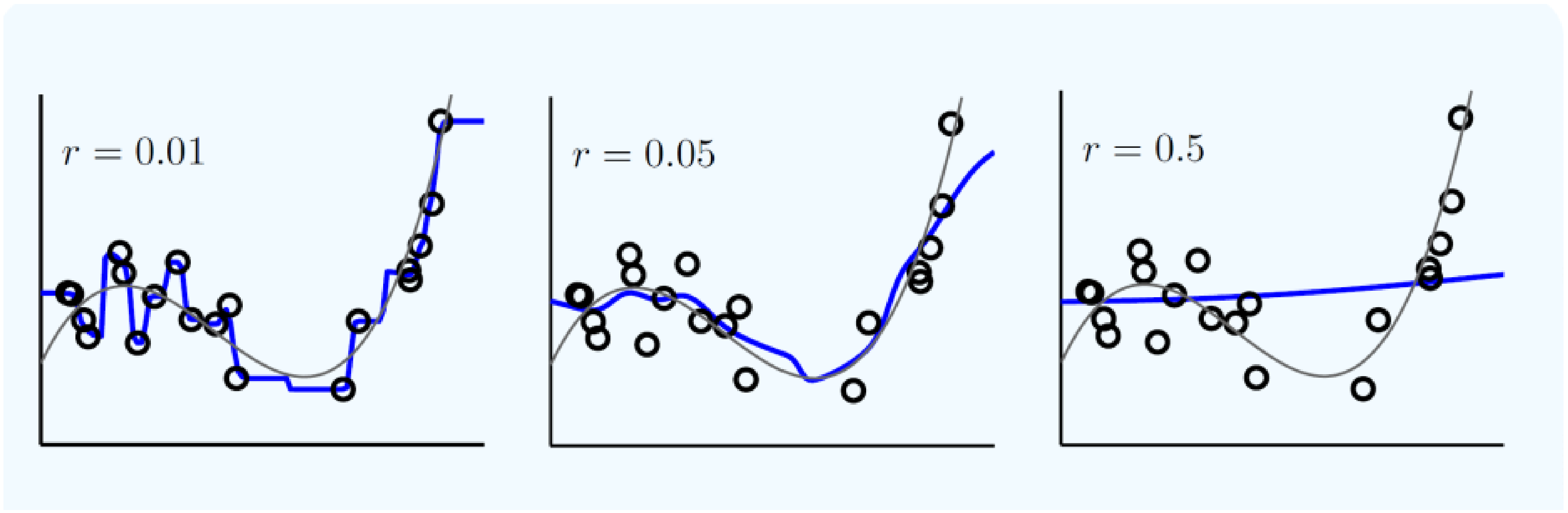
We compute $g(\mathbf{x})$ as the weighted sum of the y-values: $g(\boldsymbol{x}) = \dfrac{\sum_{n=1}^{N} \alpha_n(x) y_n}{\sum_{m=1}^{N} \alpha_m(x)}$

Regression: $g(\mathbf{x})$ is the k-NN generalization

Classification: sign($g(\mathbf{x})$) will be used

Logistic regression: $\theta(g(\mathbf{x}))$ will be the estimation. θ(x) is the sigmoidal

The influence of the scale



The figure shows the influence of the scale on the estimated function. Too small an r results in a complex hypothesis that overfits. Too large an r results in an excessively smooth hypothesis that underfits

- One way to choose *r* is using cross validation.
- Another is to use the convergence result as a guide:

$$r = (\sqrt[2d]{N})^{-1} \; equivalent \; to \; k \approx \sqrt{N}$$

Radial Basis Function Networks

- There are two ways of interpreting the prediction equation:

1.- $$g(\boldsymbol{x}) = \frac{\sum_{n=1}^{N} \alpha_n(\boldsymbol{x}) y_n}{\sum_{m=1}^{N} \alpha_m(\boldsymbol{x})}$$

Weigthed sum of y values. This corresponds to fix only a kernel at $\boldsymbol{x}$ and compute the influence at $\boldsymbol{x}$, $\alpha_n(\boldsymbol{x})$, of the samples.

2.- $$g(\boldsymbol{x}) = \sum_{n=1}^{N} w_n(\boldsymbol{x}) \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_n\|}{r}\right), \qquad w_n(\boldsymbol{x}) = \frac{y_n}{\sum_{m=1}^{M} \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_m\|}{r}\right)}$$

This second option corresponds to fix a kernel at each point $\boldsymbol{x}_n$ and compute the influence at $\boldsymbol{x}$ of each kernel. The w_n value represents the bump height centered on $\boldsymbol{x}_n$, different for each $\boldsymbol{x}$.

Radial Basis Function Networks. (Cont'd)

If we fix the bump heights to w_n, independent of the test point, then the same bumps centered on the datapoints apply to every test point $\boldsymbol{x}$, and the functional form simplifies.

$$h(\boldsymbol{x}) = \sum_{n=1}^{N} w_n \Phi_n(\boldsymbol{x}) \qquad \Phi_n(\boldsymbol{x}) = \phi\left(\frac{\|x - x_n\|}{r}\right)$$

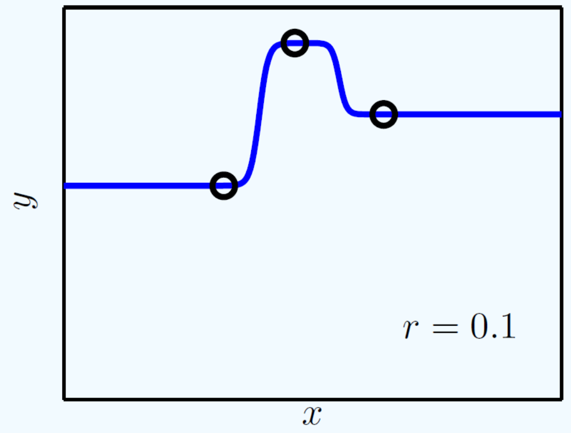
The w_n are parameters that have to be determined

- Now the $h(x)$ determination is a parametric linear problem with augmented variables

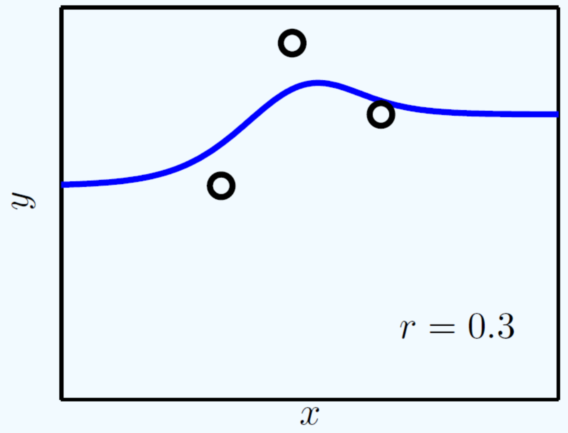
$$h(\boldsymbol{z}) = \sum_{n=1}^{N} w_n z_n \quad with \quad \boldsymbol{z} = \left(\Phi_1(\boldsymbol{x}), \Phi_2(\boldsymbol{x}), \cdots, \Phi_n(\boldsymbol{x})\right)^{\boldsymbol{T}}$$

- These nonlinear transforms are often called *local basis functions*
- Notice that the nonlinear transform is not known ahead of time. It is only known once the data points are seen.

Nonparametric RBF vs Parametric RBF

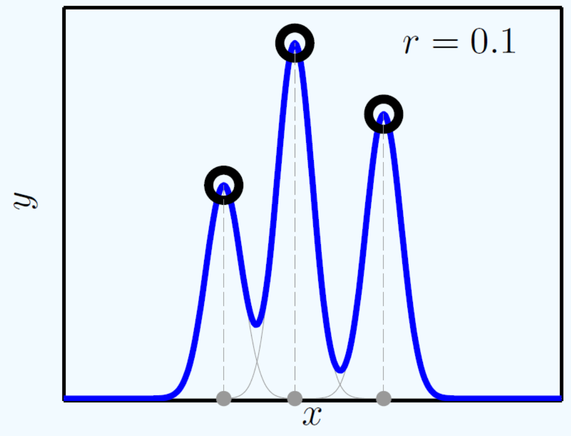


(a) Nonparametric RBF, small r

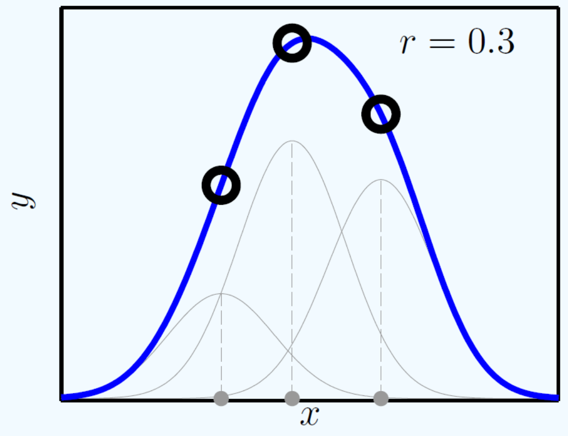


(b) Nonparametric RBF, large r

The Nonparamertric model is step-like with with large |x| behaviour depending on the peripherical data.



(c) Parametric RBF, small r



(b) Parametric RBF, large r

The parametric models is bump-like with large |x| behaviour going to zero

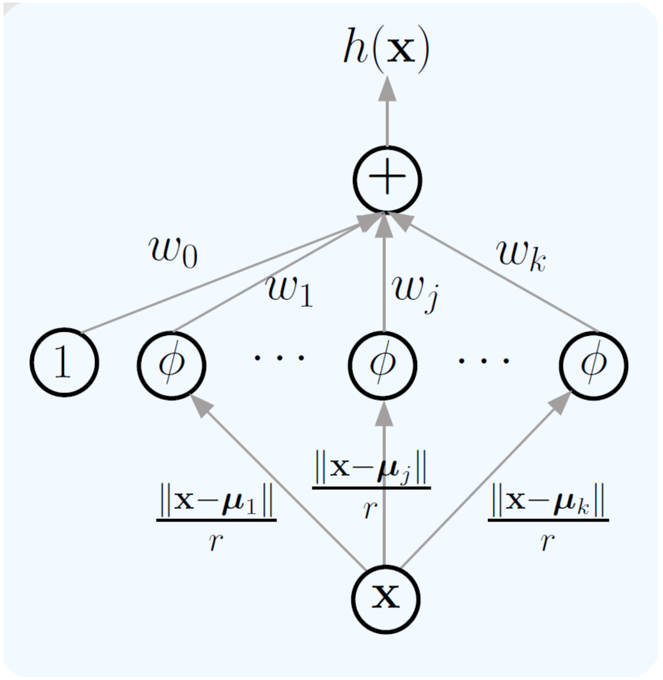
The parametric model has N parameters, $w_1, .., w_N$ ensuring that we always can fit the data. In noisy conditions it will overfit.

How to choose a good model?

Solution: restrict the number of bumps to k ≪ N. If we restrict the number of bumps to k, then only k weights $w_1, \ldots, w_k$ need to be learned.

But, if we restrict the number of weights, its location must be estimated. Denote the centers of the bump by $\mu_1, \cdots, \mu_k$. Then a hypothesis has the parametric form

$$h(x) = w_0 + \sum_{i=1}^{k} w_i \phi\left(\frac{\|x - \mu_i\|}{r}\right) = \mathbf{w}^T \Phi(\boldsymbol{x})$$



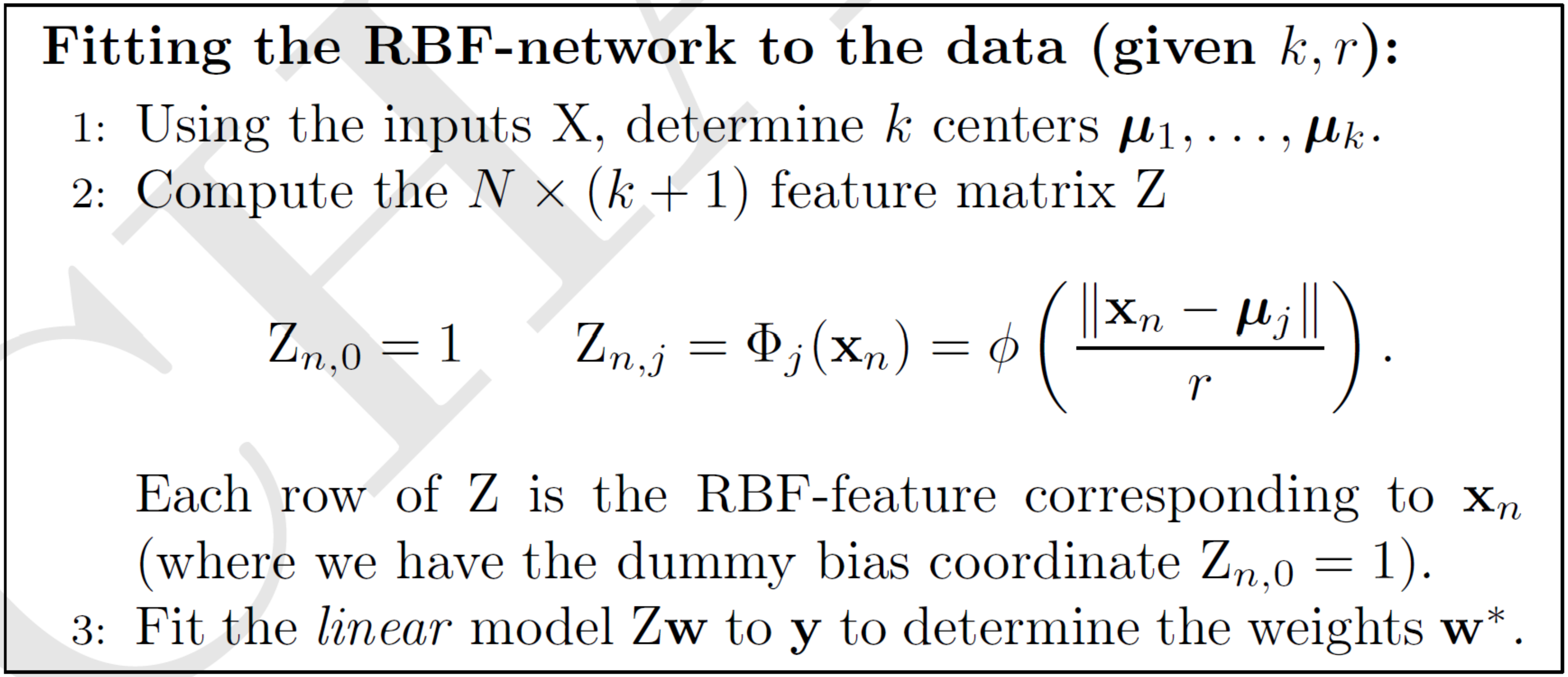
$h(\mathbf{x})$
$+$
w_0
w_1
w_j
w_k
1
ϕ
ϕ
ϕ
$\frac{\|\mathbf{x}-\boldsymbol{\mu}_1\|}{r}$
$\frac{\|\mathbf{x}-\boldsymbol{\mu}_j\|}{r}$
$\frac{\|\mathbf{x}-\boldsymbol{\mu}_k\|}{r}$
$\mathbf{x}$

Now we again introduce a bias term to account for the mean value of y (The same that in regression model)

This model is called the *Radial basis Function Network* (RBF-network).

The model parameter are $w_0, \cdots, w_k, \mu_1, \cdots, \mu_k$, k and r
This model is linear in **w** but nonlinear in **μ**

Fitting the data

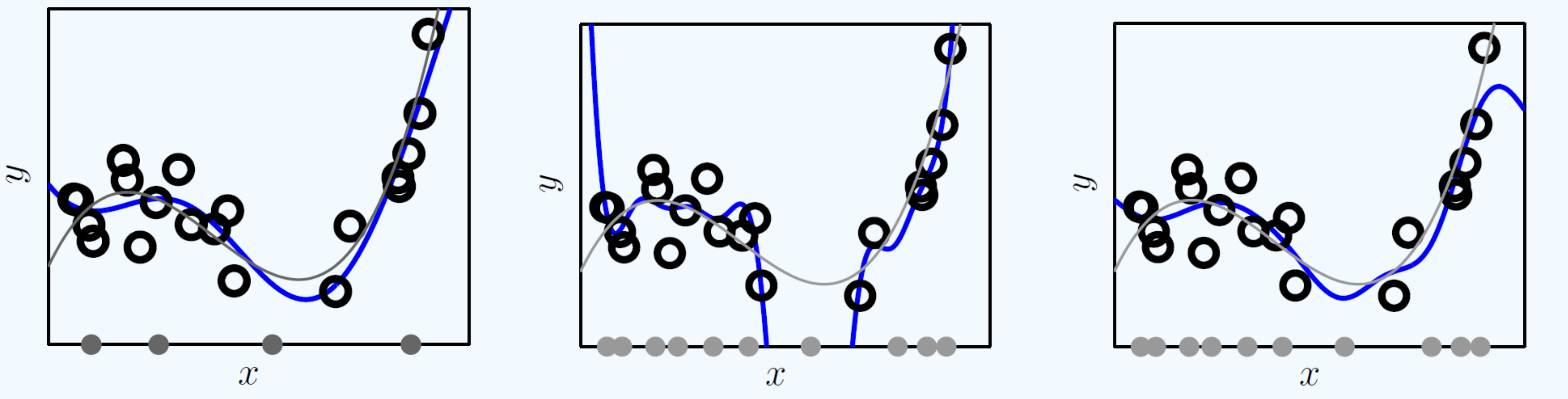


Fitting the RBF-network to the data (given k, r):

```
1: Using the inputs X, determine $k$ centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$.
2: Compute the $N \times (k+1)$ feature matrix Z
       $\mathrm{Z}_{n,0} = 1 \qquad \mathrm{Z}_{n,j} = \Phi_j(\mathbf{x}_n) = \phi\left(\frac{\|\mathbf{x}_n - \boldsymbol{\mu}_j\|}{r}\right).$
   Each row of Z is the RBF-feature corresponding to $\mathbf{x}_n$
   (where we have the dummy bias coordinate $\mathrm{Z}_{n,0} = 1$).
3: Fit the *linear* model $\mathrm{Z}\mathbf{w}$ to $\mathbf{y}$ to determine the weights $\mathbf{w}^*$.
```

- Depending on the specific problem (regression, classification or logistic regression), the step 3 can be solved using a different algorithm as seen in the Linear Models.
- Now most relevant problem is the centers estimation.

Influence of the number of centers



(a) $k = 4$ (b) $k = 10$ (c) $k = 10$, regularized

- This figure shows the influence of the centers location on the fitted function.
- Grey curve is the target function. Blue curve the estimation
- k is the number of centers and r=1/k

- Low number of centers produces a resonable fit.
- High number of centers produces overfitting
- Regularization helps when k is high
- **The center locations were chosen without any specific criteria !**

How choose the centers?

One way to formulate the task is to require that no $\boldsymbol{x}_n$ be far away from a bump center. The $\boldsymbol{x}_n$ should cluster around the centers, with each center μ_j representing one cluster. This is known as the k-center problem, known to be NP-hard

- The centers only depend on the input sample data, not on the labels

This is a unsupervised problem

- The problem remains as, how to partition the data in k clusters ?
- According to the influence of the centers its location it is not important to find very good solutions

So, an iterative algorithm starting from an initial guessing of the centers can achieve sub-optimal solutions.

The K-Means clustering is a good algorithm to choose the centers

Unsupervised k-Means Clustering

The goal of k-means clustering is to partition the input data points $x_1, \ldots, x_N$ into k sets $S_1, \ldots, S_k$ and select centers $\mu_1, \ldots, \mu_k$ for each cluster.

The quality of each cluster is defined by the error: $E_j = \sum_{x_n \in S_j} \left\| x_n - \mu_j \right\|^2$

The k-means error function just sums this cluster error over all clusters,

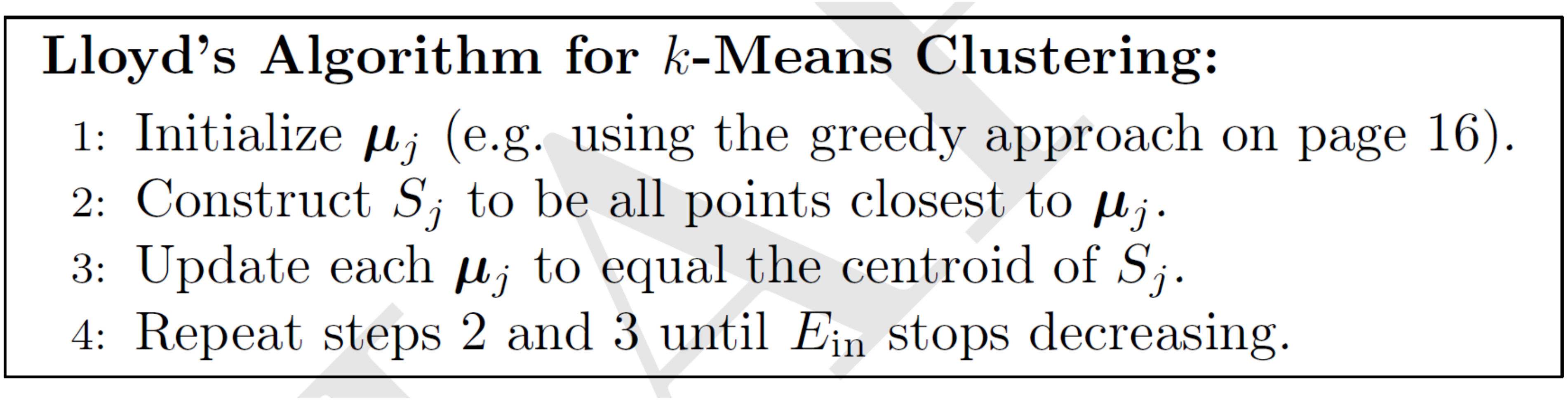
$$E_{in}(S_1, \cdots, S_k; \mu_1, \cdots, \mu_k) = \sum_{j=1}^{k} E_j = \sum_{j=1}^{k} \sum_{x_n \in S_j} \left\| x_n - \mu_j \right\|^2$$

We search for the sets $S_1, \cdots, S_k$ and centers $\mu_1, \cdots, \mu_k$ minimizing this expression

$\boxed{\text{But, minimizing the k-means error is an NP-hard problem}}$

However, fixing the sets or the centers the problem to solve is simple

Lloyd’s Algorithm

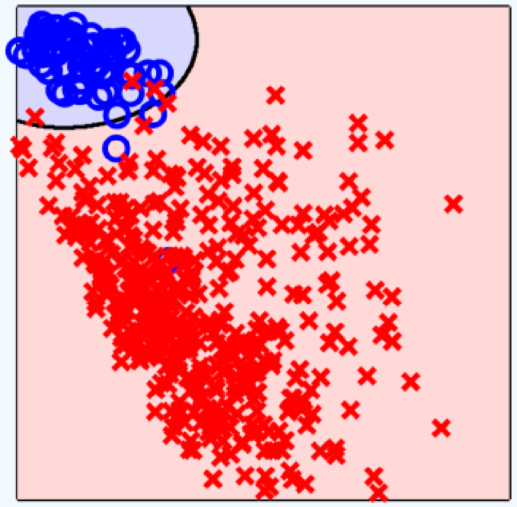


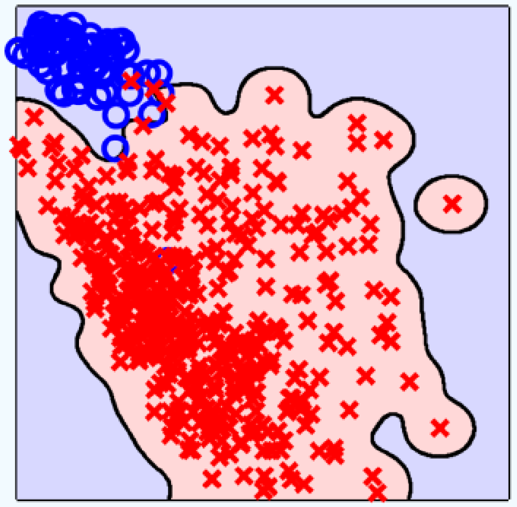
Lloyd’s Algorithm for k-Means Clustering:

```
1: Initialize $\boldsymbol{\mu}_j$ (e.g. using the greedy approach on page 16).
2: Construct $S_j$ to be all points closest to $\boldsymbol{\mu}_j$.
3: Update each $\boldsymbol{\mu}_j$ to equal the centroid of $S_j$.
4: Repeat steps 2 and 3 until $E_{\text{in}}$ stops decreasing.
```

- The algorithm starts with candidate centers and iteratively improves them until convergence.
- Lloyd’s algorithm is not optimal
- Lloyd’s algorithm falls into a class of algorithms known as *E-M* (expectation-maximization) algorithms. It minimizes a complex error function by separating the variables to be optimized into two sets.

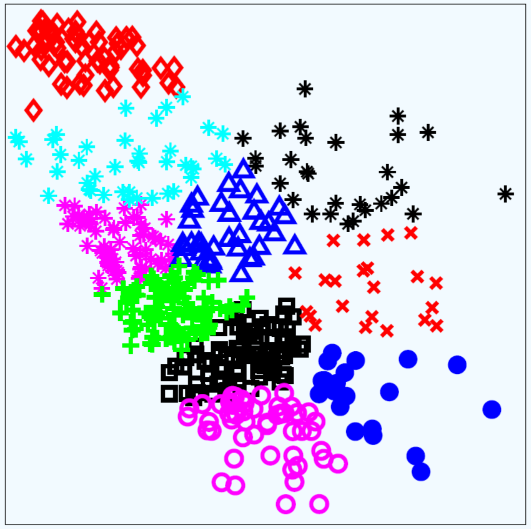
Lloyd’s Algorithm: examples


(a) 10- center RBF-network


(b) 300-center RBF-network

The centers are first obtained by the k-means clustering algorithm.

The resulting classifier is obtained after fitting the weights using the linear regression algorithm for classification



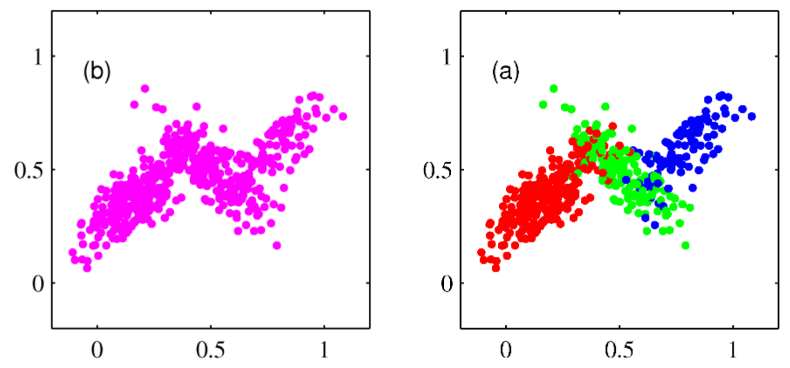
K-means clustering of the 10-classes digit data

It can be compared with the decision regions obtained with 21-NN

Gaussian Mixture Model and E.M.

Motivation

- The GMM model can be seen as a generalization of the K-means, in the sense that each sample belongs to all clusters with a probability.
- In many cases the sample data present a multimodal distribution that can not be fitted by one Gausssian

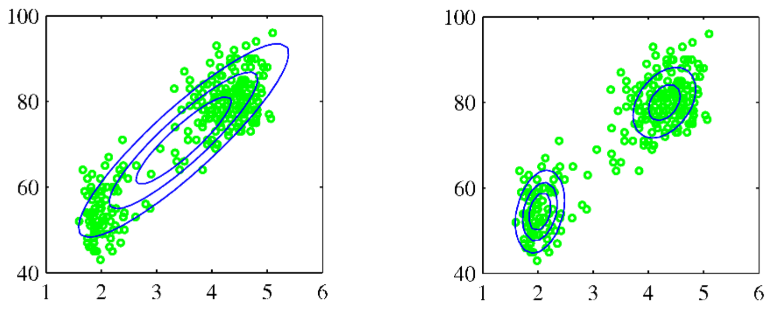


- In this case a more complex multimodal distribution has to be fitted.

- A model based on a linear combination of Gaussians appears to be a good solution to this problem.
- Nevertheles, the parameter estimation problem has to be solved

Mixture of Gaussians-1

- Let us consider a phenomenon which is better modelled by a linear combination of Gaussians



- In this case the model for the sample distribution can be expressed as

$$p(\mathbf{x}) = \sum_{i=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \mathbf{\Sigma}_k)$$

denominated Gaussians Mixture Model (GMM) **(Generative model)**

- Each Gaussian density $\mathcal{N}(x|\mu_k, \mathbf{\Sigma}_k)$ is called a component of the mixture
- The parameters π_k are called *mixing coefficients*

- Integrating $p(\mathbf{x})$ respect to **x,** we obtain $\sum_k \pi_k = 1$
- $p(\mathbf{x}) \geq 0$ implies $\pi_k \geq 0$, then $0 \leq \pi_k \leq 1$

Mixture of Gaussians-2

- From the probability rules the marginal density of the sample can be expressed as

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k) \quad \text{(K is fixed)}$$

  - where $\pi_k = p(k)$ and $\mathcal{N}(x|\mu_k, \mathbf{\Sigma}_k) = p(\mathbf{x}|k)$

- The log of the sample likelihood is given by

$$\log \mathbf{p}(X|\pi, \mu, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}$$

- It can be observed that the optimization of this function on the parameters is complex due to the sum on k inside the logarithm.

- The posterior probability $p(k|\mathbf{x})$, *a.k.a responsabilities*, can be calculated from the Bayes's theorem

$$\gamma_k(\mathbf{x}) \equiv p(k|x) = \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \sigma_k)}{\sum_l \pi_l \mathcal{N}(\mathbf{x}|\mu_l, \sigma_l)}$$

- This distribution is governed by the parameters $\boldsymbol{\pi}=(\pi_1, \ldots, \pi_k)$, $\boldsymbol{\mu}=(\mu_1, \ldots, \mu_k)$ y $\boldsymbol{\Sigma}=(\Sigma_1, \ldots, \Sigma_k)$

Expectation-Minimization(**EM**) for Gaussian Mixtures

- The EM algorithm is an elegant heuristic of finding maximum likelihood solutions for models with latent variables:

- Denote $\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(x|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j,\Sigma_j)}$ (usando teorema de Bayes)

- Lets compute

$$\frac{\partial}{\partial \mu} \log p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = 0$$

$$\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \mathbf{\Sigma}_j)} \Sigma_k(\mathbf{x}_n - \mu_k) = \sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k(\mathbf{x}_n - \mu_k) = 0$$

$$\sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k(\mathbf{x}_n - \mu_k) = 0 \quad \rightarrow \quad \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

EM for Guassian Mixtures - 3

$$\frac{\partial}{\partial \mathbf{\Sigma}_k} \log p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = 0$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T$$

- Finally,

$$\frac{\partial}{\partial \pi_k} \log p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) = 0$$

$$\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \mathbf{\Sigma}_j)} + \lambda = 0$$

$$\pi_k = \frac{N_k}{N}$$

EM for Guassian Mixtures - 4

- Relevant points
  - The solution is not a closed one
    - The responsabilities depend on the parameters in a complex way
- An iteration procedure is suggested
    - We start fixing initial means, covariances and mixing coefficients
    - We alternate between the following two updates
      - E-step: from the parameters we estimate the responsabilities
      - M-step: from the new probabilities we reestimate the parameters
- This iteration guaranteed the increasing of the log-likelihood function

EM algorithm for Gaussian Mixtures

1. Initialize $\{\mu_k, \Sigma_k, \pi_k\}$
2. **E-step**: Evaluate the responsabilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_l \pi_l \mathcal{N}(\mathbf{x}|\mu_l, \Sigma_l)}$$

3. **M-step**: re-estimate the parameters

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

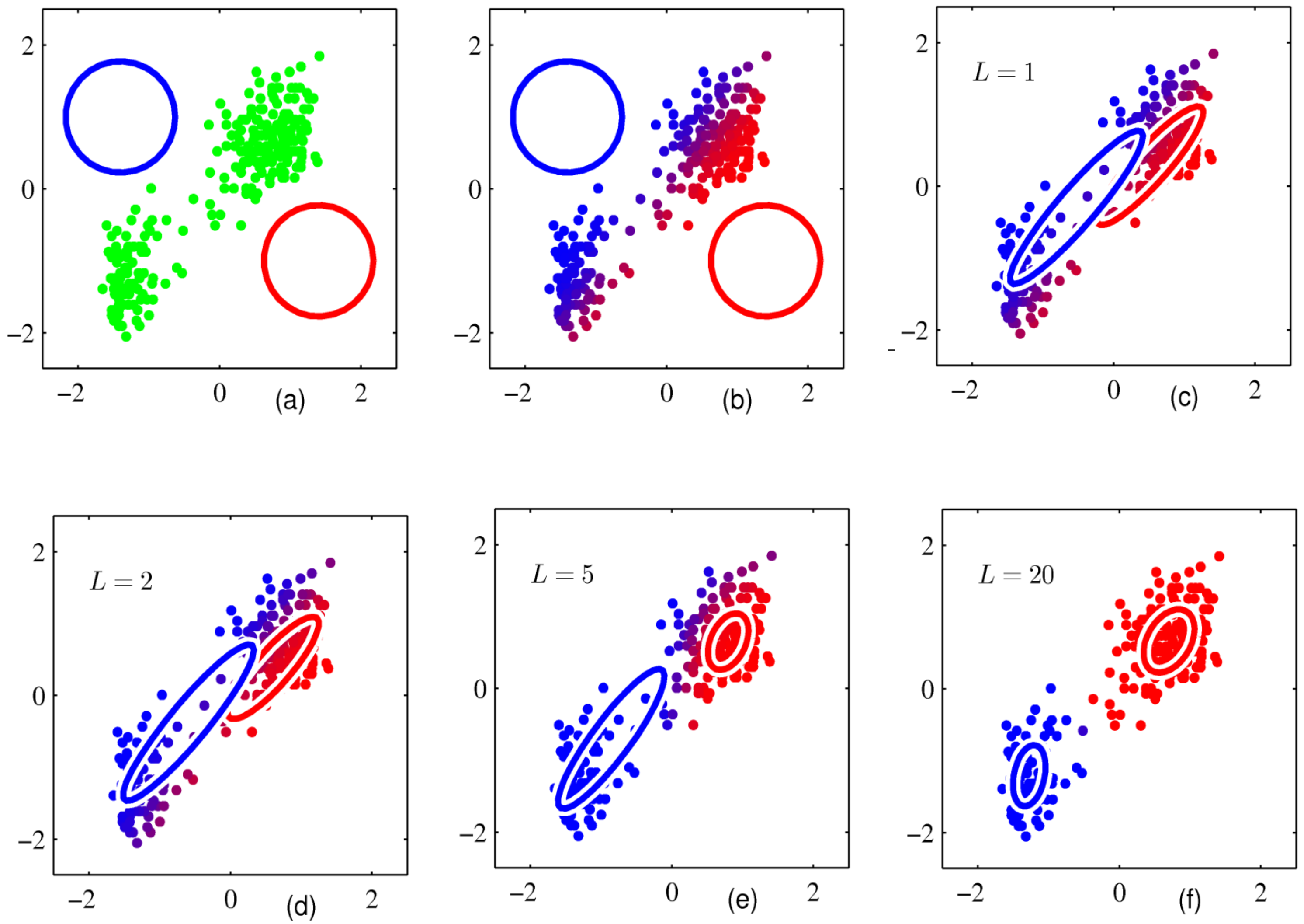
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N} \quad N = \sum_k N_k$$

4. Evaluate the log-likelihhod for convergence

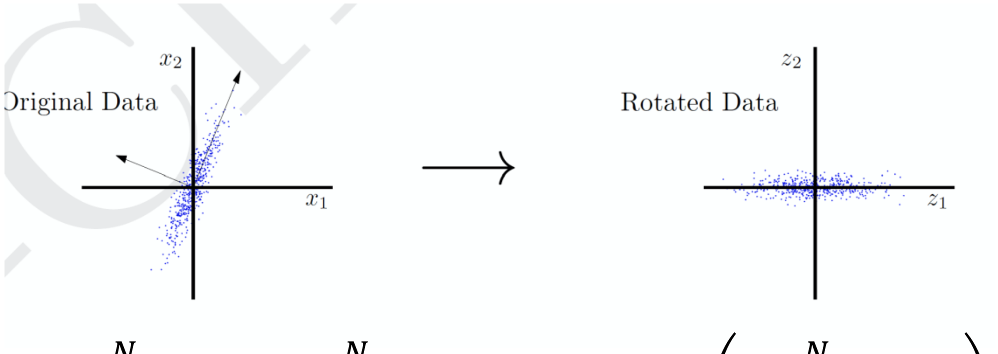
$$\log p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = \sum_{i=1}^{N} \log \left\{ \sum_{i=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right\}$$

Iteration example



Dimension Reduction and Feature Selection: PCA

- PCA attempts to get rid of redundancy to help generalization.
- PCA constructs a small number of linear features to summarizae the input data
  - To do this a rotation of the data is carried out to alingn the axis with the largest fluctuations in the data



$$z_n = \mathbf{v}^T \mathbf{x}_n$$

$$var[\mathbf{z}] = \frac{1}{N}\sum_{n=1}^{N} z_n^2 = \frac{1}{N}\sum_{n=1}^{N} \mathbf{v}^{\mathbf{T}}\mathbf{x}_n\mathbf{x}_n^T\mathbf{v} = \mathbf{v}^{\mathbf{T}}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T\right)\mathbf{v} = \mathbf{v}^{\mathbf{T}}\Sigma\mathbf{v}$$

- So, given a new coordinate system $\{\mathbf{v}_1, \mathbf{v}_{2,\ldots,}\mathbf{v}_d\}$ the projection of a vector **x** in the new system is, with $k \leq d$

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}\mathbf{v}_1 \\ \vdots \\ \mathbf{x}^{\mathrm{T}}\mathbf{v}_{\mathrm{k}} \end{bmatrix} = \Phi(\mathbf{x}) \quad \text{and} \quad \mathbf{x} = \sum_{i=1}^{d} \mathrm{z}_i\mathbf{v}_i$$

Dimension Reduction and Feature Selection: PCA

Using only the first k components, we can reconstruct **x** via: $\hat{\mathbf{x}} = \sum_{i=1}^{k} z_i \mathbf{v}_i$

And the reconstruction error is given by

$$\|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \left\|\sum_{i=k+1}^{d} z_i \mathbf{v}_i\right\|^2 = \sum_{i=k+1}^{d} z_i^2$$

PCA finds a coordinate system that minimizes this total reconstruction error

The first k basis vectors, $\mathrm{v}_1, \mathbf{v}_1, \ldots, \mathbf{v}_k$ of this optimal coordinate basis are called the top-k principal directions.

PCA Algorithm:

```
Inputs: The *centered* data matrix X and $k \geq 1$.
  1: Compute the SVD of X: $[\mathrm{U}, \Gamma, \mathrm{V}] = \mathsf{svd}(\mathrm{X})$.
  2: Let $\mathrm{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ be the first $k$ columns of V.
  3: The PCA-feature matrix and the reconstructed data are
         $\mathrm{Z} = \mathrm{X}\mathrm{V}_k, \qquad \hat{\mathrm{X}} = \mathrm{X}\mathrm{V}_k\mathrm{V}_k^{\mathrm{T}}$.
```