Classification: Nearest Neighbor and Bayes Methods

Morteza H. Chehreghani
morteza.chehreghani@chalmers.se

Chalmers University of Technology

January 30, 2024

Reference

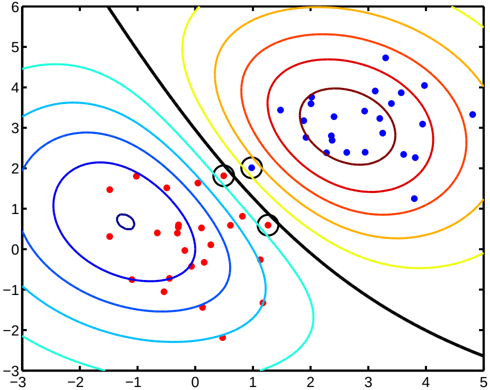
The content and the slides are adapted from

S. Rogers and M. Girolami, A First Course in Machine Learning (FCML), 2nd edition, Chapman & Hall/CRC 2016, ISBN: 9781498738484

Introduction

- Supervised learning
  - Regression → target: any number, cont.
    - Minimised loss (least squares)
    - Maximised likelihood
    - Bayesian approach
  - **Classification** → dis. number/group indices.
- Unsupervised learning
  - Clustering
  - Projection

Classification



- A set of N objects with attributes (usually vector) $\mathbf{x}_n$.
- Each object has an associated response (or label) t_n.
- Binary classification: $t_n = \{0, 1\}$ or $t_n = \{-1, 1\}$,
  - (depends on algorithm).
- Multi-class classification: $t_n = \{1, 2, \ldots, K\}$.

Classification syllabus

- 4 classification algorithms.
- Of which:
  - 2 are probabilistic.
    - Bayes classifier.
    - Logistic regression.
  - 2 are non-probabilistic.
    - K-nearest neighbours.
    - Support Vector Machines.
- There are many others!

Probabilistic vs non-probabilistic classifiers

Classifier is trained on $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and $t_1, \ldots, t_N$ and then used to classify $\mathbf{x}_{\text{new}}$.

- Probabilistic classifiers produce a probability of class membership $P(t_{\text{new}} = k | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$
  - e.g. binary classification: $P(t_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$ and $P(t_{\text{new}} = 0 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$.

Probabilistic vs non-probabilistic classifiers

Classifier is trained on $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and $t_1, \ldots, t_N$ and then used to classify $\mathbf{x}_{\text{new}}$.

- Probabilistic classifiers produce a probability of class membership $P(t_{\text{new}} = k | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$
  - e.g. binary classification: $P(t_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$ and $P(t_{\text{new}} = 0 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$.
- Non-probabilistic classifiers produce a hard assignment
  - e.g. $t_{\text{new}} = 1$ or $t_{\text{new}} = 0$.

Probabilistic vs non-probabilistic classifiers

Classifier is trained on $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and $t_1, \ldots, t_N$ and then used to classify $\mathbf{x}_{\text{new}}$.

- Probabilistic classifiers produce a probability of class membership $P(t_{\text{new}} = k | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$
  - e.g. binary classification: $P(t_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$ and $P(t_{\text{new}} = 0 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$.
- Non-probabilistic classifiers produce a hard assignment
  - e.g. $t_{\text{new}} = 1$ or $t_{\text{new}} = 0$.
- Which to choose depends on application....

Probabilistic vs non-probabilistic classifiers

- Probabilities provide us with more information – $P(t_{\text{new}} = 1) = 0.6$ is more useful than $t_{\text{new}} = 1$.
  - Tells us how **sure** the algorithm is.

Probabilistic vs non-probabilistic classifiers

- Probabilities provide us with more information – $P(t_{\text{new}} = 1) = 0.6$ is more useful than $t_{\text{new}} = 1$.
  - Tells us how **sure** the algorithm is.
- Particularly important where cost of misclassification is high and imbalanced.
  - e.g. Diagnosis: telling a diseased person they are healthy is much worse than telling a healthy person they are diseased.

Probabilistic vs non-probabilistic classifiers

- Probabilities provide us with more information – $P(t_{\text{new}} = 1) = 0.6$ is more useful than $t_{\text{new}} = 1$.
  - Tells us how **sure** the algorithm is.
- Particularly important where cost of misclassification is high and imbalanced.
  - e.g. Diagnosis: telling a diseased person they are healthy is much worse than telling a healthy person they are diseased.
- Extra information (probability) often comes at a cost.
- For large datasets, might have to go with non-probabilistic.

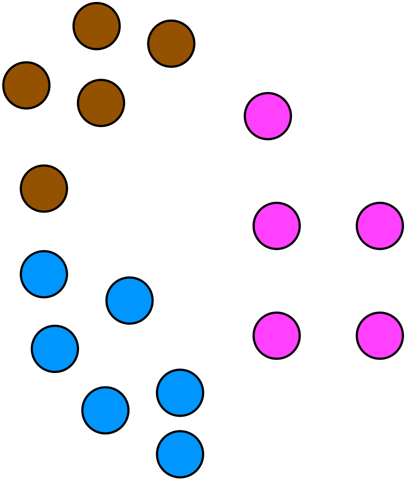
Algorithm 1: K-Nearest Neighbours

- Non-probabilistic.
- Can do binary or multi-class.
- No ‘training’ phase.

Algorithm 1: K-Nearest Neighbours

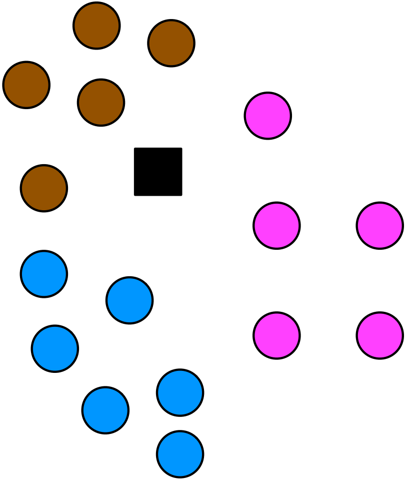
- Non-probabilistic.
- Can do binary or multi-class.
- No 'training' phase.
- How it works:
  - Choose K
  - For a test object $\mathbf{x}_{\text{new}}$:
  - Find the K closest points from the training set.
  - Find majority class of these K neighbours.
  - (Assign randomly in case of a tie)

KNN



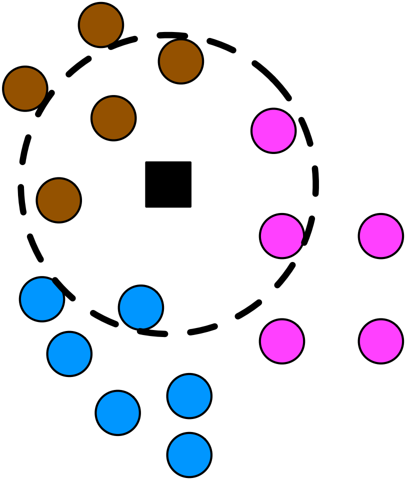
Training data from 3 classes.

KNN



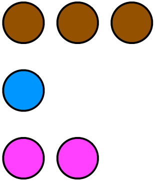
Test point.

KNN



Find $K = 6$ nearest neighbours.

KNN



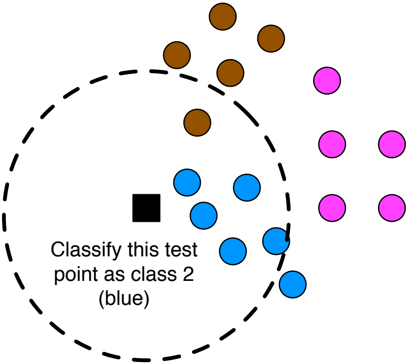
3 from class 1

1 from class 2

2 from class 3

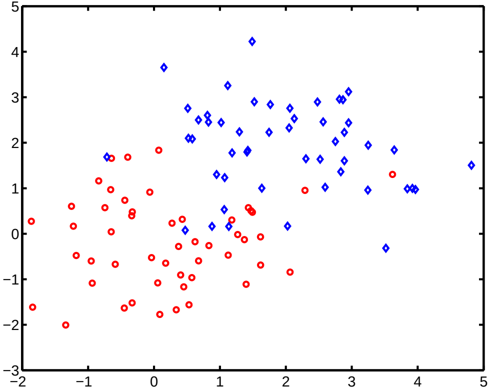
Class one has most votes – classify $\mathbf{x}_{new}$ as belonging to class 1.

KNN



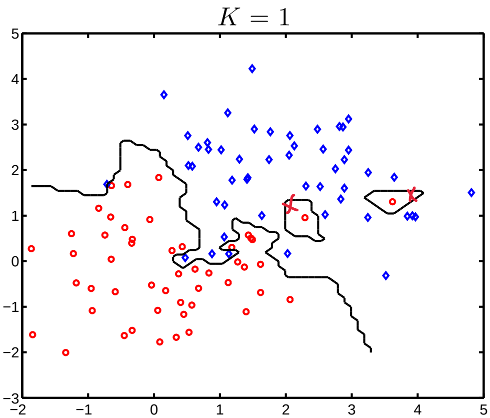
Second example – class 2 has most votes.

KNN – real example



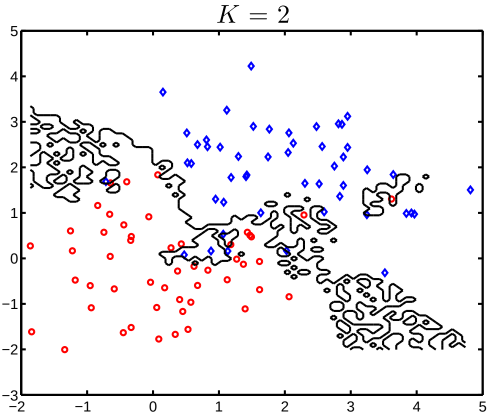
- Binary labels.

KNN – real example



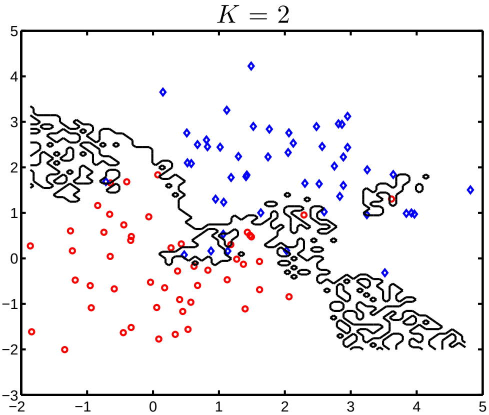
- 1-Nearest Neighbour.
- Line shows decision boundary.
- Too complex – should the islands exist?

KNN – real example



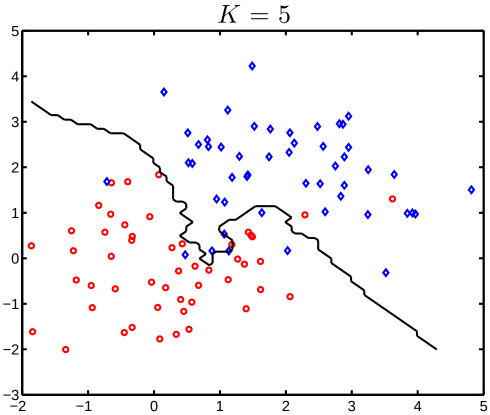
- 2-Nearest Neighbour.
- What's going on?

KNN – real example



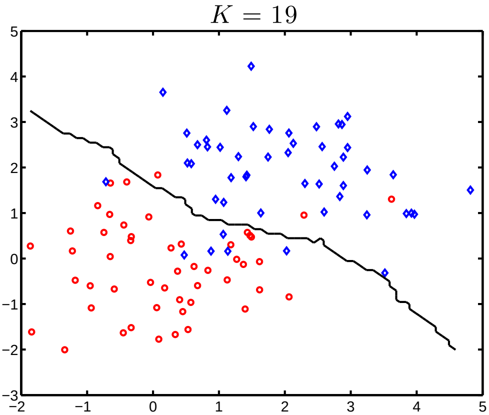
- 2-Nearest Neighbour.
- What's going on?
- Lots of ties – random guessing.

KNN – real example



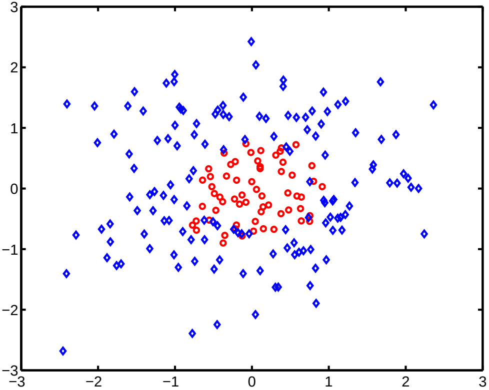
- 5-Nearest Neighbour.
- Much smoother.

KNN – real example



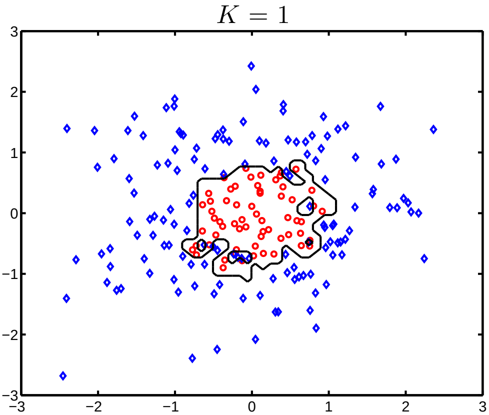
- 19-Nearest Neighbour.
- Very smooth.

KNN – real example 2



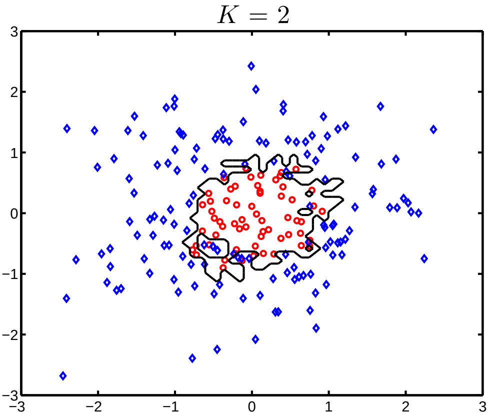
- Binary labels.

KNN – real example 2



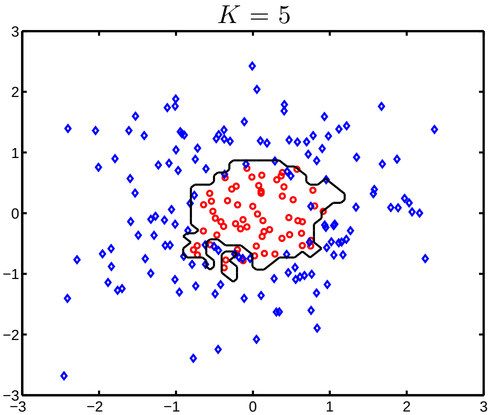
- Non-smooth – too complex again?

KNN – real example 2



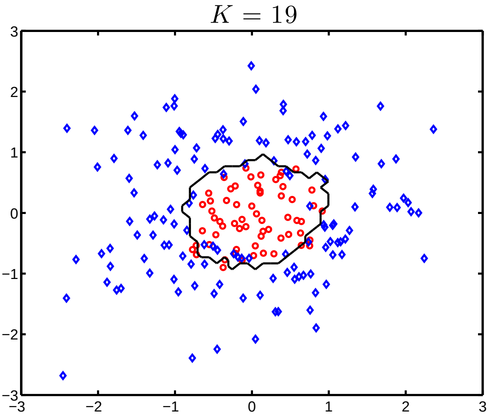
- Random effects again...

KNN – real example 2



- Getting smoother.

KNN – real example 2



- Smoother still.

Problems with KNN

- Class imbalance
  - As K increases, small classes will disappear!
  - Imagine we had only 5 training objects for class 1 and 100 for class 2.
  - For $K \geq 11$, class 2 will **always** win!

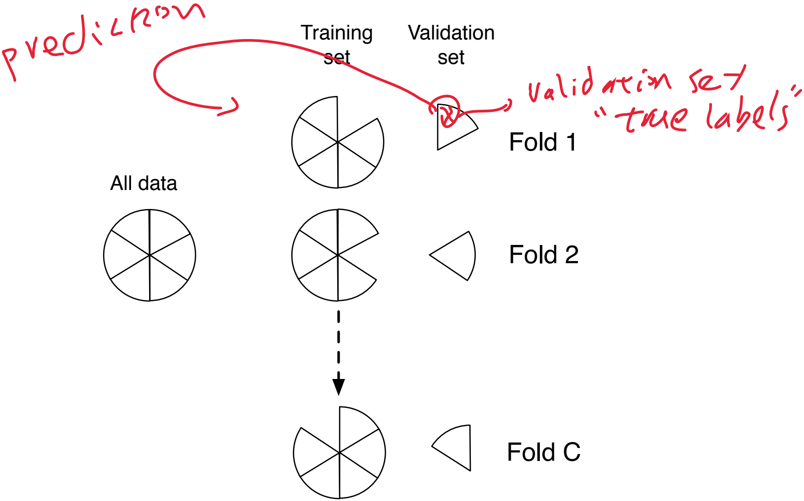
Problems with KNN

- Class imbalance
  - As K increases, small classes will disappear!
  - Imagine we had only 5 training objects for class 1 and 100 for class 2.
  - For $K \geq 11$, class 2 will **always** win!
- How do we choose K?
  - Right value of K will depend on data.
  - Cross-validation!

Cross-validation for classification

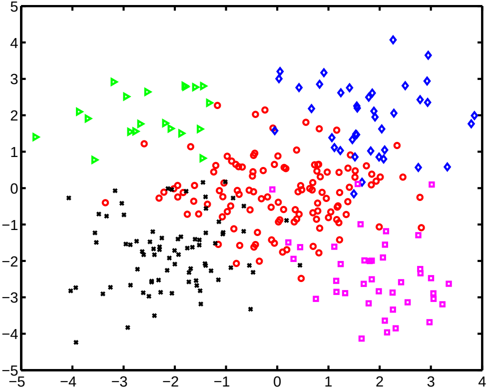
- E.g. to find K in KNN:
- Exactly the same as we have seen before.
- Split the (training) data up – use some to train, some to validation.
- Need a measure of 'goodness'.
- Use number of mis-classifications.....
- ....and use K that minimises it!

Remember...



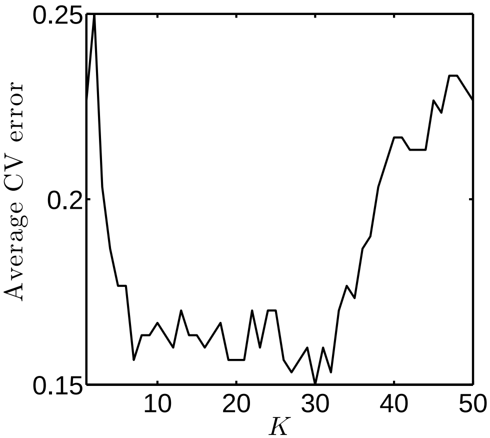
Average number of misclassifications over the C folds.

Example – 5 classes



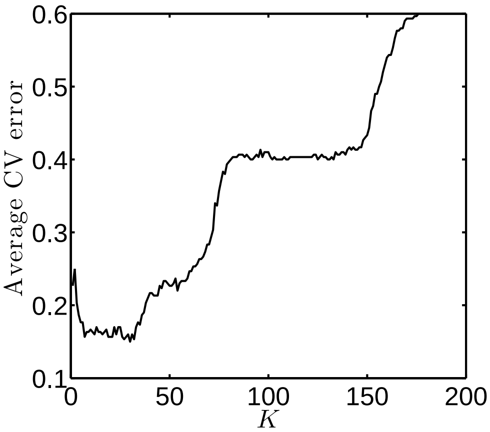
- 5 classes.
- Smallest has 20 instances, biggest 120.

Example – 5 classes



- Curve shows average misclassification error for 10-fold CV.
- Minimum at approximately $K = 30$.

Example – 5 classes



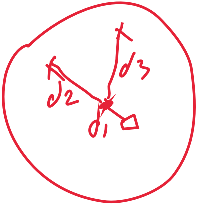
- As K increases, classes 'disappear'
- Causes the 'steps' in error.

KNN – summary

- Non-probabilistic.
- Fast.
- Only one parameter to tune (K).
- Important to tune it well....
- ...can use CV.

KNN – summary

- Non-probabilistic.
- Fast.
- Only one parameter to tune (K).
- Important to tune it well....
- ...can use CV.
- There is a probabilistic version.
  - Not covered in this course.



$$pr(t_{new} = \square) = \frac{d_1}{d_1 + d_2 + d_3}$$

$$Pr(t_{new} = X) = \frac{d_2 + d_3}{d_1 + d_2 + d_3}$$

KNN – summary

- Non-probabilistic.
- Fast.
- Only one parameter to tune (K).
- Important to tune it well....
- ...can use CV.
- There is a probabilistic version.
  - Not covered in this course.
- Now onto a (different) probabilistic classifier...

Bayes classifier

training data

feat. for test data

- Our first probabilistic classifier is based on Bayes rule:

$$P(t_{\text{new}} = k | \underbrace{\mathbf{X}, \mathbf{t}}_{\text{training data}}, \underbrace{\mathbf{x}_{\text{new}}}_{\text{feat. for test data}})$$

$$= \frac{P(\mathbf{x}_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t}) P(t_{\text{new}} = k)}{\underbrace{\sum_j p(\mathbf{x}_{\text{new}} | t_{\text{new}} = j, \mathbf{X}, \mathbf{t}) P(t_{\text{new}} = j)}_{\text{marginal likelihood}}}$$

different values of $j \in \{1, \dots, K\}$

- We need to define a likelihood and a prior and we're done!

marginal likelihood

$$= \sum_j P(x_{new}, t_{new} = j | X, t) = P(x_{new} | X, t)$$

$$\boxed{P(A, B) = P(A) P(B | A)}$$

Bayes classifier – likelihood

$$p(\mathbf{x}_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t})$$

- How likely is $\mathbf{x}_{\text{new}}$ if it is in class k? (not necessarily a probability...)

Bayes classifier – likelihood

$$p(\mathbf{x}_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t})$$

- How likely is $\mathbf{x}_{\text{new}}$ if it is in class k? (not necessarily a probability...)
- We are free to define this *class-conditional distribution* as we like.
- Will depend on type of data.
- e.g.
  - Data are D-dimensional vectors of real values – Gaussian likelihood.
  - Data are number of heads in N coin tosses – Binomial likelihood.

Bayes classifier – likelihood

$$p(\mathbf{x}_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t})$$: e.g. $\mathcal{N}(\mu_k, \Sigma_k)$

- How likely is $\mathbf{x}_{\text{new}}$ if it is in class k? (not necessarily a probability...)
- We are free to define this *class-conditional distribution* as we like.
- Will depend on type of data.
- e.g.
  - Data are D-dimensional vectors of real values – Gaussian likelihood.
  - Data are number of heads in N coin tosses – Binomial likelihood.
- In both cases, training data with $t = k$ used to determine parameters of likelihood for class k (e.g. Gaussian mean and covariance).

Bayes classifier – prior

$$P(t_{\text{new}} = k)$$

- $\mathbf{x}_{\text{new}}$ not present.
- Used to specify prior probabilities for different classes.
- e.g.
  - There are far fewer instances of class 0 than class 1: $P(t_{\text{new}} = 1) > P(t_{\text{new}} = 0)$.
  - No prior preference: $P(t_{\text{new}} = 0) = P(t_{\text{new}} = 1)$.
  - Class 0 is very rare: $P(t_{\text{new}} = 0) \ll P(t_{\text{new}} = 1)$.

Naive-Bayes $\Longrightarrow$ no. of elements to estimate = D

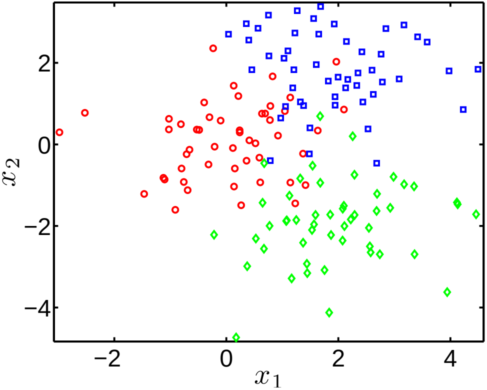
- Naive-Bayes makes the following additional likelihood assumption: Σ_k is diagonal for Gaussian dist.
- The components of $\mathbf{x}_{\text{new}}$ are independent for a particular class:

$$p(\mathbf{x}_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t}) = \underline{\prod_{d=1}^{D} p(x_d^{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t})}$$

- Where D is the number of dimensions and x_d^{new} is the value of the dth one.
- Often used when D is high:
  - Fitting D uni-variate distributions is easier than fitting one D-dimensional one.

elements in $\Sigma_k = D \times D \longrightarrow \dfrac{D \times (D+1)}{2}$

Bayes classifier, example 1



$3 \mathcal{N}(\cdot,\cdot)$

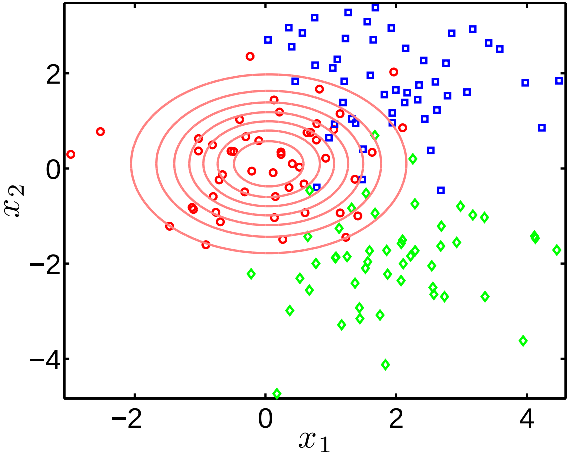
$\left\{ \begin{array}{l} \mathcal{N}(\mu_1, \Sigma_1) \\ \mathcal{N}(\mu_2, \Sigma_2) \\ \mathcal{N}(\mu_3, \Sigma_3) \end{array} \right.$

$\overline{6} + \overline{6} = 12$

- Each object has two attributes: $\mathbf{x} = [x_1, x_2]^\mathsf{T}$.
- $K = 3$ classes.
- We'll use Gaussian class-conditional distributions (with Naive-Bayes assumption).
- $P(t_{\text{new}} = k) = 1/K$ – uniform prior.

1/3

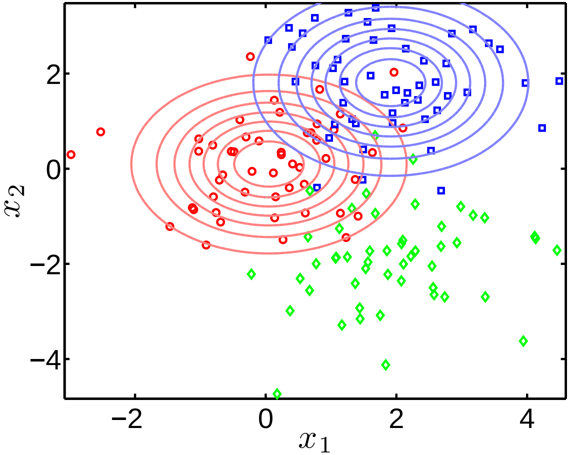
Step 1: fitting the class-conditional densities



$$p(\mathbf{x}|t=k,\mathbf{X},\mathbf{t}) = \prod_{d=1}^{2} \mathcal{N}(\mu_{kd},\sigma_{kd}^2)$$

$$\mu_{kd} = \frac{1}{N_k}\sum_{n:t_n=k} x_{nd} \qquad \sigma_{kd}^2 = \frac{1}{N_k}\sum_{n:t_n=k}(x_{nd}-\mu_{kd})^2$$

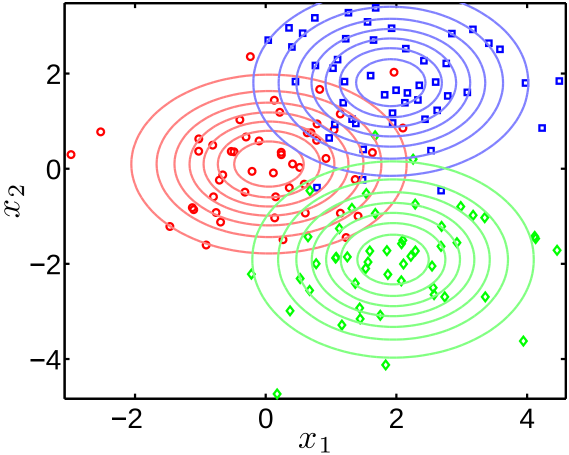
Step 1: fitting the class-conditional densities



$$p(\mathbf{x}|t=k,\mathbf{X},\mathbf{t}) = \prod_{d=1}^{2} \mathcal{N}(\mu_{kd}, \sigma^2_{kd})$$

$$\mu_{kd} = \frac{1}{N_k} \sum_{n:t_n=k} x_{nd} \qquad \sigma^2_{kd} = \frac{1}{N_k} \sum_{n:t_n=k} (x_{nd} - \mu_{kd})^2$$

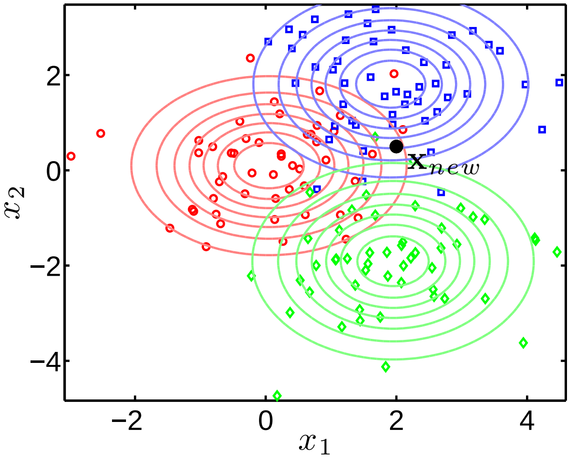
Step 1: fitting the class-conditional densities



$$p(\mathbf{x}|t=k,\mathbf{X},\mathbf{t}) = \prod_{d=1}^{2}\mathcal{N}(\mu_{kd},\sigma_{kd}^2)$$

$$\mu_{kd} = \frac{1}{N_k}\sum_{n:t_n=k} x_{nd} \qquad \sigma_{kd}^2 = \frac{1}{N_k}\sum_{n:t_n=k}(x_{nd}-\mu_{kd})^2$$

Step 2: Evaluate densities at test point

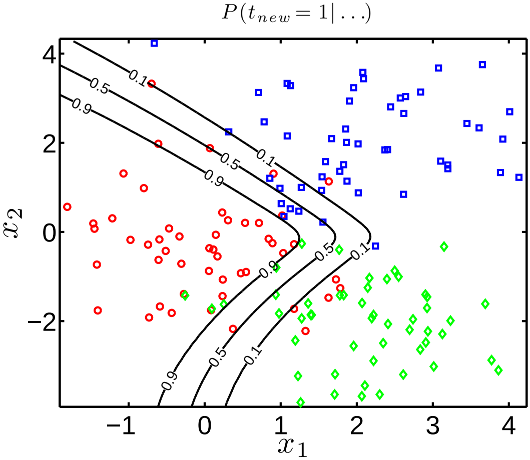


$$p(\mathbf{x}_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t}) = \prod_{d=1}^{D} \mathcal{N}(\mu_{kd}, \sigma_{kd}^2)$$

Compute predictions

- Remember that we assumed $P(t_{\text{new}} = k) = 1/K$.

$$P(t_{\text{new}} = k|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{p(\mathbf{x}_{\text{new}}|t_{\text{new}} = k, \mathbf{X}, \mathbf{t})p(t_{\text{new}} = k)}{\sum_j p(\mathbf{x}_{\text{new}}|t_{\text{new}} = j, \mathbf{X}, \mathbf{t})P(t_{\text{new}} = j)}$$

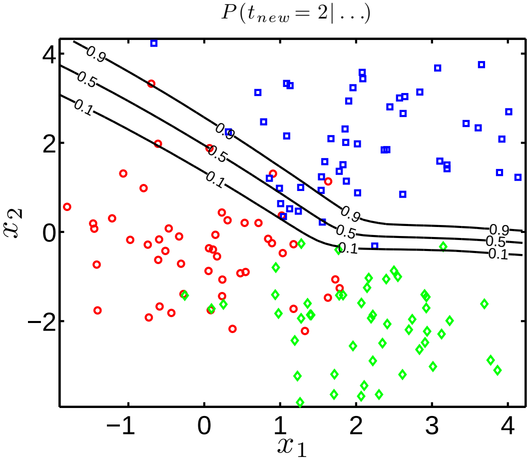


Contours of $P(t_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

Compute predictions

- Remember that we assumed $P(t_{\text{new}} = k) = 1/K$.

$$P(t_{\text{new}} = k|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{p(\mathbf{x}_{\text{new}}|t_{\text{new}} = k, \mathbf{X}, \mathbf{t})p(t_{\text{new}} = k)}{\sum_j p(\mathbf{x}_{\text{new}}|t_{\text{new}} = j, \mathbf{X}, \mathbf{t})P(t_{\text{new}} = j)}$$

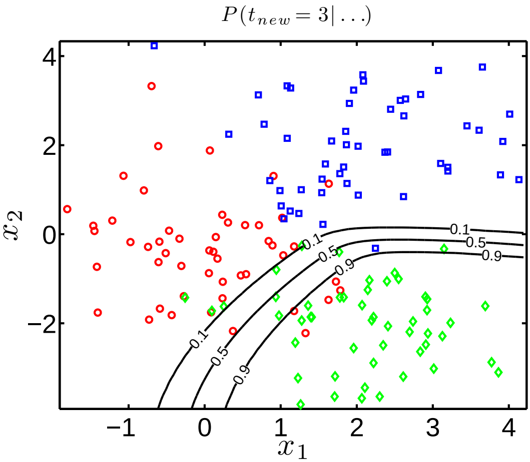


Contours of $P(t_{\text{new}} = 2|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

Compute predictions

- Remember that we assumed $P(t_{\text{new}} = k) = 1/K$.

$$P(t_{\text{new}} = k|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{p(\mathbf{x}_{\text{new}}|t_{\text{new}} = k, \mathbf{X}, \mathbf{t})P(t_{\text{new}} = k)}{\sum_j p(\mathbf{x}_{\text{new}}|t_{\text{new}} = j, \mathbf{X}, \mathbf{t})P(t_{\text{new}} = j)}$$



Contours of $P(t_{\text{new}} = 3|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

Bayes classifier, example 2

- Data are number of heads in 20 tosses (repeated 50 times for each) from one of two coins:
  - Coin 1 ($t_n = 0$): $x_n = 4,\ 7,\ 7,\ 7,\ 4, \ldots$ → r_0
  - Coin 2 ($t_n = 1$): $x_n = 18,\ 16,\ 18,\ 14,\ 17, \ldots$ → r_1
- Use binomial class conditional densities:

$$P(x_n | r_k) = \left(\begin{array}{c} 20 \\ x_n \end{array} \right) r_k^{x_n} (1 - r_k)^{20 - x_n}$$

- Where r_k is the probability that coin k lands heads on any particular toss.
- Problem – predict the coin, t_{new} given a new count, x_{new}.
- (Again assume $P(t_{\text{new}} = k) = 1/K$)

Fit the class conditionals...

- Fitting is just finding r_k:

$$r_k = \frac{1}{20N_k} \sum_{n:t_n=k} x_n$$

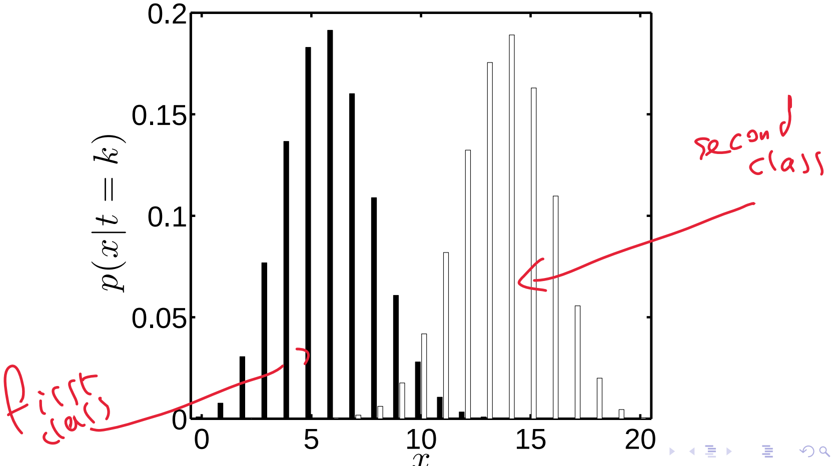
- $r_0 = 0.287$, $r_1 = 0.706$.

Fit the class conditionals...

- Fitting is just finding r_k:

$$r_k = \frac{1}{20N_k} \sum_{n:t_n=k} x_n$$

- $r_0 = 0.287$, $r_1 = 0.706$.



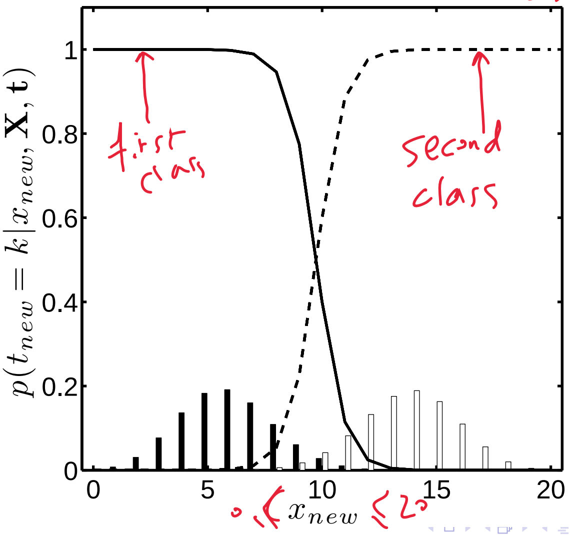
Compute predictions

$$P(t_{\text{new}} = k | x_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{\overbrace{p(x_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t})}^{\text{likelihood}} \overbrace{\cancel{P(t_{\text{new}} = k)}}^{1/2}}{\sum_j p(x_{\text{new}} | t_{\text{new}} = j, \mathbf{X}, \mathbf{t}) \underbrace{\cancel{P(t_{\text{new}} = j)}}_{1/2}}$$

Compute predictions

$$P(t_{\text{new}} = k | x_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{p(x_{\text{new}} | t_{\text{new}} = k, \mathbf{X}, \mathbf{t}) P(t_{\text{new}} = k)}{\sum_j p(x_{\text{new}} | t_{\text{new}} = j, \mathbf{X}, \mathbf{t}) P(t_{\text{new}} = j)}$$

→ for different classes



Bayes classifier – summary

- Decision rule based on Bayes rule.
- Choose and fit class conditional densities.
- Decide on prior.
- Compute predictive probabilities.
- Naive-Bayes:
  - Assume that the dimensions of $\mathbf{x}$ are independent within a particular class.
  - Our Gaussian used the Naive Bayes assumption (could have written $p(\mathbf{x}|t = k, \ldots)$ as product of two independent Gaussians).