Classification

Herman Kamper

2023-03

Classification

From regression to classification:

- Regression: Predict scalar output $y \in \mathbb{R}$ given input $\mathbf{x}$.
- Classification: Predict categorical class label y given input $\mathbf{x}$.

Classification examples:

- Disease diagnoses: Classifying whether a patient is healthy or not.
- Text classification: Classifying documents according to topic.
- Fault diagnoses: Is a photovoltaic system/antenna operating as expected or not?

Representing the target output

Classification: Predict categorical class label y given input $\mathbf{x}$.

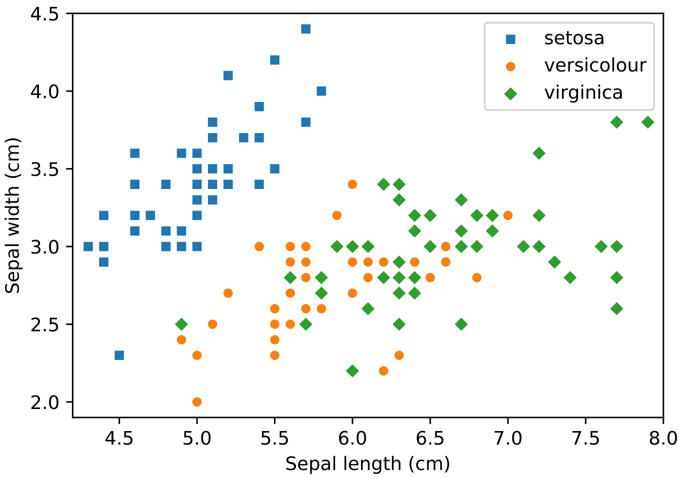
Data: In $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, the label $y^{(n)}$ should tell us which class $\mathbf{x}^{(n)}$ belongs to.

There is a number of ways to encode y numerically.

- Binary classification: $y \in \{0, 1\}$ or $y \in \{-1, 1\}$
- Classification among K classes: $y \in \{1, 2, \ldots, K\}$

Iris flower dataset

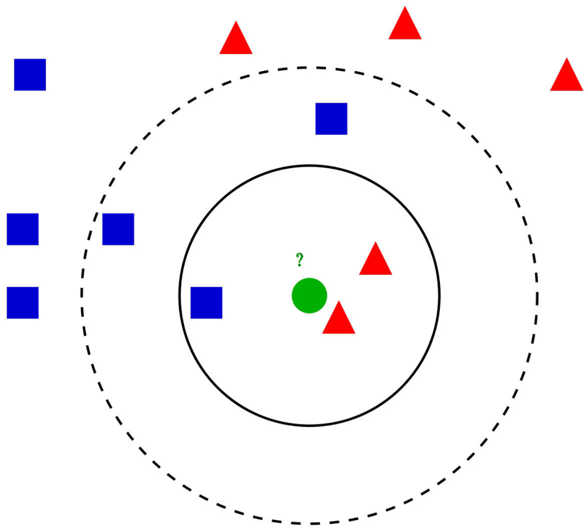
Our running example:[1]



[1]Figure from Wikipedia.

K-nearest neighbours (K-NN)

The entire algorithm in one figure:[2]



[2]Figure from Wikipedia.

K-NN details

The entire algorithm in two bullets

- For a new test input $\mathbf{x}$, identify the K points in the training data closest to $\mathbf{x}$.
- Predict the class of $\mathbf{x}$ as the label that occurs most often in the set $\mathcal{X}_K$ of closest points.

Soft predictions

Can also get "soft" predictions, where the probability of $\mathbf{x}$ belonging to class k is given by:

$$P(y = k|\mathbf{x}) = \frac{1}{K} \sum_{n \in \mathcal{X}_K} \mathbb{I}(y^{(n)} = k)$$

with $\mathbb{I}$ the indicator function and $\mathcal{X}_K$ the set of indices of the nearest neighbours.

Choice of distance function

Euclidean distance:

$$\begin{aligned}
&d_{\text{euclid}}\left(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}\right) \\
&\quad = \sqrt{\left(x_1^{(a)} - x_1^{(b)}\right)^2 + \left(x_2^{(a)} - x_2^{(b)}\right)^2 + \ldots + \left(x_D^{(a)} - x_D^{(b)}\right)^2} \\
&\quad = \left\|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\right\|
\end{aligned}$$

Cosine distance: θ is the angle between $\mathbf{x}^{(a)}$ and $\mathbf{x}^{(b)}$.

$$\begin{aligned}
d_{\cos}\left(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}\right) &= 1 - \cos\theta \\
&= 1 - \frac{\mathbf{x}^{(a)} \cdot \mathbf{x}^{(b)}}{\|\mathbf{x}^{(a)}\| \, \|\mathbf{x}^{(b)}\|}
\end{aligned}$$

K-NN in practice

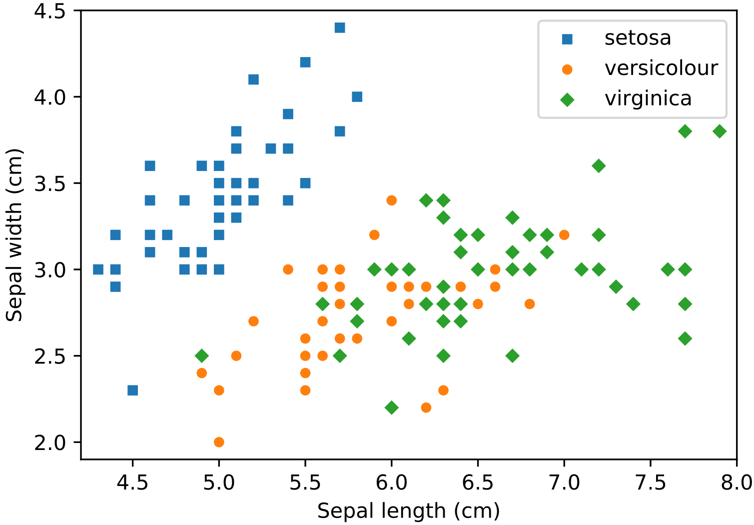
Problems with K-NN

- Computational complexity: To classify one point, we need to run through entire dataset (issues when $N \gg$).
- Distance functions can be inaccurate (need to make some assumptions).
- Curse of dimensionality (issues when $D \gg$): Everything seems far away.

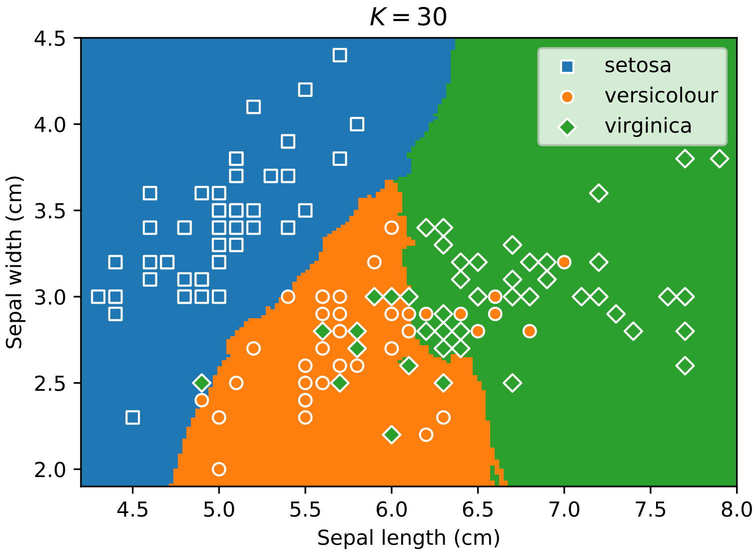
Terminology

- K-NN is a *non-parametric* classification approach.
- It is an example of *memory-based* or *instance-based* learning.

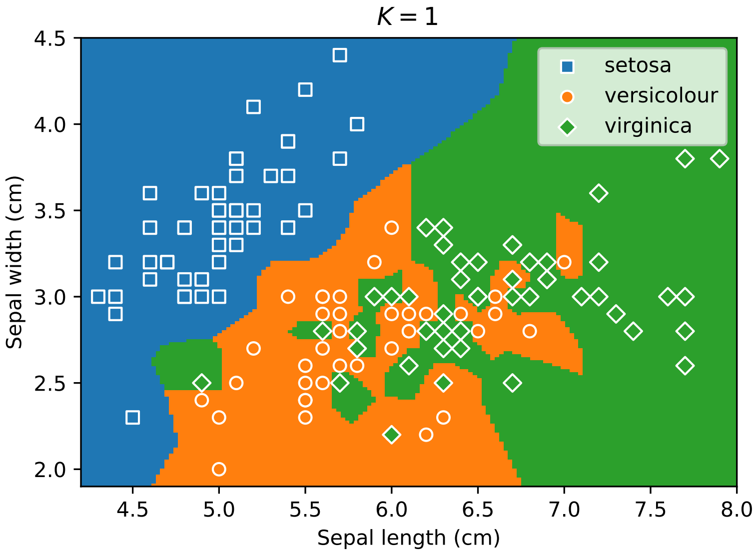
K-NN on Iris dataset



K-NN on Iris dataset



K-NN on Iris dataset



K-NN for voice conversion

Example from the LSL group at Stellenbosch University.

Bayes classifier

If we wanted to follow a probabilistic approach, we could use the following prediction model:

$$f(\mathbf{x};\boldsymbol{\theta}) = \arg\max_k P(y=k|\mathbf{x})$$

To use this model, we need to know $P(y=k|\mathbf{x})$. We can use Bayes' rule:

$$P(y=k|\mathbf{x}) = \frac{p(\mathbf{x}|y=k)P(y=k)}{p(\mathbf{x})}$$

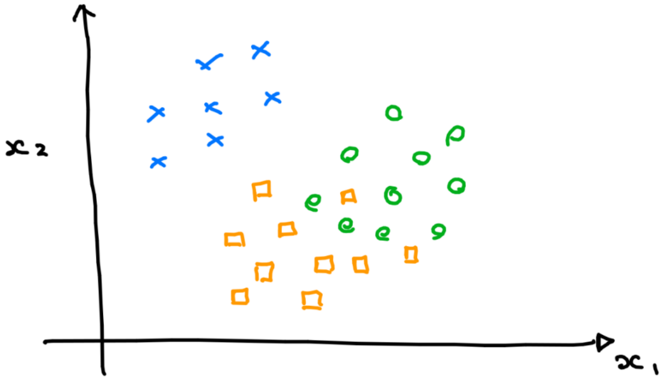
Since $p(\mathbf{x})$ is the same for all k and we are only interested in the max, we can throw away the denominator:

$$P(y=k|\mathbf{x}) \propto p(\mathbf{x}|y=k)P(y=k)$$

This equation is very general. To actually use it, we need to decide on forms for $p(\mathbf{x}|y=k)$ and $P(y=k)$ and then figure out how we will learn their parameters $\boldsymbol{\theta}$ from training data $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$.

Intuitively: What do we need for the Bayes classifier?

$$P(y = k|\mathbf{x}) \propto p(\mathbf{x}|y = k)P(y = k)$$



Quadratic and linear discriminant analysis

For $P(y = k) = \pi_k$, a common approach is to simply count the number of training points assigned to class k:

$$\hat{\pi}_k = \frac{\sum_{n=1}^{N} \mathbb{I}(y^{(n)} = k)}{N}$$

We could decide that for each class we use

$$p(\mathbf{x}|y = k; \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

and then set $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ to the MLE for each class. This is called *quadratic discriminant analysis* (QDA).

This could be problematic, though. If the dimensionality D is high and we have few training points N, there might not be enough data to estimate $\{(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^{K}$. E.g. each $\boldsymbol{\Sigma}_k$ is a $D \times D$ matrix, so there can be many parameters!

We could make the assumption that all classes share the same covariance matrix $\boldsymbol{\Sigma}$ and then only fit $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$, giving us more data to fit the single $\boldsymbol{\Sigma}$. This is called *linear discriminant analysis* (LDA).

The *naive Bayes* assumption goes even further!

Naive Bayes

In naive Bayes we assume that each feature is independent, i.e. that each dimension of $\mathbf{x}$ is independent:

$$p(\mathbf{x}|y = k; \boldsymbol{\theta}) = \prod_{d=1}^{D} p(x_d|y = k; \boldsymbol{\theta})$$

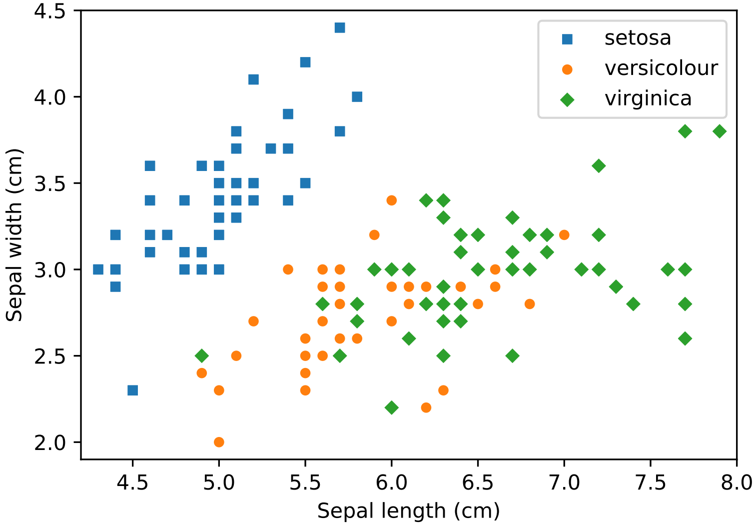
The naive Bayes assumption can be made for any distribution. For the Gaussian case specifically, it leads to

$$p(\mathbf{x}|y = k; \boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(x_d; \mu_{k,d}, \sigma_{k,d}^2)$$

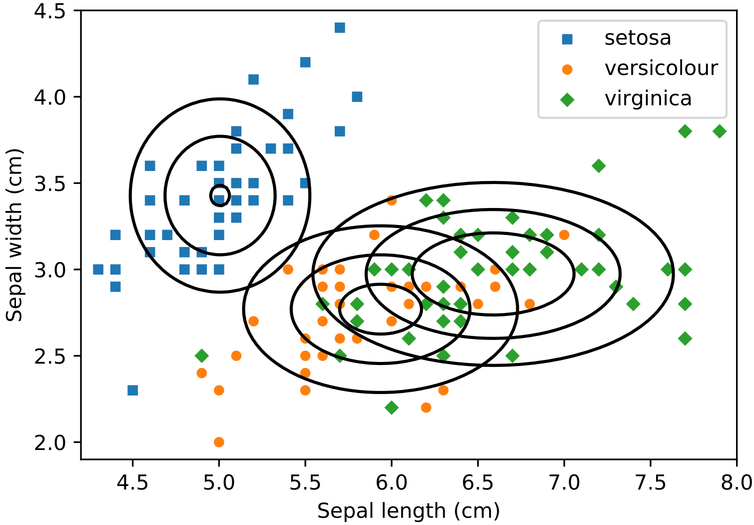
where the set of parameters $\boldsymbol{\theta}$ are all the means and variances.

This can easily be fit using the MLE for each of the D univariate Gaussians for each of the K classes, i.e. we will have to fit $D \cdot K$ univariate Gaussians.

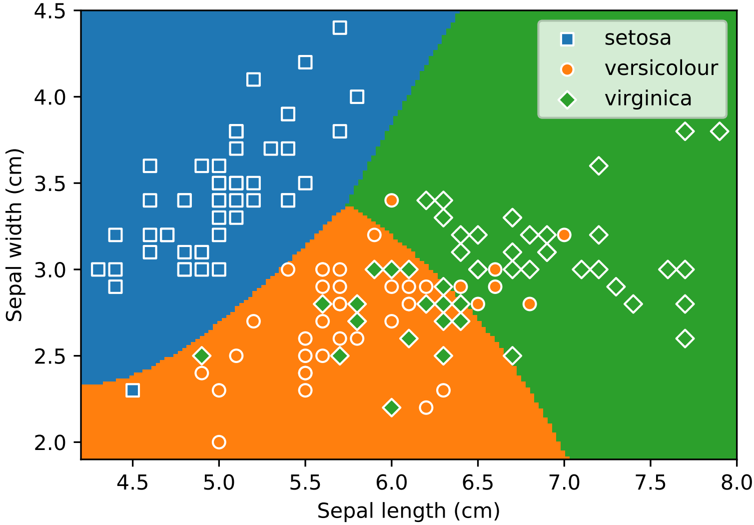
Iris dataset



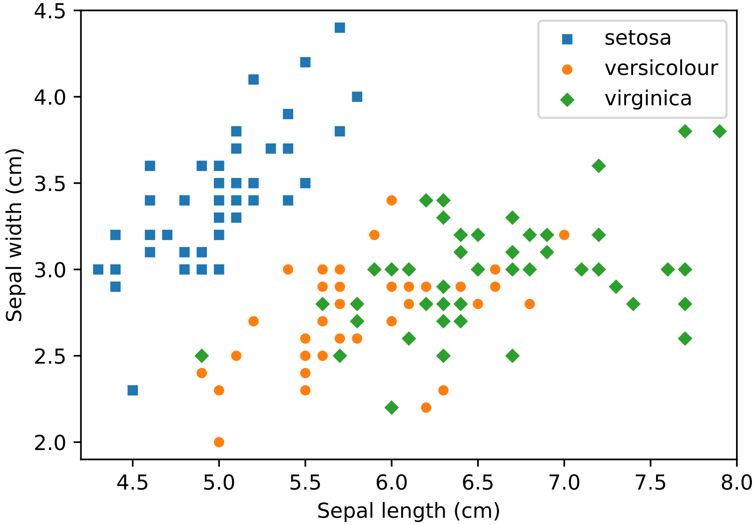
Gaussian Naive Bayes



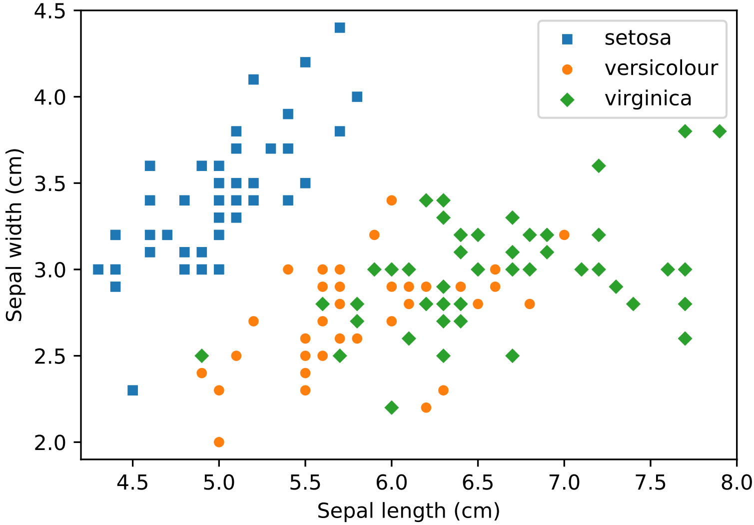
Gaussian Naive Bayes



Quadratic discriminant analysis



Linear discriminant analysis



Generative vs discriminative classification

Generative models

- Bayes classifier: $P(y = k|\mathbf{x}) \propto p(\mathbf{x}|y = k)P(y = k)$
- Choose forms for $p(\mathbf{x}|y = k)$ and $P(y = k)$ and learn the parameters from data.
- Referred to as *generative modelling* since we can generate data:
  - First sample class from $P(y)$.
  - Then sample data from $p(\mathbf{x}|y = \text{sampled class})$.
- But often we aren't actually interested in generating data: We just want to classify!
- And it might be tricky to model $p(\mathbf{x}|y = k)$ for each class.

Discriminative models

- Just model $P(y = k|\mathbf{x})$ directly!
- Use training data $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ to directly fit the probability we are actually interested in.
- Logistic regression (next) is an example of discriminative modelling.

Videos covered in this note

- Classification 1: Task (9 min)
- Classification 2: K-nearest neighbours (15 min)
- Classification 3: Bayes classifier and naive Bayes (17 min)
- Classification 4: Generative vs discriminative (8 min)

Reading

- ISLR 2.2.3
- ISLR 4.1 intro
- ISLR 4.4 intro
- ISLR 4.4.4