Foundations of Machine Learning
CentraleSupélec Paris — Fall 2017

8. Nearest neighbors

Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Practical matters

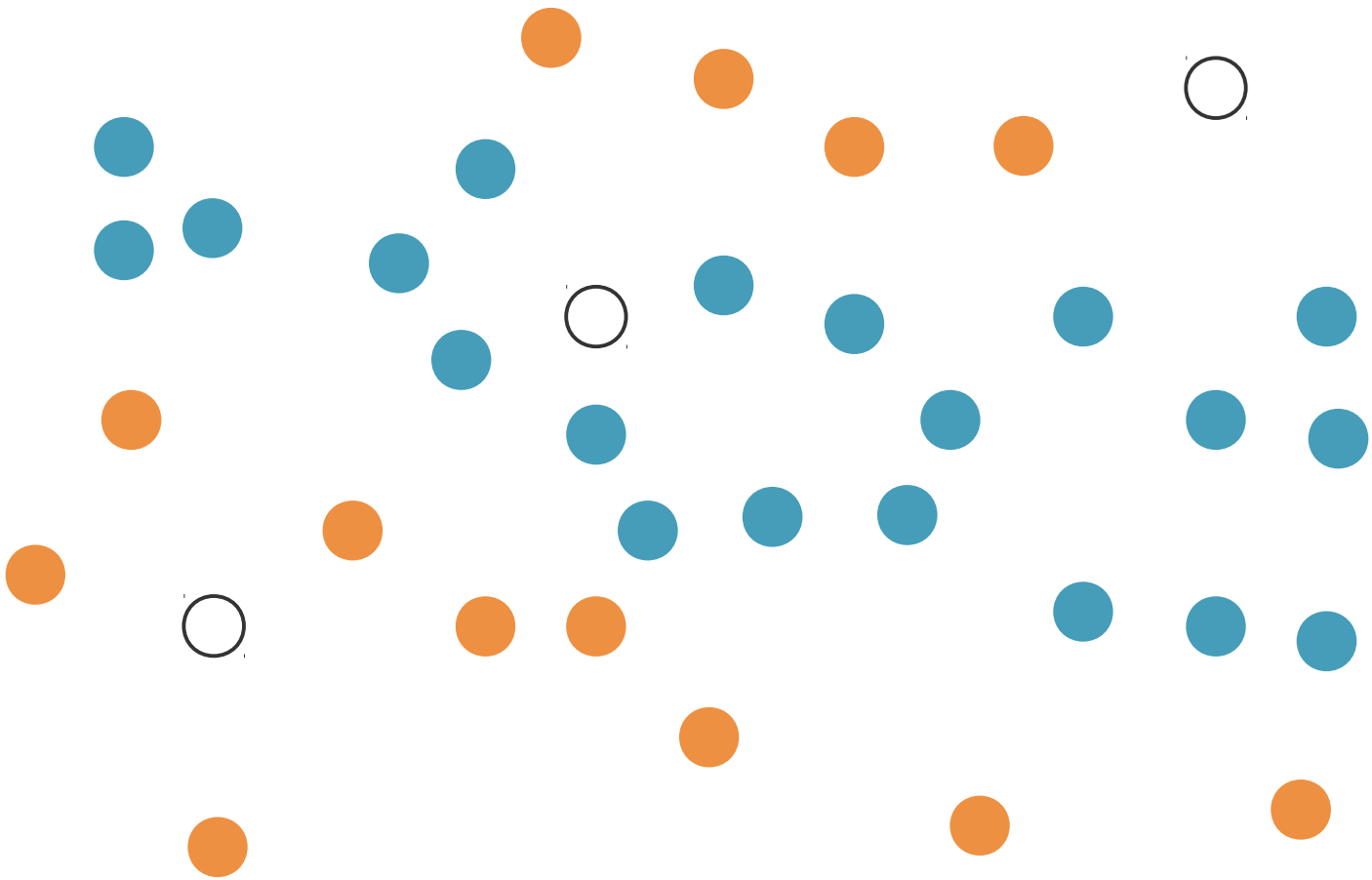
- **Class representatives**
  - William PALMER william.palmer@student.ecp.fr
  - Léonard BOUSSIOUX leonard.boussioux@student.ecp.fr
- **Kaggle project**

Learning objectives

- Implement the **nearest-neighbor** and **k-nearest-neighbors** algorithms.
- Compute **distances** between **real-valued vectors** as well as objects represented by **categorical features**.
- Define the **decision boundary** of the nearest-neighbor algorithm.
- Explain why kNN might not work well in **high dimension**.

Nearest neighbors

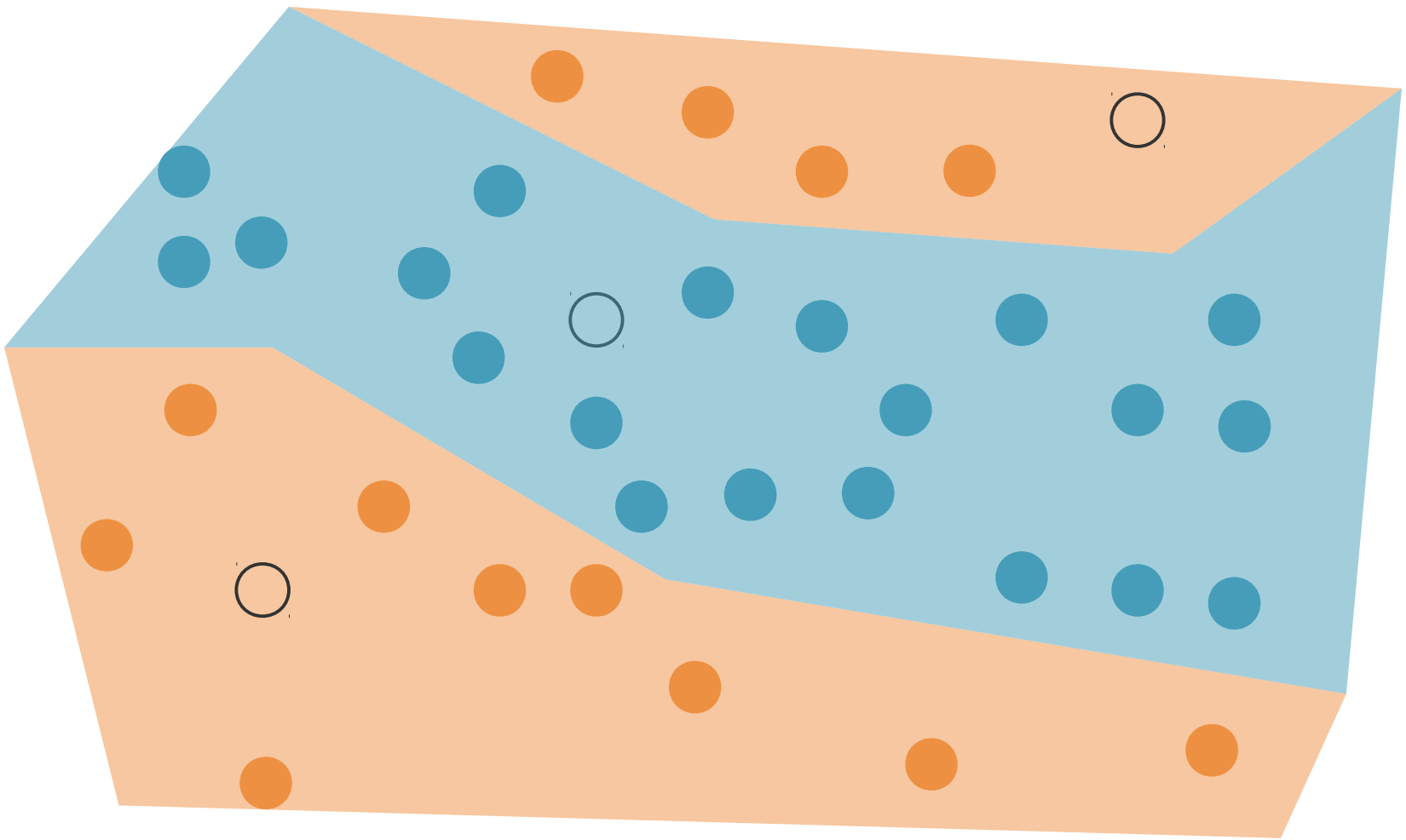
- **How would you color the blank circles?**



- **How would you color the blank circles?**

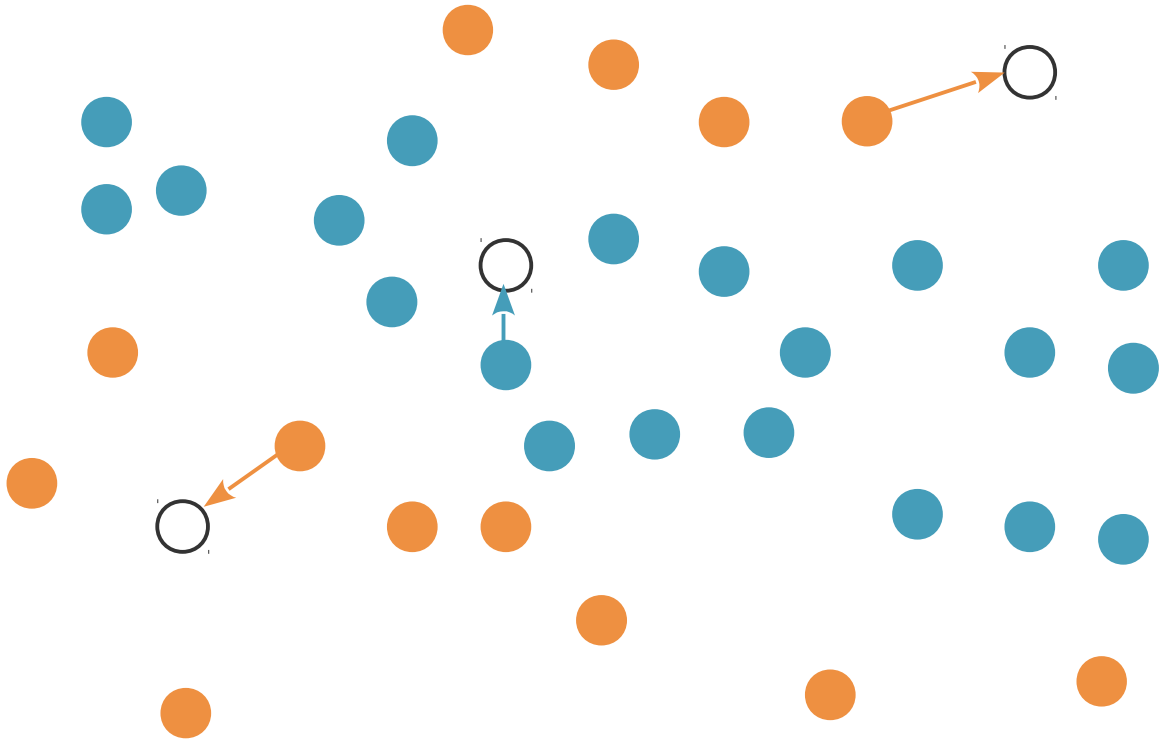
Partitioning the space

The training data partitions the entire space



Nearest neighbor

- **Learning:**
  - Store all the training examples
- **Prediction**:
  - For **x**: the label of the training example closest to it



k nearest neighbors

- **Learning:**
  - Store all the training examples
- **Prediction:**
  - Find the k training examples closest to **x**
  - **Classification?**

k nearest neighbors

- **Learning:**
  - Store all the training examples
- **Prediction:**
  - Find the k training examples closest to **x**
  - **Classification**

    Majority vote: Predict the class of the most frequent label among the k neighbors.

k nearest neighbors

- **Learning:**
  - Store all the training examples
- **Prediction:**
  - Find the k training examples closest to **x**
  - **Classification**

    Majority vote: Predict the class of the most frequent label among the k neighbors.
  - **Regression?**

k nearest neighbors

- **Learning:**
  - Store all the training examples
- **Prediction:**
  - Find the k training examples closest to **x**
  - **Classification**

    Majority vote: Predict the class of the most frequent label among the k neighbors.
  - **Regression**

    Predict the average of the labels of the k neighbors.

Choice of k

- **Small k:** noisy

  The idea behind using more than 1 neighbor is to average out the noise

- **Large k:** computationally intensive

  If k = n

Choice of k

- **Small k:** noisy

  The idea behind using more than 1 neighbor is to average out the noise

- **Large k:** computationally intensive

  If k=n, then we predict

  – for classification: the majority class

  – for regression: the average value

- Set k by **cross-validation**

- Heuristic: **$k \approx \sqrt{n}$**

Non-parametric learning

Non-parametric learning algorithm:

- **the complexity of the decision function grows with the number of data points.**
- contrast with linear regression (≈ as many parameters as features).
- Usually: decision function is expressed directly in terms of the training examples.
- Examples:
  - kNN (this chapter)
  - tree-based methods (Chap. 9)
  - SVM (Chap. 10)

Instance-based learning

- **Learning:**
  - Storing training instances.
- **Predicting:**
  - Compute the label for a new instance based on its **similarity** with the stored instances.

- Also called **lazy learning.**
- Similar to **case-based reasoning**
  - Doctors treating a patient based on how patients with similar symptoms were treated,
  - Judges ruling court cases based on legal precedent.

Instance-based learning

- **Learning:**
  - Storing training instances.
- **Predicting:**
  - Compute the label for a new instance based on its **similarity** with the stored instances.

where the magic happens!

- Also called **lazy learning.**
- Similar to **case-based reasoning**
  - Doctors treating a patient based on how patients with similar symptoms were treated,
  - Judges ruling court cases based on legal precedent.

Computing distances & similarities

Distances between instances

- **Distance**

$$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$$

Distances between instances

- **Distance**

$$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$$

1. $d(\boldsymbol{x}, \boldsymbol{x}) = 0$
2. $d(\boldsymbol{x}, \boldsymbol{z}) = d(\boldsymbol{z}, \boldsymbol{x})$
3. $d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{u}) + d(\boldsymbol{u}, \boldsymbol{x})$

Distances between instances

$\boldsymbol{x} \in \mathbb{R}^p$

- **Euclidean distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_2 = \sqrt{\sum_{j=1}^{p} \left(x_j^1 - x_j^2\right)^2}$$

Distances between instances

$\boldsymbol{x} \in \mathbb{R}^p$

- **Euclidean distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_2 = \sqrt{\sum_{j=1}^{p} \left(x_j^1 - x_j^2\right)^2}$$

- **Manhattan distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_1 = \sum_{j=1}^{p} |x_j^1 - x_j^2|$$

Why is this called the Manhattan distance?

Distances between instances

$\boldsymbol{x} \in \mathbb{R}^p$

- **Euclidean distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_2 = \sqrt{\sum_{j=1}^{p} \left(x_j^1 - x_j^2\right)^2}$$

- **Manhattan distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_1 = \sum_{j=1}^{p} |x_j^1 - x_j^2|$$

- **Lq-norm: Minkowski distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_q = \left(\sum_{j=1}^{p} |x_j^1 - x_j^2|^q\right)^{1/q}$$

  - L1 = Manhattan.
  - L2 = Euclidean.
  - L_∞ ?

Distances between instances

$\boldsymbol{x} \in \mathbb{R}^p$

- **Euclidean distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_2 = \sqrt{\sum_{j=1}^{p} \left(x_j^1 - x_j^2\right)^2}$$

- **Manhattan distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_1 = \sum_{j=1}^{p} |x_j^1 - x_j^2|$$

- **Lq-norm: Minkowski distance**

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = ||\boldsymbol{x}^1 - \boldsymbol{x}^2||_q = \left(\sum_{j=1}^{p} |x_j^1 - x_j^2|^q\right)^{1/q}$$

  - L1 = Manhattan.
  - L2 = Euclidean.
  - L_∞

$$L_\infty = \max_j(|x_j^1 - x_j^2|)$$

Similarity between instances

$$s = \frac{1}{1+d}$$

- **Pearson's correlation**

$$\rho(\boldsymbol{x}, \boldsymbol{z}) = \frac{\sum_{j=1}^{p} (x_j - \bar{x})(z_j - \bar{z})}{\sqrt{\sum_{j=1}^{p} (x_j - \bar{x})^2} \sqrt{\sum_{j=1}^{p} (z_j - \bar{z})^2}}$$

$$\bar{x} = \frac{1}{p} \sum_{j=1}^{p} x_j$$

- Assuming the **data is centered**

$$\rho(\boldsymbol{x}, \boldsymbol{z}) = \frac{\sum_{j=1}^{p} x_j z_j}{\sqrt{\sum_{j=1}^{p} x_j^2} \sqrt{\sum_{j=1}^{p} z_j^2}}$$

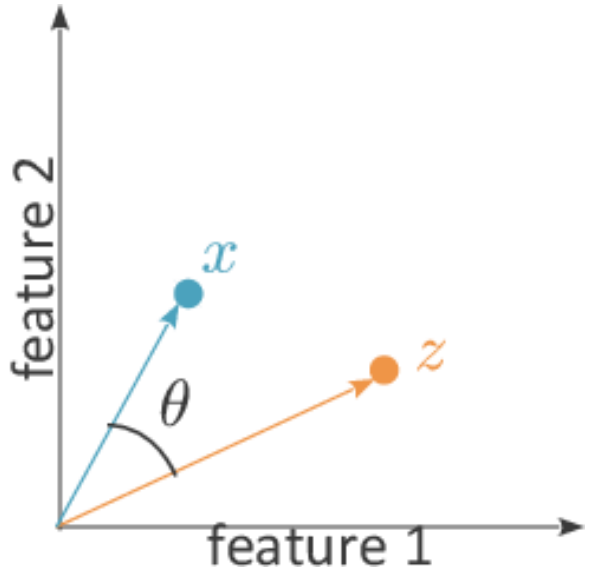
Geometric interpretation?

Similarity between instances

- Pearson's correlation (centered data)

$$\rho(\boldsymbol{x}, \boldsymbol{z}) = \frac{\sum_{j=1}^{p} x_j z_j}{\sqrt{\sum_{j=1}^{p} x_j^2}\sqrt{\sum_{j=1}^{p} z_j^2}} = \frac{\langle \boldsymbol{x}, \boldsymbol{z} \rangle}{||\boldsymbol{x}||.||\boldsymbol{z}||} = \cos\theta$$

- **Cosine similarity:** the dot product can be used to measure similarities.



Categorical features

- Ex: a feature that can take 5 values
  - Sports
  - World
  - Culture
  - Internet
  - Politics
- **Naive encoding:** x_1 in {1, 2, 3, 4, 5}:
  - Why is Sports closer to World than Politics?
- **One-hot encoding:** x_1, x_2, x_3, x_4, x_5
  - Sports: [1, 0, 0, 0, 0]
  - Internet: [0, 0, 0, 1, 0]

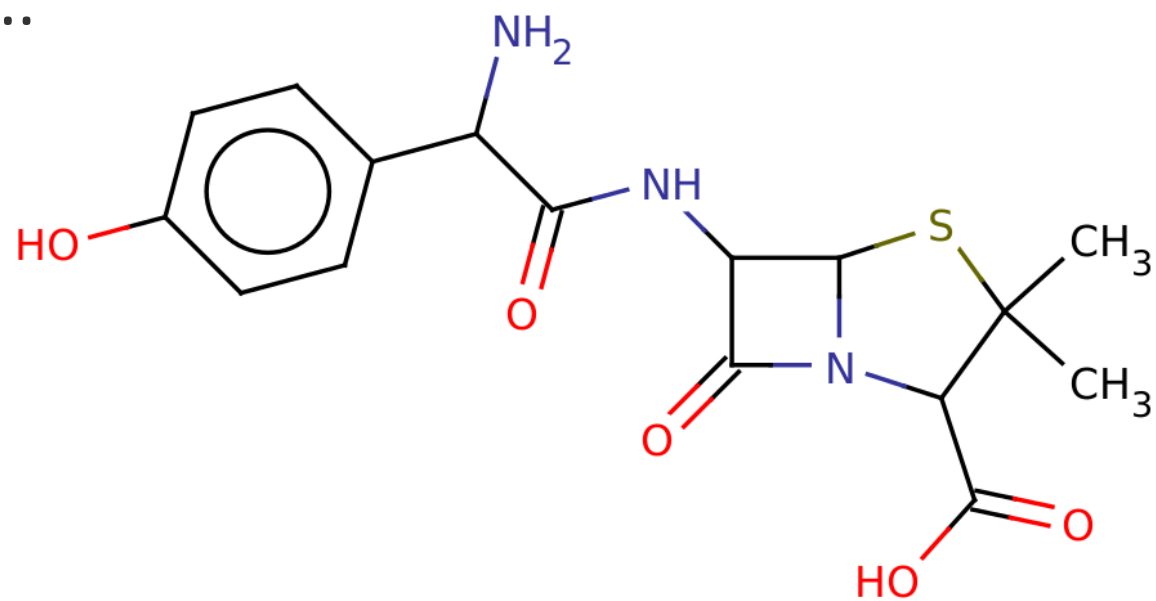
Categorical features

- Represent object as the list of **presence/absence** (or counts) of **features that appear in it.**
- **Example**: small molecules

  features = atoms and bonds of a certain type

  – C, H, S, O, N...

  – O-H, O=C, C-N....



Binary representation

0	1	1	0	0	1	0	0	0	1	0	1	0	0	1

no occurrence of the 1st feature

1+ occurrences of the 10th feature

- **Hamming distance**

  Number of bits that are different

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sum_{j=1}^{p} (x_j^1 \text{ XOR } x_j^2)$$

  Equivalent to ?

Binary representation

0 1 1 0 0 1 0 0 0 1 0 1 0 0 1

no occurrence of the 1st feature

1+ occurrences of the 10th feature

- **Hamming distance**

  Number of bits that are different

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sum_{j=1}^{p} (x_j^1 \text{ XOR } x_j^2)$$

  Equivalent to

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sum_{j=1}^{p} |x_j^1 - x_j^2| = \sum_{j=1}^{p} (x_j^1 - x_j^2)^2$$

Binary representation

0	1	1	0	0	1	0	0	0	1	0	1	0	0	1

- **Tanimoto/Jaccard similarity**

  Number of shared features (normalized)

$$s(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{\sum_{j=1}^{p} (x_j^1 \text{ AND } x_j^2)}{\sum_{j=1}^{p} (x_j^1 \text{ OR } x_j^2)}$$

Counts representation

0	1	2	0	0	1	0	0	0	4	0	1	0	0	7

no occurrence of the 1st feature

occurrences of the 10th feature

- **MinMax similarity**

Number of shared features (normalized)

$$s(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{\sum_{j=1}^{p} \min(x_j^1, x_j^2)}{\sum_{j=1}^{p} \max(x_j^1, x_j^2)}$$

If x is binary, MinMax and Tanimoto are equivalent

$$s(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{\sum_{j=1}^{p} (x_j^1 \text{ AND } x_j^2)}{\sum_{j=1}^{p} (x_j^1 \text{ OR } x_j^2)}$$

Categorical features

- Features

- **Compute the Hamming distance and Tanimoto and MinMax similarities between these objects:**

Categorical features

- Features

- **Compute the Hamming distance and Tanimoto and MinMax similarities between these objects:**

100011010110
300011010120

111011011110
211021011120

111011010100
311011010100

Categorical features

- **A = 100011010110 / 300011010120**
- **B = 111011011110 / 211021011120**
- **C = 111011010100 / 311011010100**

- **Hamming distance**

  d(A, B) = 3 d(A, C) = 3 d(B, C) = 2

- **Tanimoto similarity**

  s(A, B) = 6/9 = 0.67 s(A, C) = 5/8 = 0.63 s(B, C) = 7/9 = 0.78

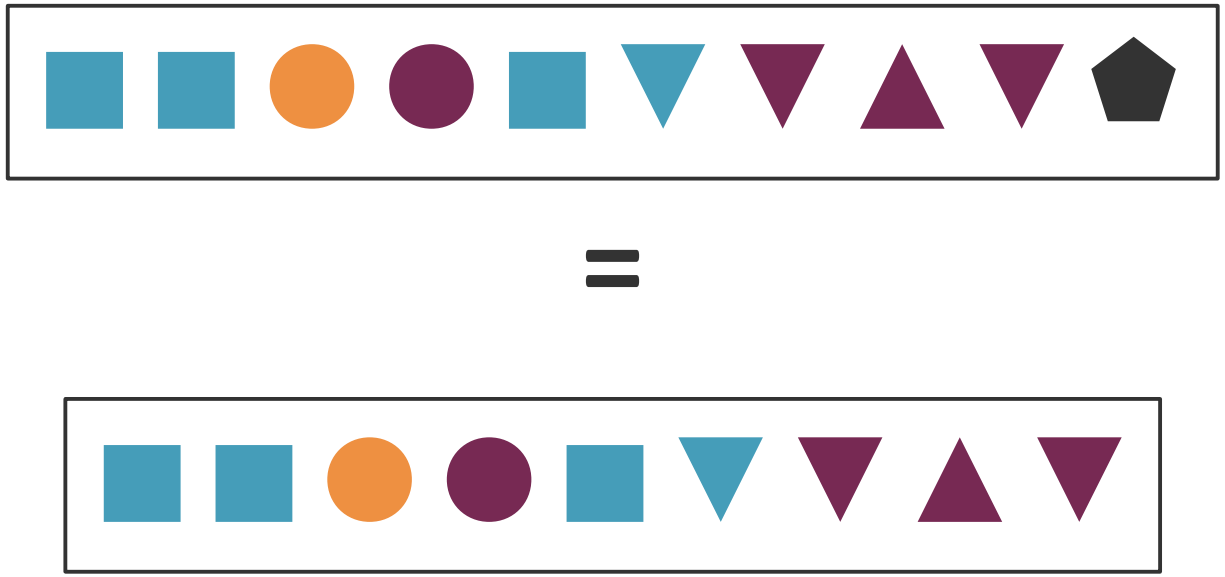
- **MinMax similarity**

  s(A, B) = 8/13 = 0.62 s(A, C) = 7/11 = 0.64 s(B, C) = 8/13 = 0.62

Categorical features

- Features

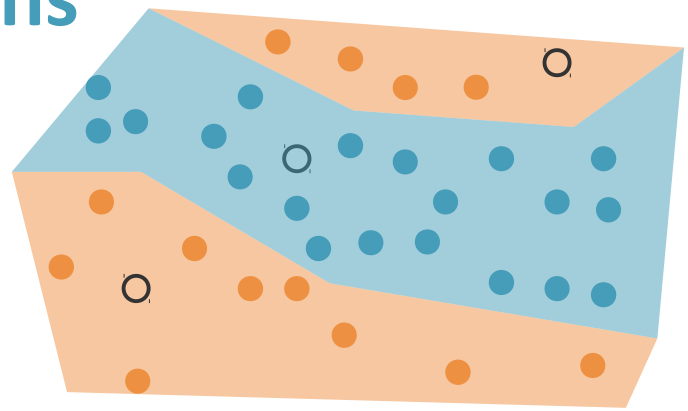
- When new data has unknown features: ignore them.



Back to nearest neighbors

Advantages of kNN

- **Training is very fast**
  - Just store the training examples.
  - Can use smart indexing procedures to speed-up testing (slower training).
- **Keeps the training data**
  - Useful if we want to do something else with it.
- Rather robust to **noisy data** (averaging k votes)
- Can learn **complex functions**



Drawbacks of kNN

- **Memory** requirements
- Prediction can be **slow.**
  - Complexity of labeling 1 new data point

Drawbacks of kNN

- **Memory** requirements
- Prediction can be **slow.**

  Complexity of labeling 1 new data point: $\mathcal{O}(pn + n \log k)$

  But kNN works best with lots of samples...

  → Efficient data structures (**k-D trees**, **ball-trees**)

  - construction space: $\mathcal{O}(pn)$ time: $\mathcal{O}(n \log n)$
  - query: $\mathcal{O}(p \log n)$ if $p < 20, \mathcal{O}(pn)$ otherwise

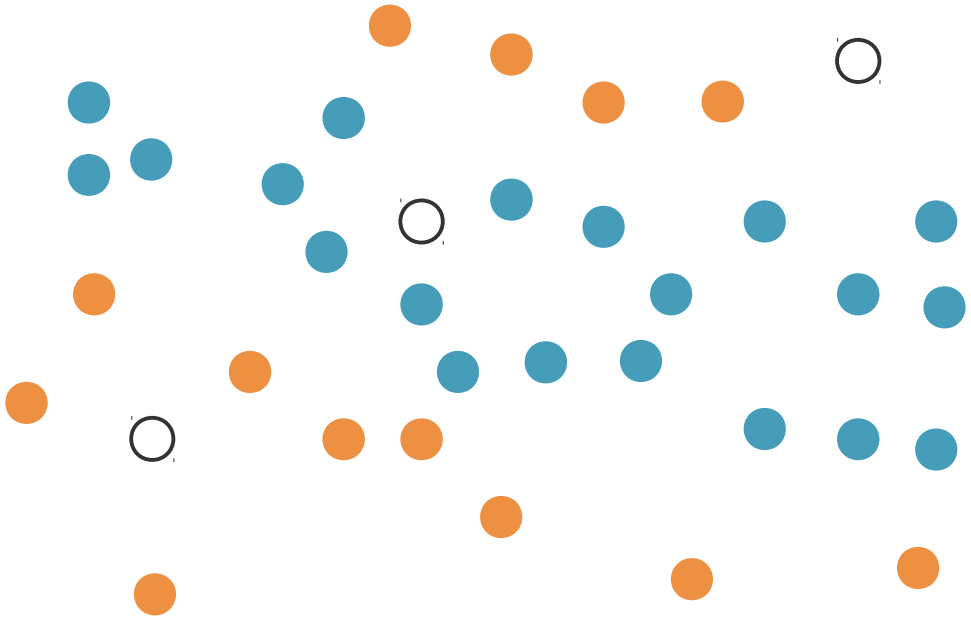
  $\mathcal{O}(p \log n)$

  → Approximate solutions based on **hashing**
- kNN are fooled by **irrelevant attributes.**

  E.g. p=1000, only 10 features are relevant; distances become meaningless.

Decision boundary of kNN

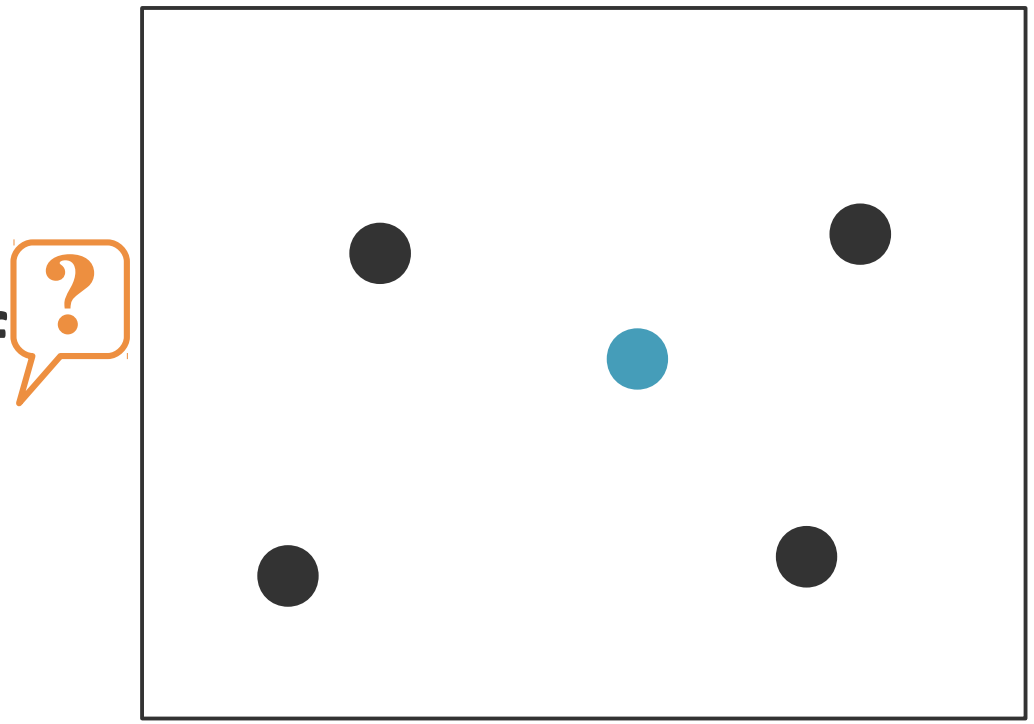
- Classification
- **Decision boundary:** Line separating the positive from negative regions.
- **What decision boundary is the kNN building?**



Voronoi tesselation

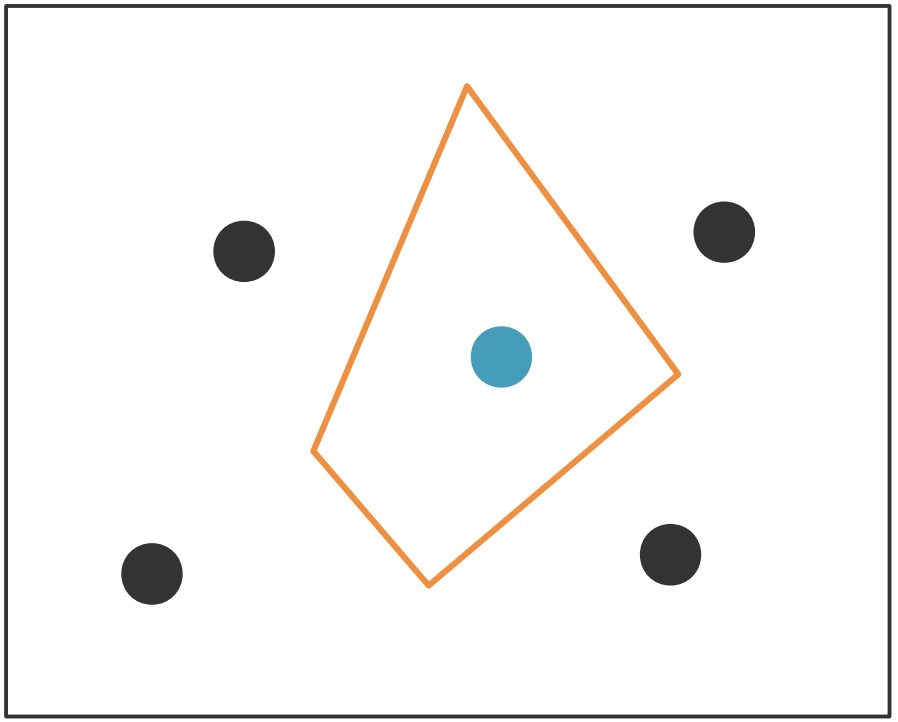
- **Voronoi cell** of **x**:
  - set of all points of the space closer to **x** than any other point of the training set
  - polyhedron
- **Voronoid tesselation** of the space: union of all Voronoi cells.

Draw the Voronoi cell of the blue dot.



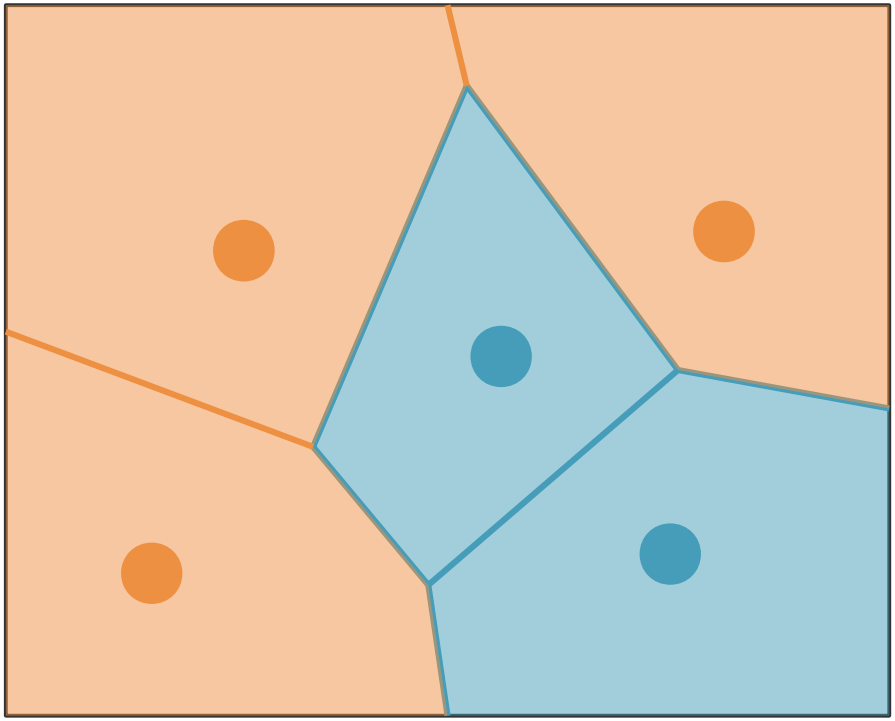
Voronoi tesselation

- **Voronoi cell** of **x**:
  - set of all points of the space closer to **x** than any other point of the training set
  - polyhedron
- **Voronoid tesselation** of the space: union of all Voronoi cells.



Voronoi tesselation

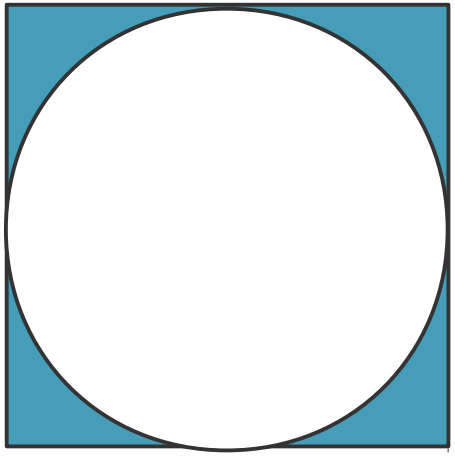
- The Voronoi tesselation defines the decision boundary of the 1-NN.



- The kNN also partitions the space (in a more complex way).

Curse of dimensionality

- Remember from Chap 3
- When p ↗ the proportion of a hypercube outside of its inscribed hypersphere approaches 1.
- Volume of a p-sphere: $\frac{2r^2\pi^{p/2}}{p\Gamma(p/2)}$



- What this means:
  - hyperspace is very big
  - all points are far apart
  - dimensionality reduction needed.

kNN variants

- **ε-ball neighbors**
  - Instead of using the k nearest neighbors, use **all points within a distance ε** of the test point.
  - What if there are no such points?

kNN variants

- **Weighted kNN**
  - **Weigh the vote of each neighbor** according to the distance to the test point.

$$w_l = \exp\left(\frac{1}{2} d(\boldsymbol{x}, \boldsymbol{x}^l)\right)$$

  - Variant: **learn** the optimal weights [e.g. Swamidass, Azencott et al. 2009, Influence Relevance Voter]

Collaborative filtering

- **Collaborative filtering:** recommend items that similar users have liked in the past

  similar users = users with similar tastes

- **item-based kNN**
  - similarity between items: **adjusted cosine similarity**

Sum over the users that rated both item A and item B

$$s(A,B) = \frac{\sum_u (R(u,A) - \bar{R}(u))(R(u,B) - \bar{R}(u))}{\sqrt{\sum_u (R(u,A) - \bar{R}(u))^2 \sum_u (R(u,B) - \bar{R}(u))^2}}$$

Rating of item A by user u

Average rating by user u

Collaborative filtering

- score of item A for user u:

$$S(u, A) = \frac{\sum_{B \in \mathcal{N}_u^k(A)} s(A, B) R(u, B)}{\sum_{B \in \mathcal{N}_u^k(A)} |s(A, B)|}$$

k nearest neighbors of A
according to s
among the items rated by user u

Summary

- **kNN**
  - very simple training
  - prediction can be expensive
- Relies on a "good" **distance/similarity** between instances
- **Decision boundary** = **Voronoi tesselation**
- **Curse of dimensionality:** hyperspace is very big.

References

- *A Course in Machine Learning.* `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`
  - **kNN:** Chap 3.2 — 3.3
  - **Categorical variables:** Chap 3.1
  - **Curse of dimensionality:** Chap 3.5
- More on
  - **Kd-trees**

  `https://www.ri.cmu.edu/pub_files/pub1/moore_andrew_1991_1/moore_andrew_1991_1.pdf`

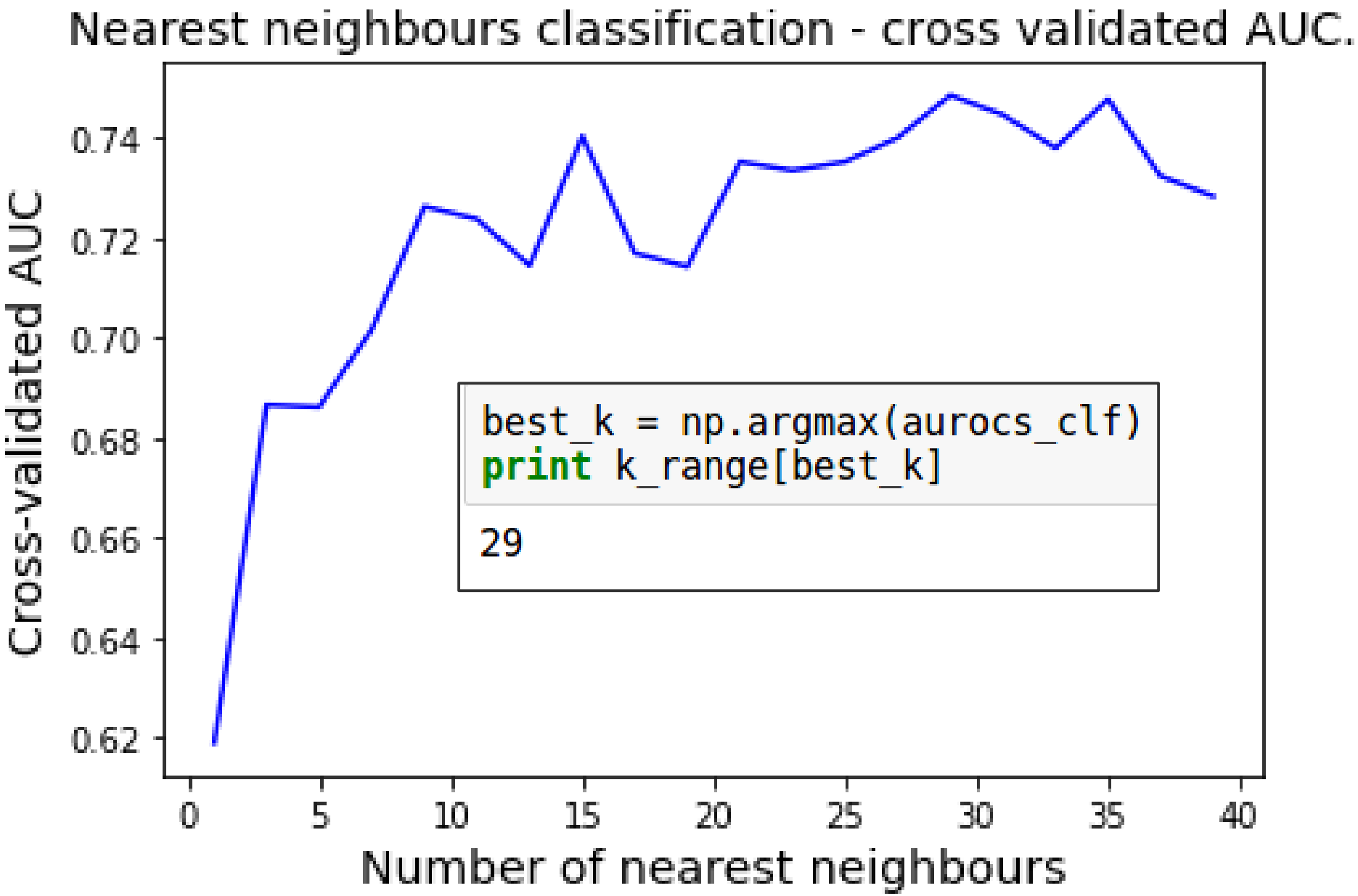
  `http://www.alglib.net/other/nearestneighbors.php`

  - **Voronoi tessellation**

  `http://philogb.github.io/blog/2010/02/12/voronoi-tessellation/`

Lab

```
from sklearn import neighbors
from sklearn import metrics

aurocs_clf = []
# Create a range of values of k. We will use this throughout the lab.
k_range    = range(1,40,2)

for k in k_range:
    clf    = neighbors.KNeighborsClassifier(n_neighbors=k)
    y_pred = cross_validate(X_clf, y_clf, clf, cv_folds)

    fpr, tpr, thresholdss = metrics.roc_curve(y_clf, y_pred[:,1])
    aurocs_clf.append(metrics.auc(fpr,tpr))
```




```
best_k = np.argmax(aurocs_clf)
print k_range[best_k]
```

```
29
```

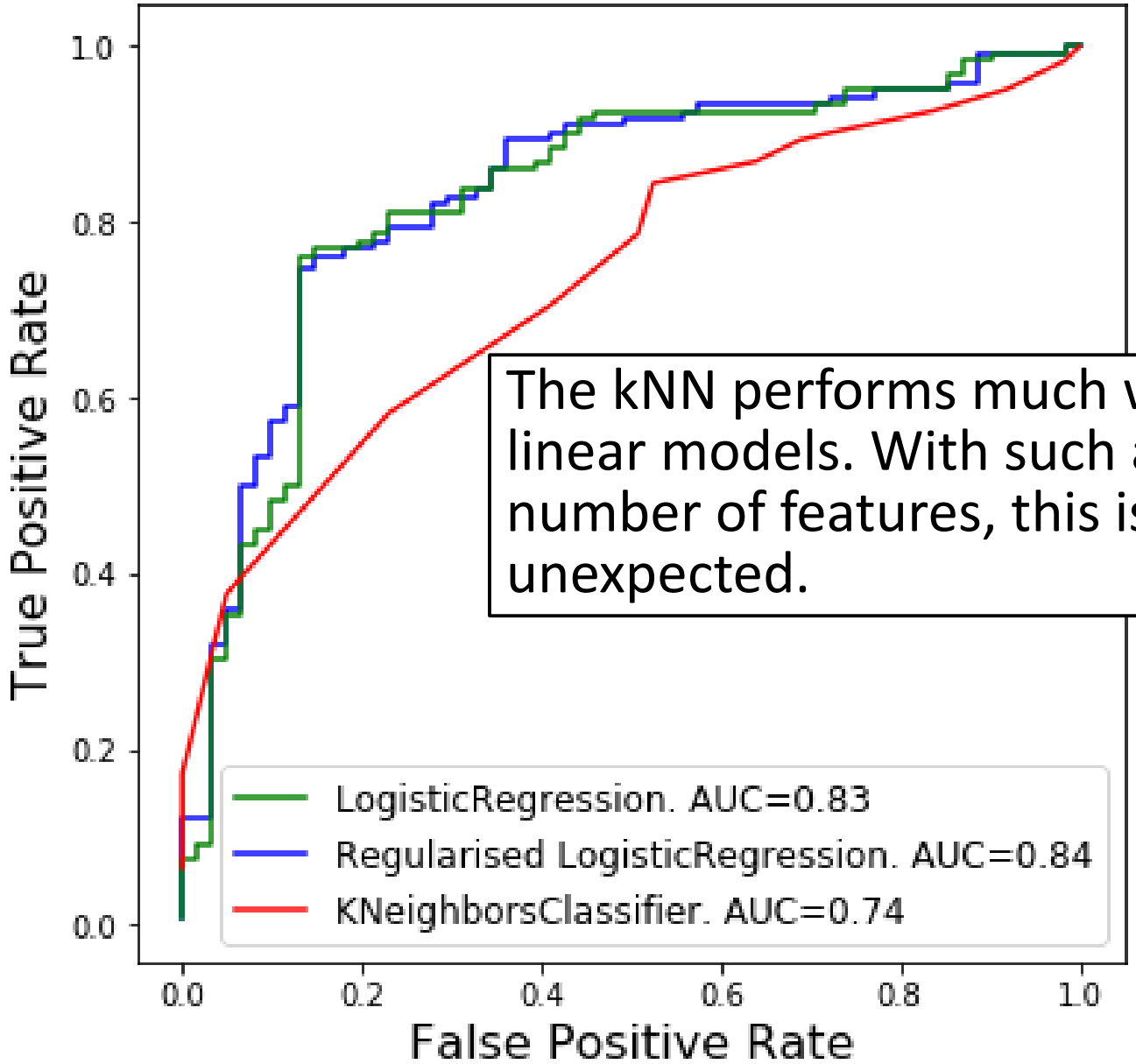
```
from sklearn import model_selection
from sklearn import metrics

classifier = neighbors.KNeighborsClassifier()

param_grid = {'n_neighbors': k_range}
clf_knn_opt = model_selection.GridSearchCV(classifier,
                                           param_grid=param_grid,
                                           cv=cv_folds,
                                          scoring='roc_auc')
clf_knn_opt.fit(X_clf, y_clf)
```

```
# Find the best parameter
print clf_knn_opt.best_params_
```

```
{'n_neighbors': 25}
```

Even though we use the same scoring strategy, we don't get the same optimum. That's because the cross-validation evaluation strategy is different: scikit-learn compute one AUC per fold and averages them.

ROC Curves comparison for logistic regression and k-nearest neighbours classifier.



The kNN performs much worse than the linear models. With such a large number of features, this is not unexpected.

```
classifiers = {}
y_preds     = {}
# Fix a set of distance metrics to use
d_metrics = ['euclidean', 'cityblock', 'correlation' , 'cosine']
aurocs    = {}

for m in d_metrics:
    aurocs[m] = []
    for k in k_range:
        classifiers[m] = neighbors.KNeighborsClassifier(n_neighbors=k, metric=m)
        y_preds[m]     = cross_validate(X_clf, y_clf, classifiers[m], cv_folds)

        fpr, tpr, thresholds = metrics.roc_curve(y_clf, y_preds[m][:,1])
        auc                  = metrics.auc(fpr, tpr)
        aurocs[m].append(auc)

        print 'Metric = %-12s | k = %3d | AUC = %.3f.' %(m, k, aurocs[m][-1])
```

```
for i in range(len(d_metrics)):
    plt.plot(k_range, aurocs[d_metrics[i]])

plt.plot(k_range, [logreg_l2_auc for kval in k_range])

plt.xlabel('Number of nearest neighbors', fontsize=14)
plt.ylabel('Cross-validated AUC', fontsize=14)
plt.title('Nearest neighbors classification', fontsize=14)

legends = [m for m in d_metrics]
legends.append('Logistic regression')
plt.legend(legends, fontsize=12)
```

Computing nearest neighbors based on **correlation** works better than based on Minkowski distances. Indeed this allows to compare the **profiles** of the gene expressions (which genes have high expression / low expression simultaneously). Still logistic regression works best.