IN2110 SPRING 2022
SPRÅKTEKNOLOGISKE METODER

Erik Velldal & Jan Tore Lønning

Plan

Lectures 2-5

- How to represent (language) data in a mathematical model.
- Vector space models.
- Representing
  - Documents (today)
  - Words (week 5)
- Vector-based machine learning
  - Classification (week 3)
  - Clustering (week 4)

Today

- Machine learning
  - supervised
    - classification
- Rocchio classifier
- *k* Nearest Neighbors (*k*NN)

Disclaimer

- I am only a substitute teacher for Erik Velldal
- The slides will be a mixture
  - Erik's slides from last year
  - My slides from IN3050 and IN4080
  - Some new slides (like this one)

Machine learning

Machine Learning

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience. (Wikipedia)

In particular :

- **Generalization**: Provide sensible outputs for inputs not encountered during training
- **extracting relevant information** from data and **applying it to analyze new data.**

Three main types of ML

Supervised learning

- Learn from labeled data

-  ?

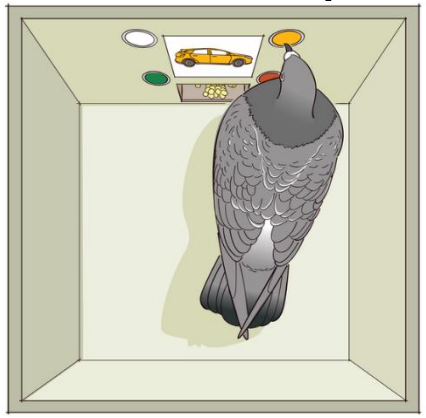
Unsupervised learning

- No labeled data

- Task: identify similarities and categorize together

Reinforcement learning

- Training with rewards (and punishments)



Source: Wikipedia

1. Classification, supervised learning

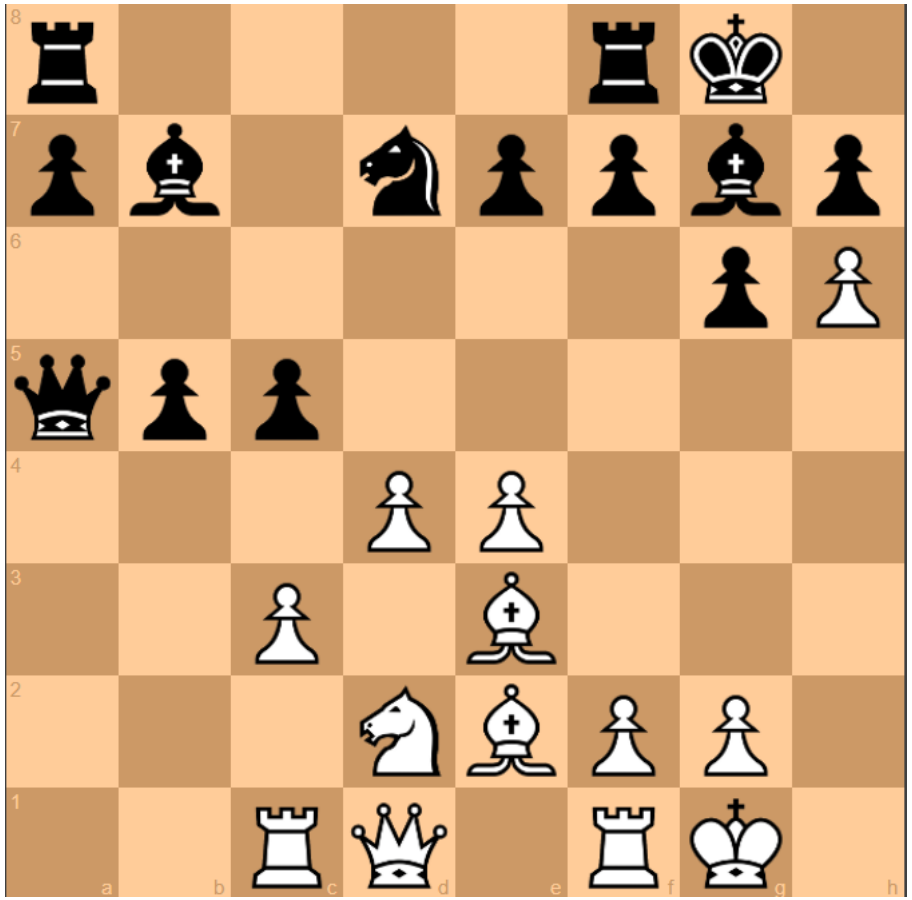
- Requires training data; pre-defined examples of what we want the algorithm to learn.
- Learning from labeled data.

2. Unsupervised learning

- *Can you sort the Lego bricks?*
  - (No instruction on how)
- You may choose sorting on
  - Color, or
  - Size, or
  - Shape, or
  - A combination
- I cannot know beforehand what you choose, but
- The result might me helpful

3. Reinforcement example: AlphaZero

- It beats humans in Go and Chess
- Totally self-learned by playing against itself:
  - Reinforcement learning
  - Neural nets to generalize over game states
- Human players, e.g., Carlsen, has learned new strategies from the program
  - "The *h* pawn"
- Trial and error



https://www.aftenposten.no/sport/sjakk/i/K3K31M/ny-oppfinnelse-er-en-sjakk-sensasjon-slik-paavirkes-carlsen

Today: Classification, supervised learning

- Requires training data; pre-defined examples of what we want the algorithm to learn.
- Learning from labeled data.

What is a classifier?

- A domain of objects/observations we are to classify
- Labels: A finite set of labels
- Classifier: A mapping which maps each object to a unique label

Classification in NLP

Text classification

- Spam detection
- Genre classification
- Language identification
- Authorship attribution
- Sentiment analysis:
  - Positive-negative
- Abusive language detection
- Etc.

Other classification tasks in NLP

- Word sense disambiguation
- Sentence splitting
- Tagging
- Named-entity recognition
- Language models
  - Predict the next word in a sentence
- etc.

Types of classifiers ("one of")

- Binary classification:
  - Two classes
- Multiclass classification:
  - Three or more classes
  - Each object belongs to one class

- Observe:
  - Some learning algorithms are by nature binary (e.g., the perceptron, Logistic regression) and have to be adapted to multiclass classification
  - Both Rocchio and *k*NN are multiclass by nature.

Multi-label classification ("any-of")

- Classify an object with respect to several binary classes
- E.g., *Who is in the picture?*
- (Uncle Tom: 1, Aunt Mary: 1, Grandma: 0, Grandpa: 0, etc.)

- Topic classification of documents
  - Example, both: China, sport

Transforming the classification task

- The task of predicting from an e-mail

yes/no

- is transformed to the task of predicting from some features

(Chars: 21,705, Lines: 551, 'dollar': 0,
'winner': no, format: html, number: small)

yes/no

- is transformed to the task of predicting from a numerical vector to a number

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

$y \in \{0, 1\}$

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

From Jurafsky & Martin

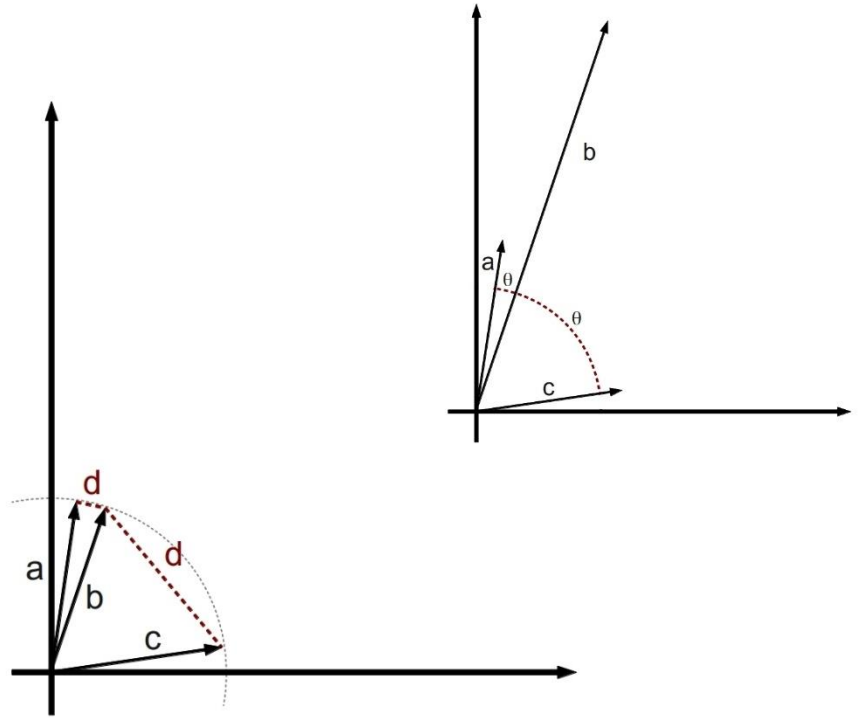
Term-document matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Example of a co-occurrence matrix
- More specifically, a $m \times n$ term-document matrix
  - m terms, n documents
- Count the number of occurrences of the terms in each document
- Each column represent a document
- Each row represents a term (word, feature)
- With 4 key words each document is represented as a 4-d vector
- (We could use any set of key words)

The geometrical view

- The classifiers we consider today can best be described and understood in geometrical terms
- Strictly speaking, we only need points, and Euclidean distances in an *n*-dimensional space - not directions and cosine
- For documents, we can think of them as length-normalized tf-idf representations.



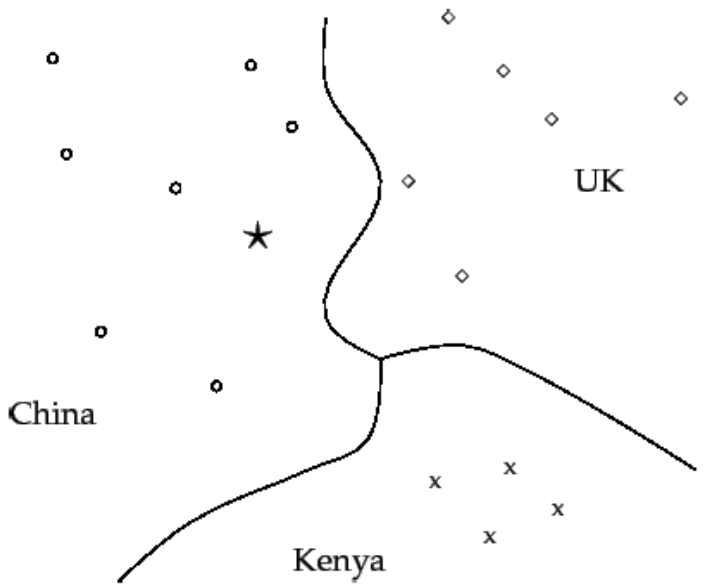
- But they can also be adopted to use cosine similarity instead of Euclidean distance

Rocchio classification

Also called Nearest Centroid

Classes and classification

- In our vector space model, objects are represented as points, so classes will correspond to collections of points; regions.
- Vector space classification is based on the contiguity hypothesis:

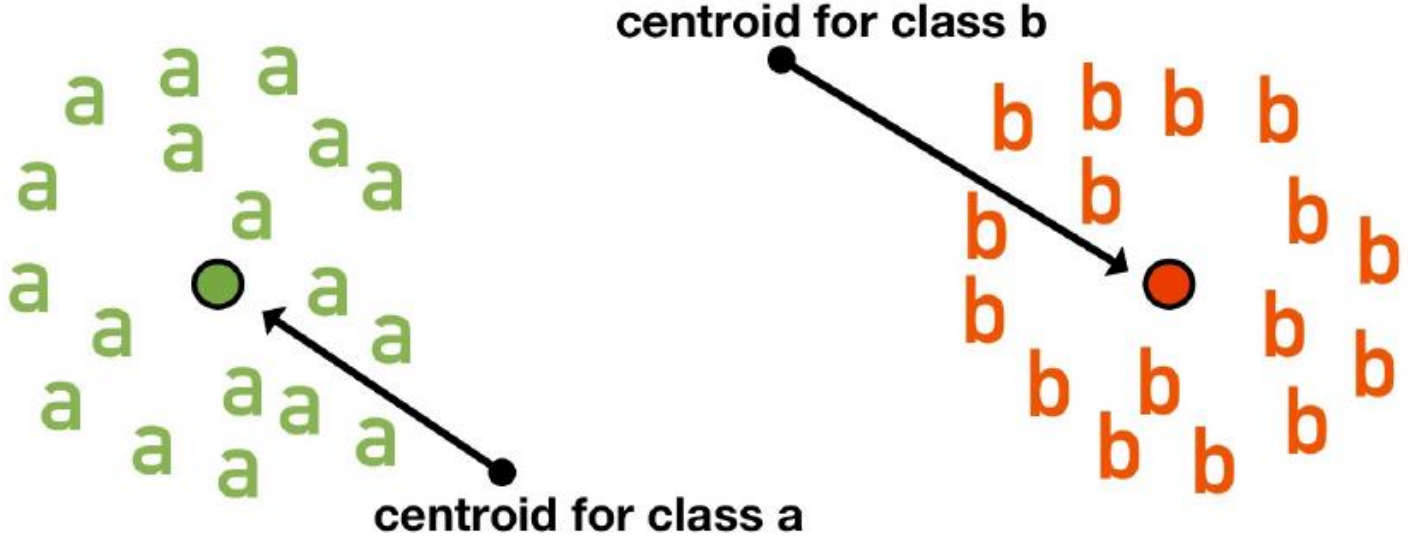


- Objects in the same class form a contiguous region, and regions of different classes do not overlap.
- Classification amounts to computing the boundaries in the space that separate the classes; the decision boundaries.

Rocchio classification (1:2)

- Each class C_i is represented by its centroid,
- or 'center of gravity'.
- Computed as the average of the vectors $\boldsymbol{x}_j$ of its members;

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$$

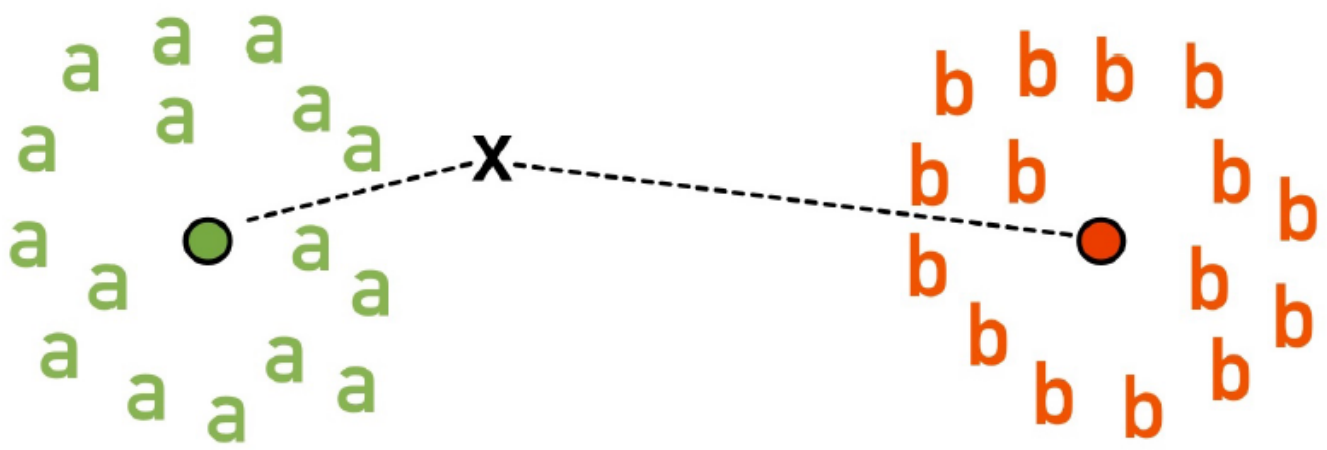


Computing a centroid; example

- General formula: $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$
- Given a class label '*music*' with 3 training documents represented by n-dimensional vectors $C_{music} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}\}$:
  $\boldsymbol{x_1} = \langle x_{11}, x_{12}, \ldots, x_{1n} \rangle$
  $\boldsymbol{x_2} = \langle x_{21}, x_{22}, \ldots, x_{2n} \rangle$
  $\boldsymbol{x_3} = \langle x_{31}, x_{32}, \ldots, x_{3n} \rangle$
- The centroid would be computed as:
  $\boldsymbol{\mu}_{music} = \langle \frac{x_{11}+x_{21}+x_{31}}{3}, \frac{x_{12}+x_{22}+x_{32}}{3}, \ldots, \frac{x_{1n}+x_{2n}+x_{3n}}{3} \rangle$

Rocchio classification (2:2)

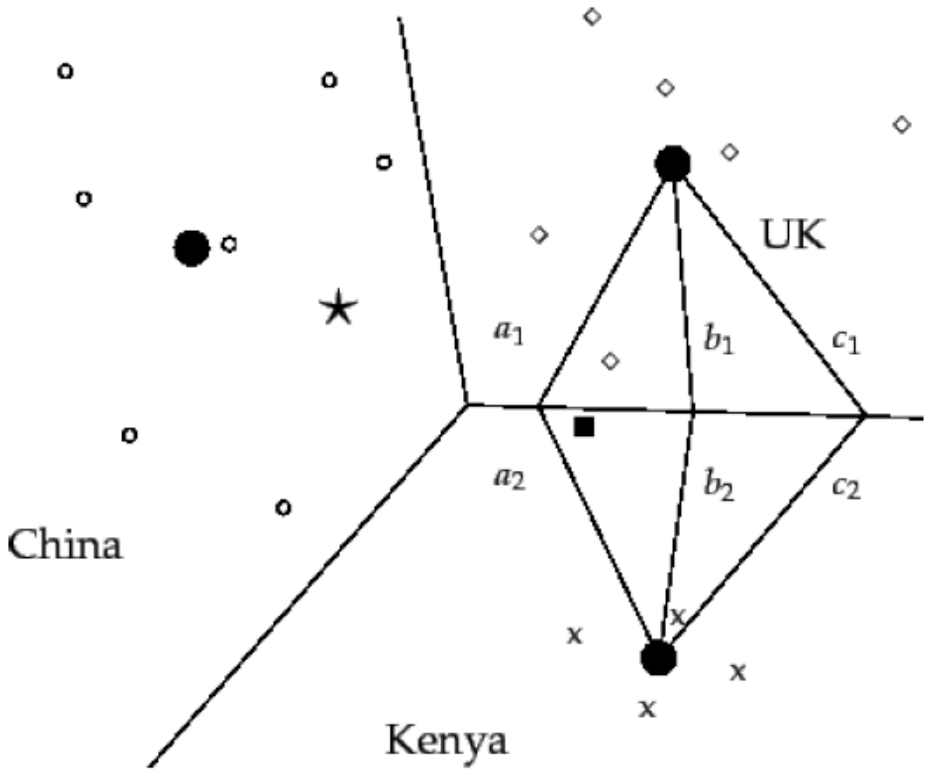
- The Rocchio decision rule:
- To classify a new object o_j (represented by a feature vector $\boldsymbol{x}_j$);
  - determine which centroid $\boldsymbol{\mu}_i$ that $\boldsymbol{x}_j$ is closest to,
  - and assign it the corresponding class label i.

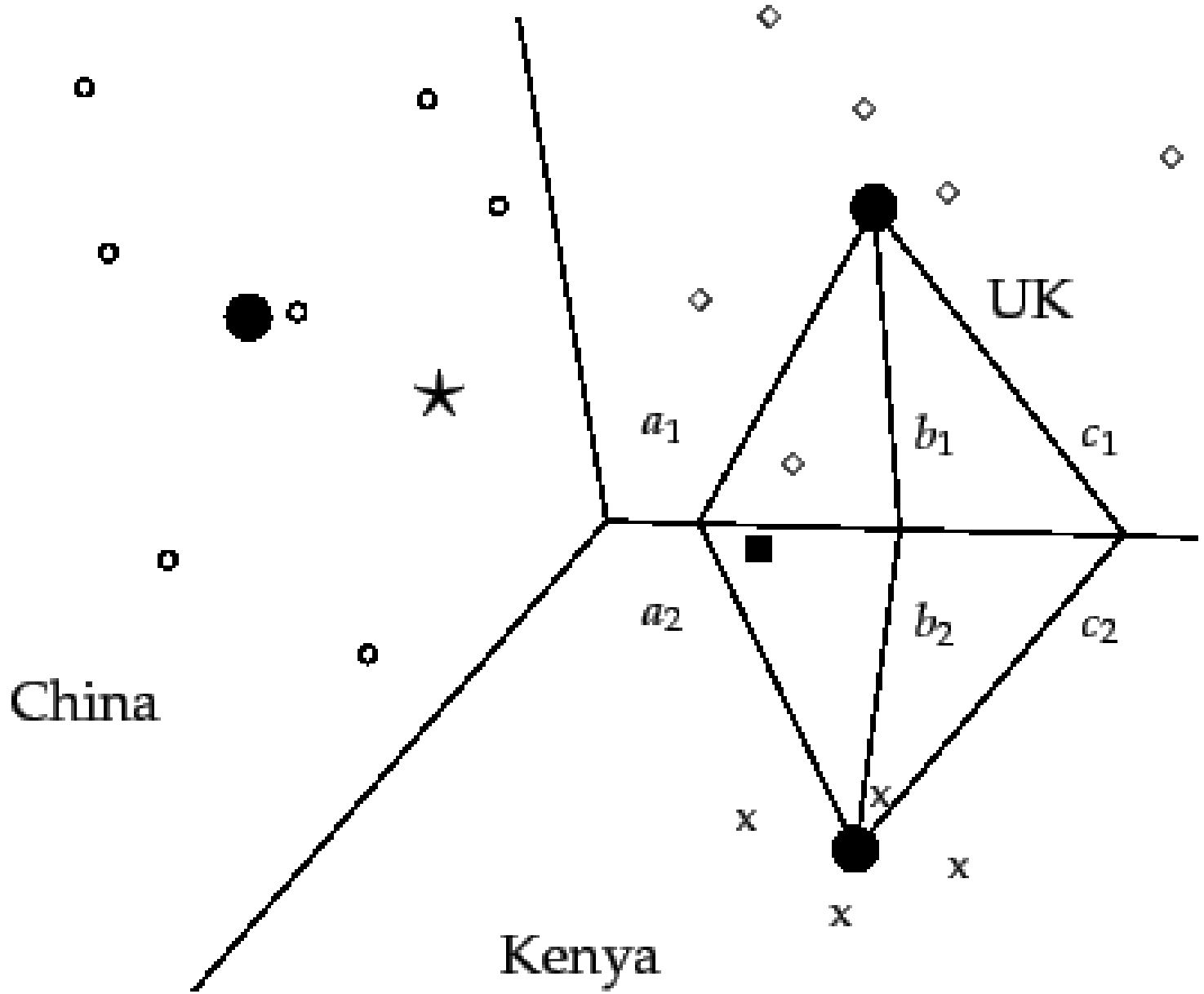


- AKA nearest centroid classifier or nearest prototype classifier.

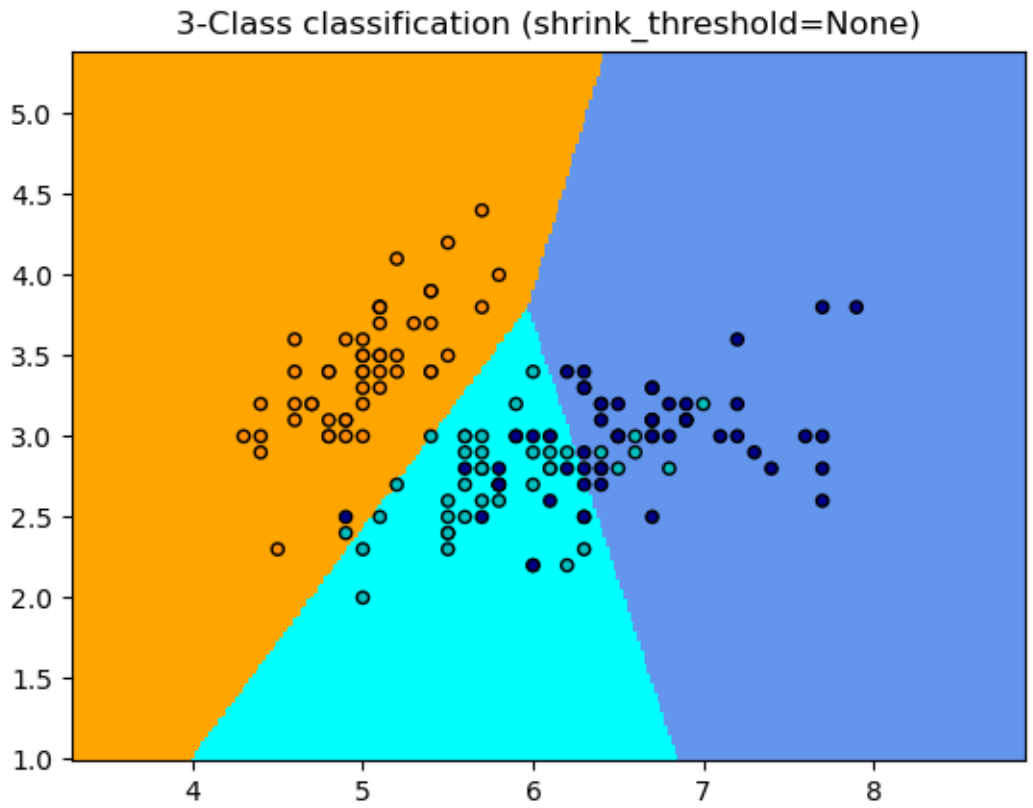
The decision boundary of the Rocchio classifier

- Defines the boundary between two classes by the set of points equidistant from the centroids.
- In two dimensions, this set of points corresponds to a *line*.
- Correspods to a plane in 3D or a hyperplane in higher dimensions.
- Boundaries *implied* by the decision rule; not explicitly computed.




UK
China
Kenya
a_1
b_1
c_1
a_2
b_2
c_2

Another example (iris dataset)

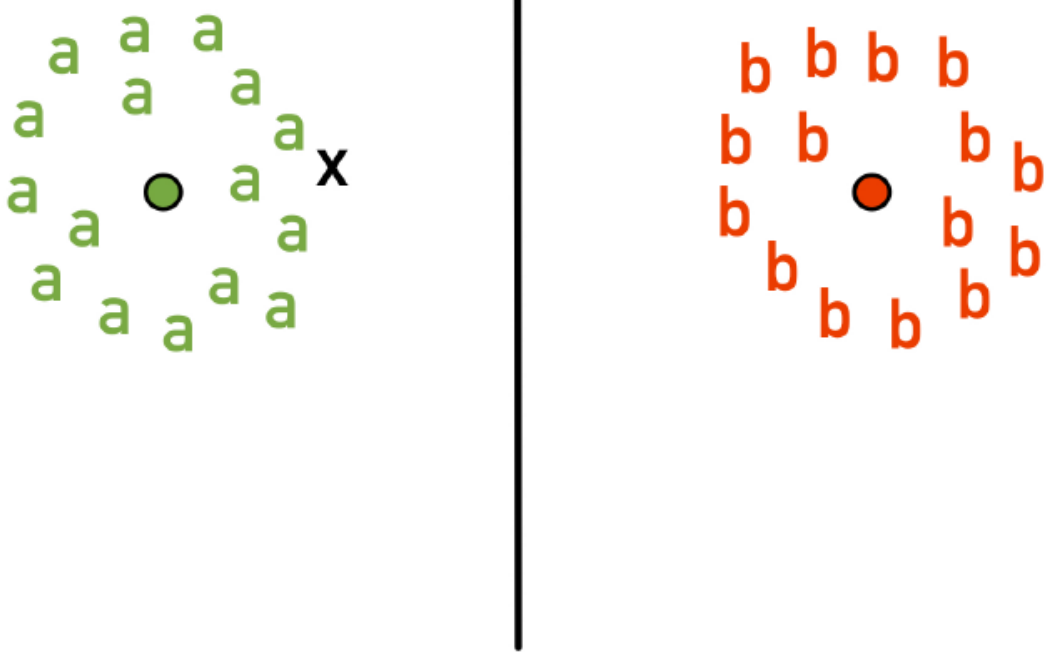


https://scikit-learn.org/stable/auto_examples/neighbors/plot_nearest_centroid.html#sphx-glr-auto-examples-neighbors-plot-nearest-centroid-py

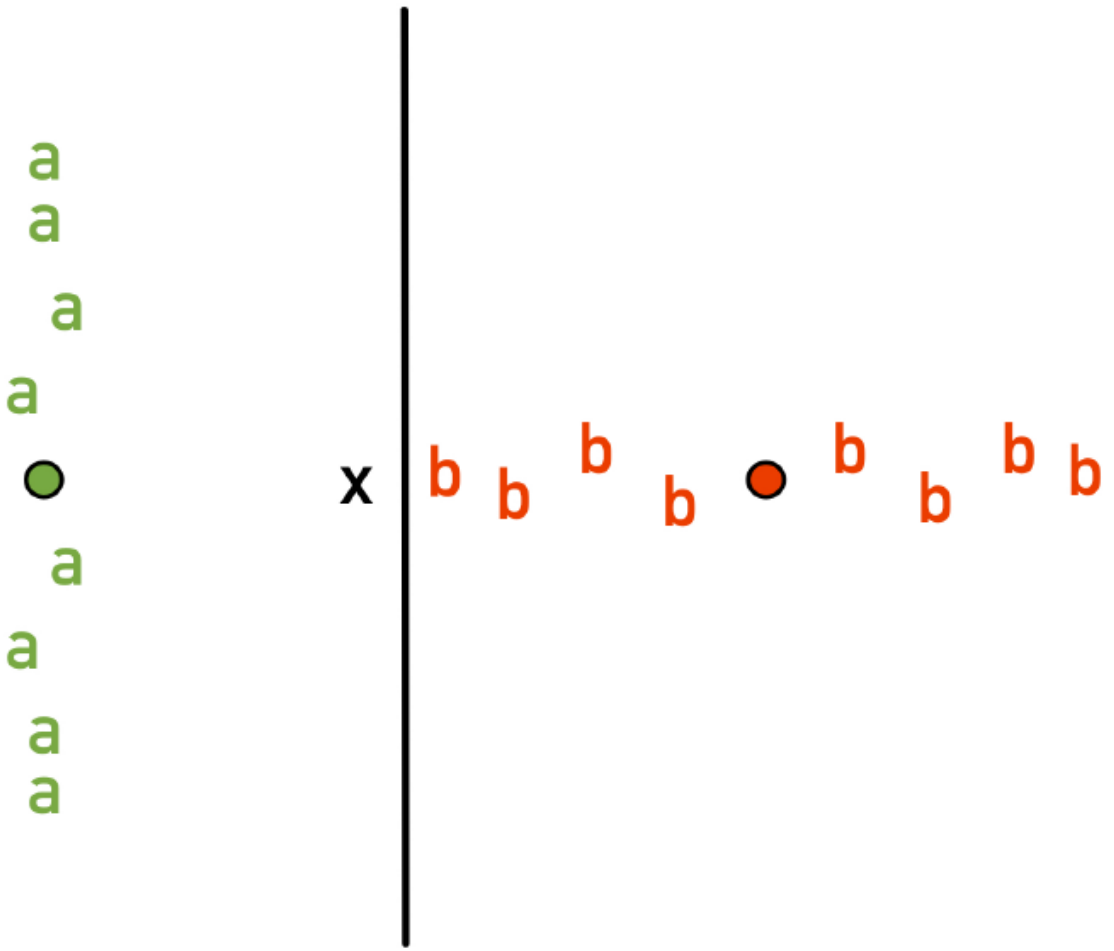
Problems with the Rocchio classifier

- The classification decision ignores the distribution of members locally within a class, only based on the centroid distance.
- Implicitly assumes that classes are spheres with similar radii.
- Does not work well for classes than cannot be accurately represented by a single prototype or center (e.g. disconnected or elongated regions).
- Because the Rocchio classifier defines a linear decision boundary, it is only suitable for problems involving *linearly separable* classes.

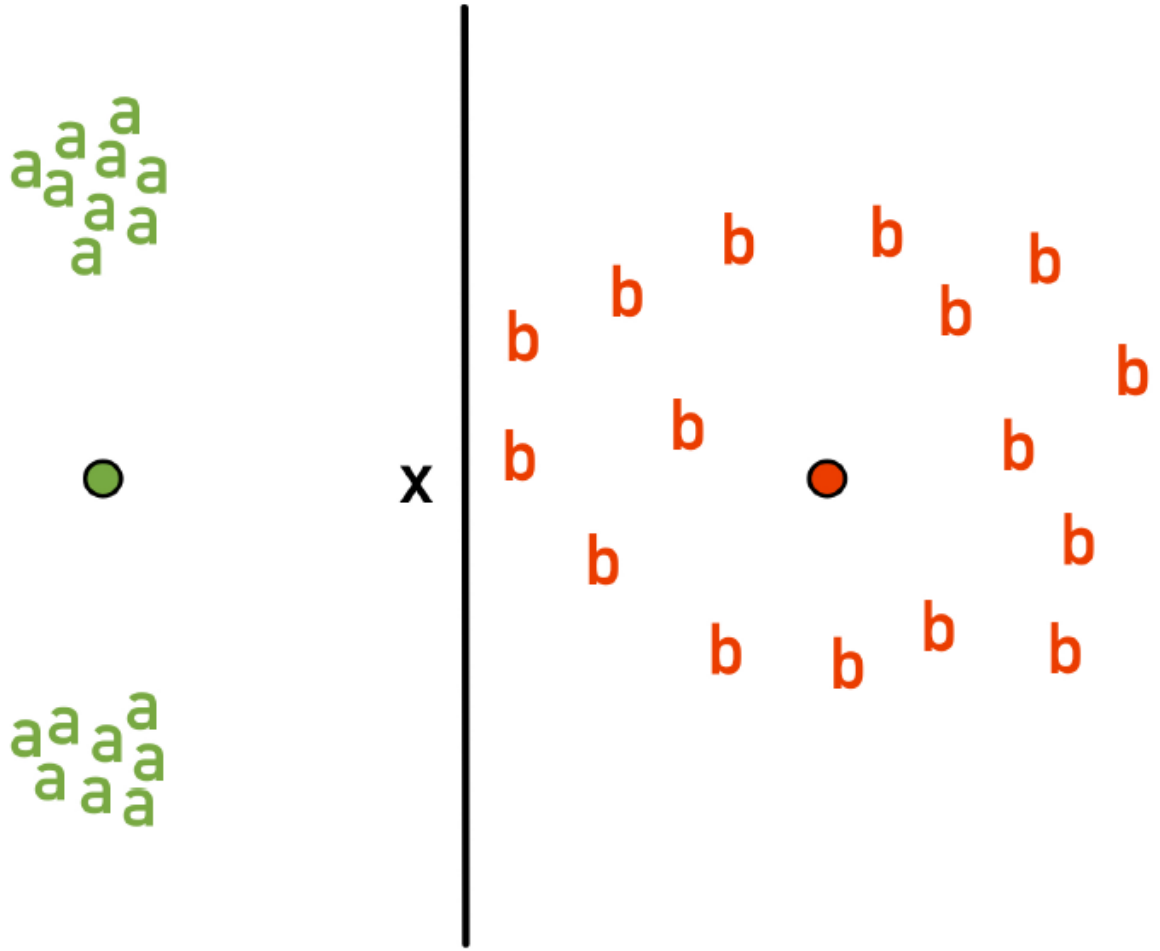
Ideal



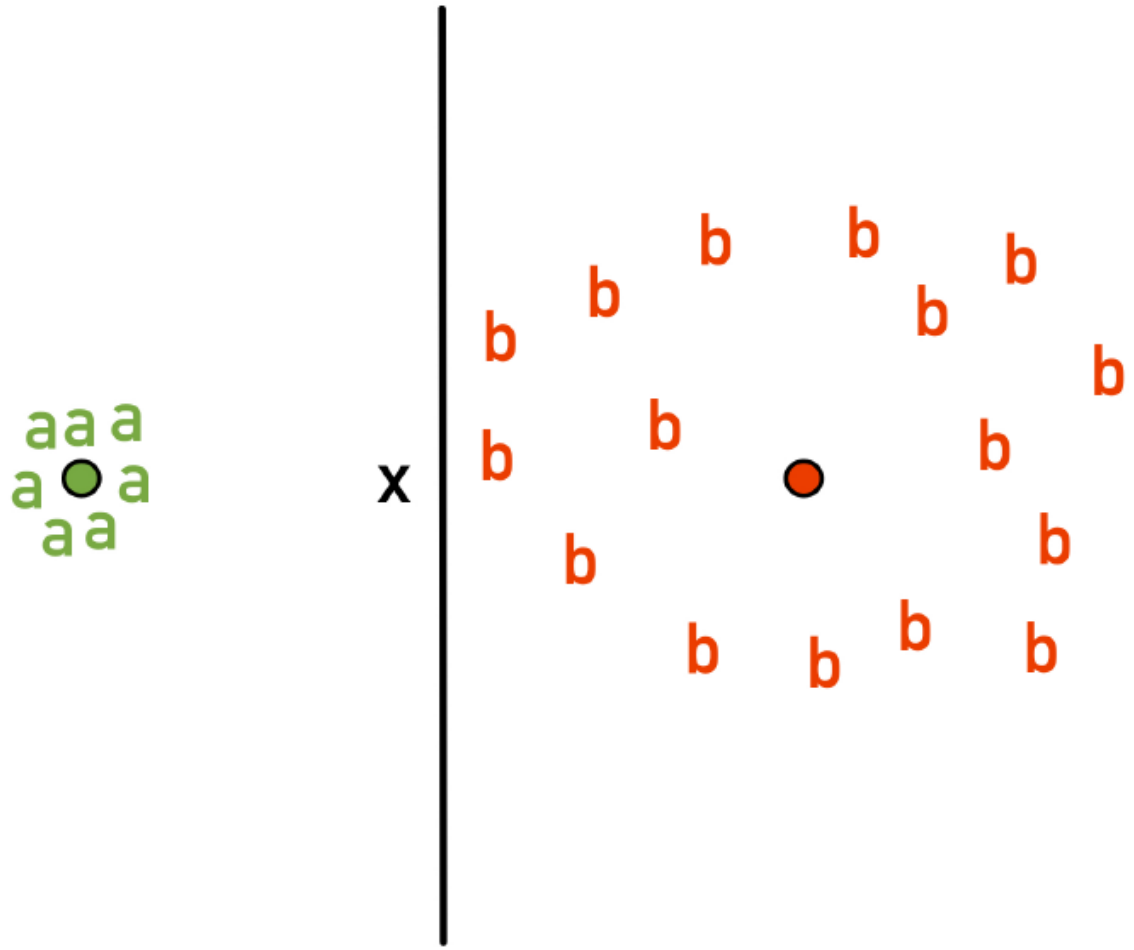
Problematic: Elongated regions



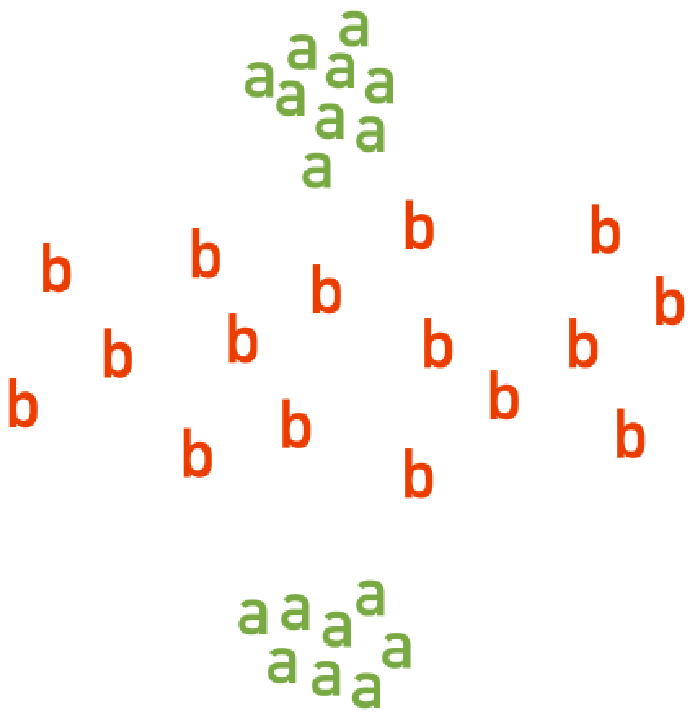
Problematic: Non-contiguous regions



Problematic: Different sizes



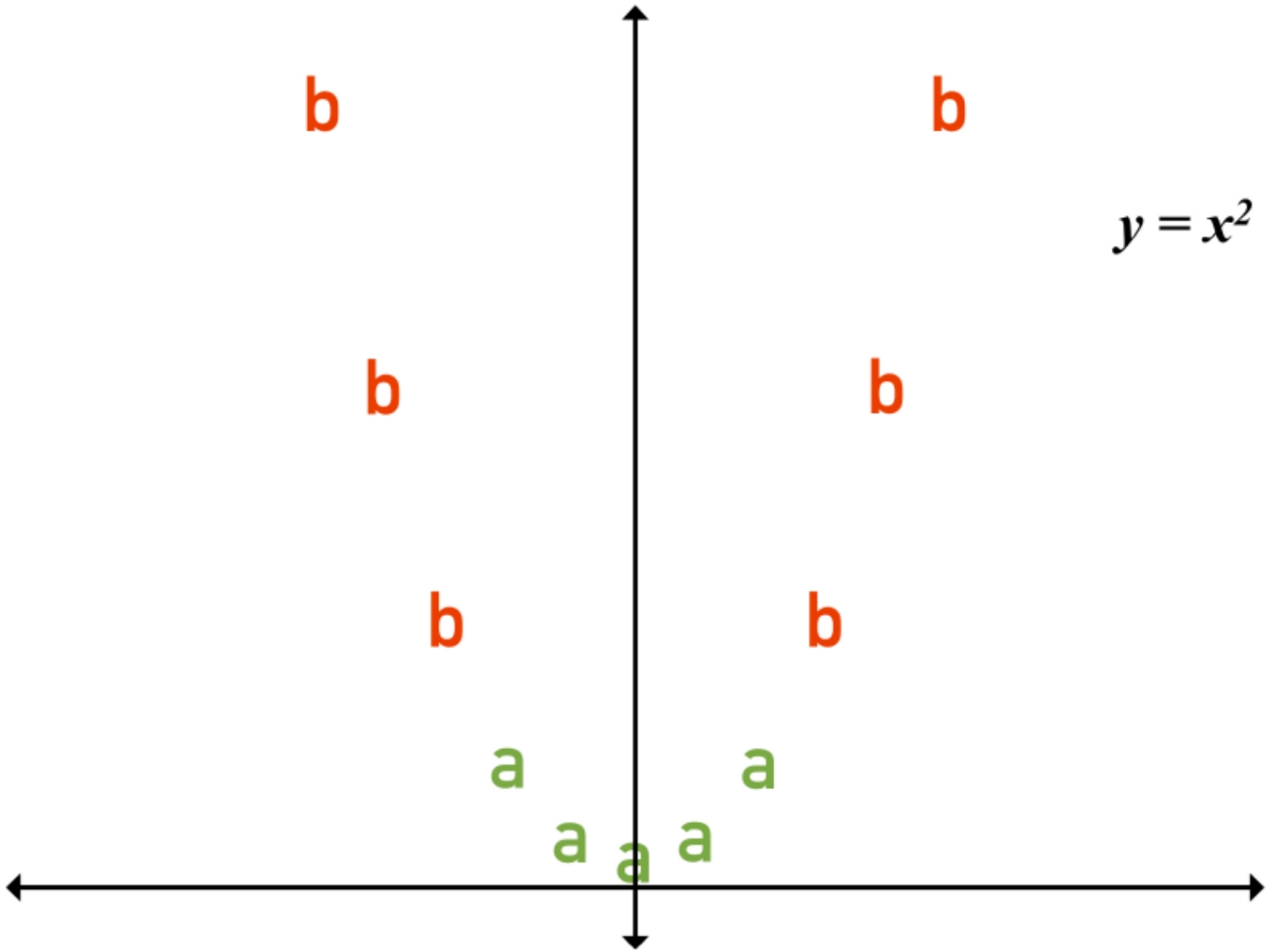
Problematic: Nonlinear boundary



A side-note on non-linearity

- Before we turn to talk about non-linear classifiers, note that:
  Classes that are not linearly separable in a given feature space...

A side-note on non-linearity



- . . . may become linearly separable when the features are mapped to a higher-dimensional space (this is the basis for so-called kernel methods).

Linear and non-linear classifiers

Feature engineering

- In 2008 (the IR-book):
  - Success in ML was based on:
    - Feature selection: choosing the right features
    - Feature engineering: adopting and manipulating the features
    - Kernels (automatic feature engineering), in particular SVM-methods (IR, ch. 15)

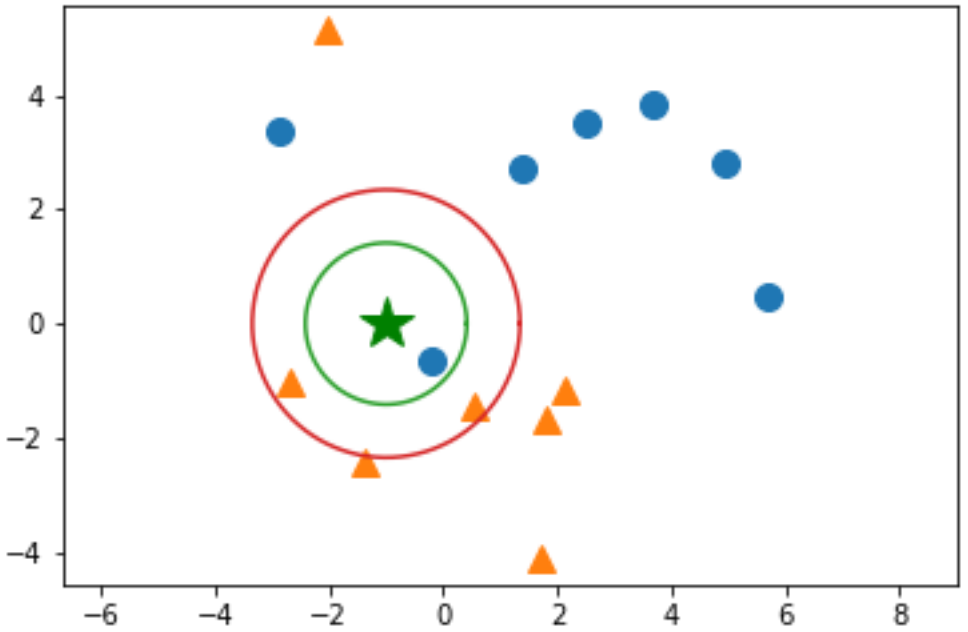
Neural networks

- Today, the best ML, also in NLP, is achieved with (deep) neural networks:
  - Less feature engineering
  - More model engineering
  - Needs large training data

k Nearest Neighbors (*k*NN)

*k*NN algorithm

1. Calculate the distance to all the training instances
2. Pick the *k* nearest ones
3. Choose the majority class for this set



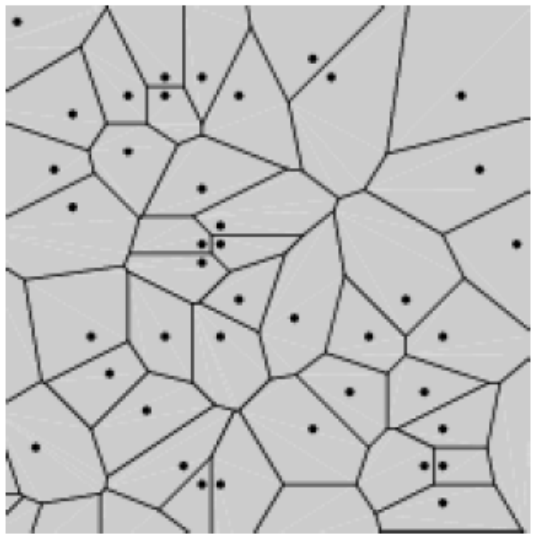
What is the result with

$k = 1$?

$k = 3$?

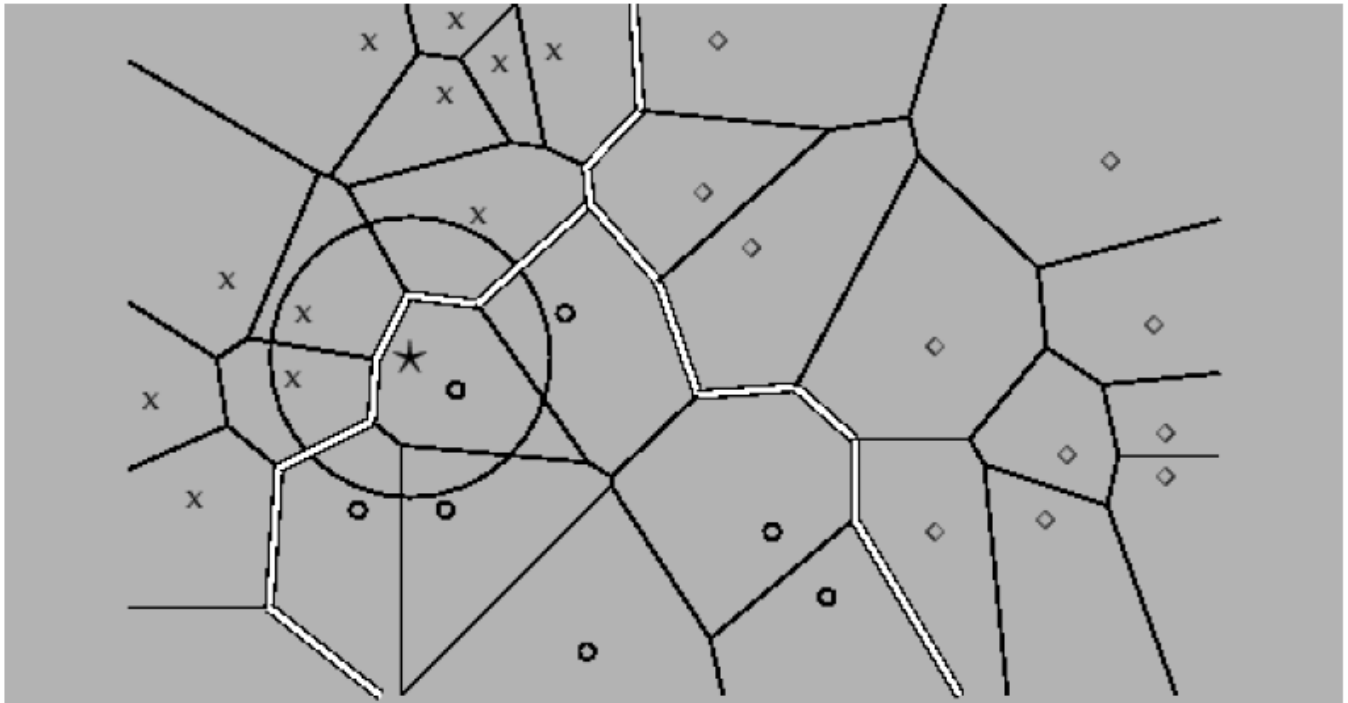
Voronoi tessellation

- Assuming $k = 1$: For a given set of objects in the space, let each object define a cell consisting of all points that are closer to that object than to other objects.
- Results in a set of convex polygons; so-called Voronoi cells.
- Decomposing a space into such cells gives us the so-called Voronoi tessellation.



- In the general case of $k \geq 1$, the Voronoi cells are given by the regions in the space for which the set of k nearest neighbors is the same.

Voronoi tessellation for 1NN

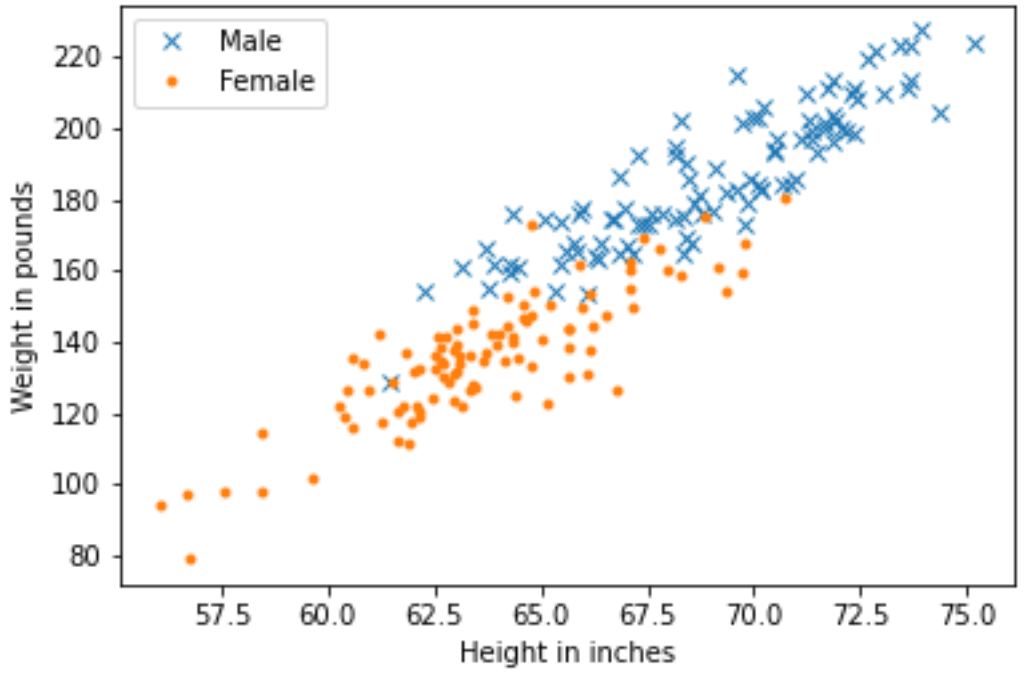


► **Figure 14.3** Voronoi tessellation and decision boundaries (double lines) in 1NN classification. The three classes are: X, circle and diamond.

Decision boundary for 1NN: defined along the regions of Voronoi cells for the objects in each class. Shows the non-linearity of kNN.

Example: Height, weight, gender

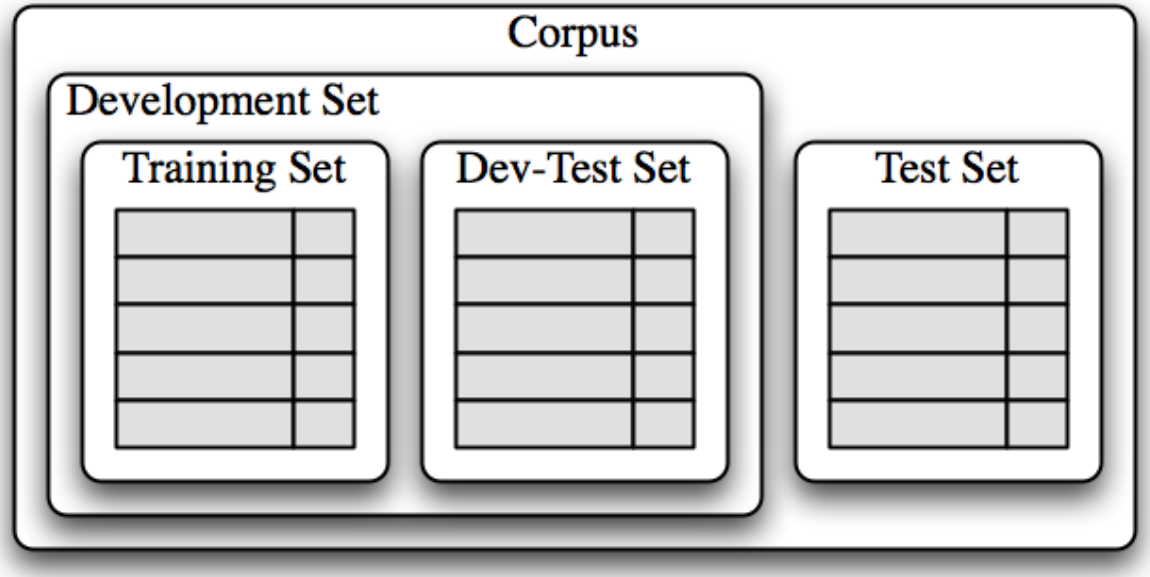
- Dataset:
  - 10,000 observations
  - 5,000 of each gender
  - Height in inches
  - Weight in pounds
- https://www.kaggle.com/mustafaali96/weight-height
- Processing:
  - Shuffled
  - Split:
    - 5000 for training
    - 2500 for development testing
    - 2500 for final testing



A random subset of 200 of the training data

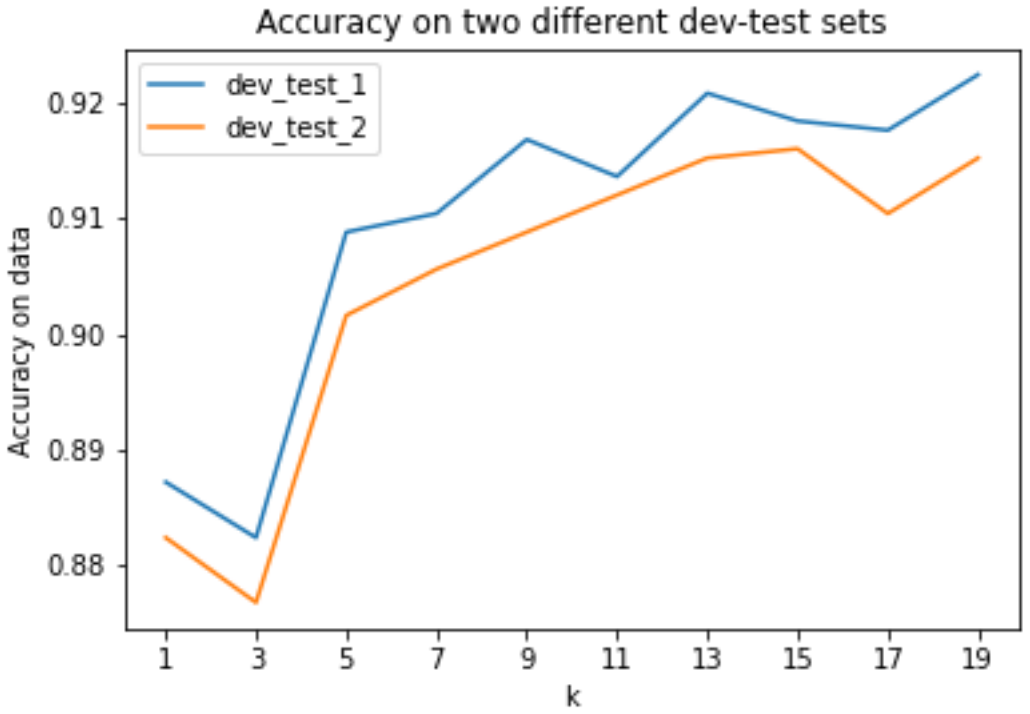
Choosing *k*

- For alternative values of *k*:
  - Train on the training set
  - Evaluate on the dev-test set
- Choose the *k* which yields the best accuracy
- You may test with this *k* on the final test set.



*k*NN with varying *k*

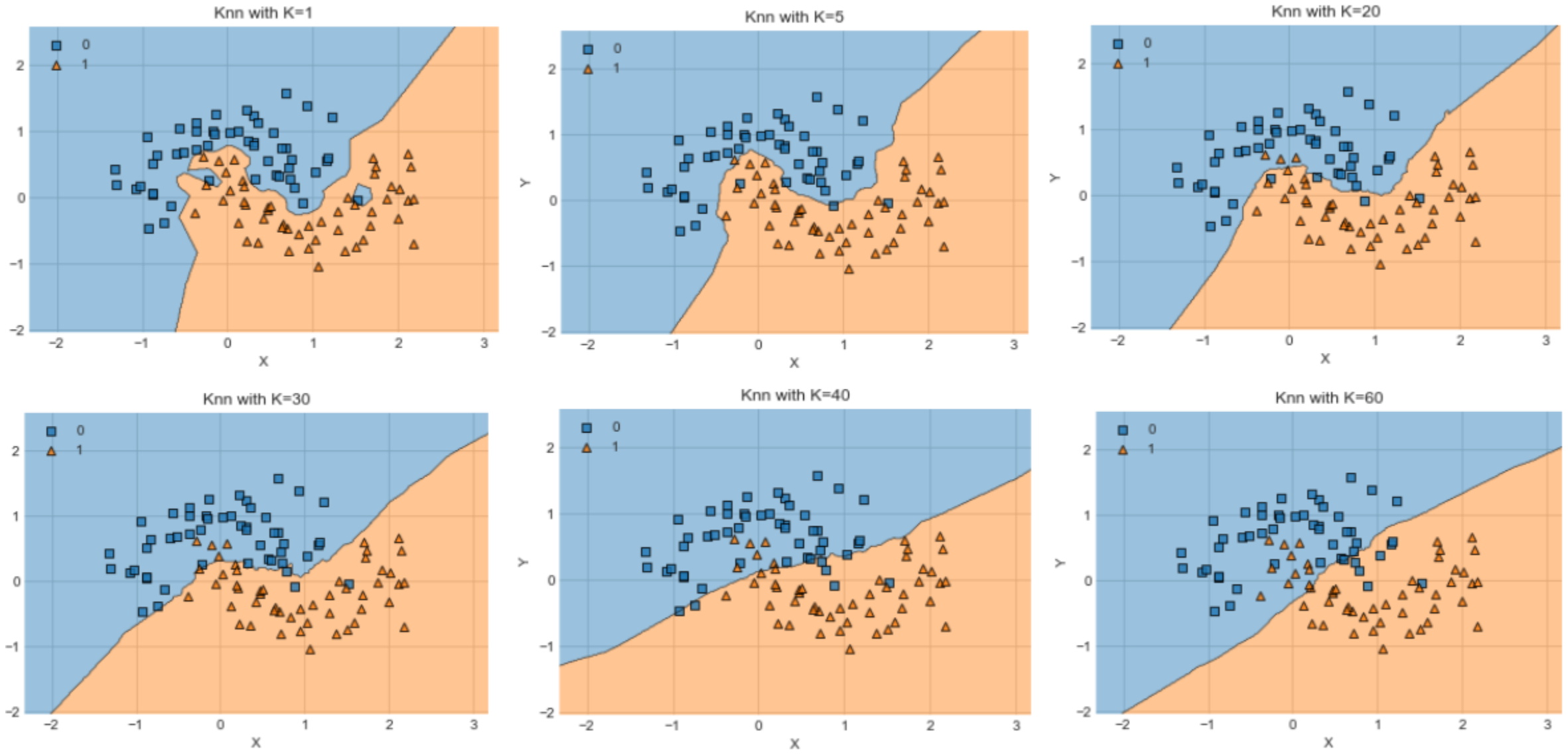
- Trained on the train set with various *k*-s
  - Accuracy on 2 different dev-test sets
  - For this task: better with larger *k*



Is larger *k* always better?

- Small *k*:
  - Good fit to training data
  - Danger of overfitting
- Larger *k*:
  - More general

- Next slide:
  - A different example task
  - The squares and triangles are the true classes
  - The background colors show the decision boundary for the classifier with various *k*-s

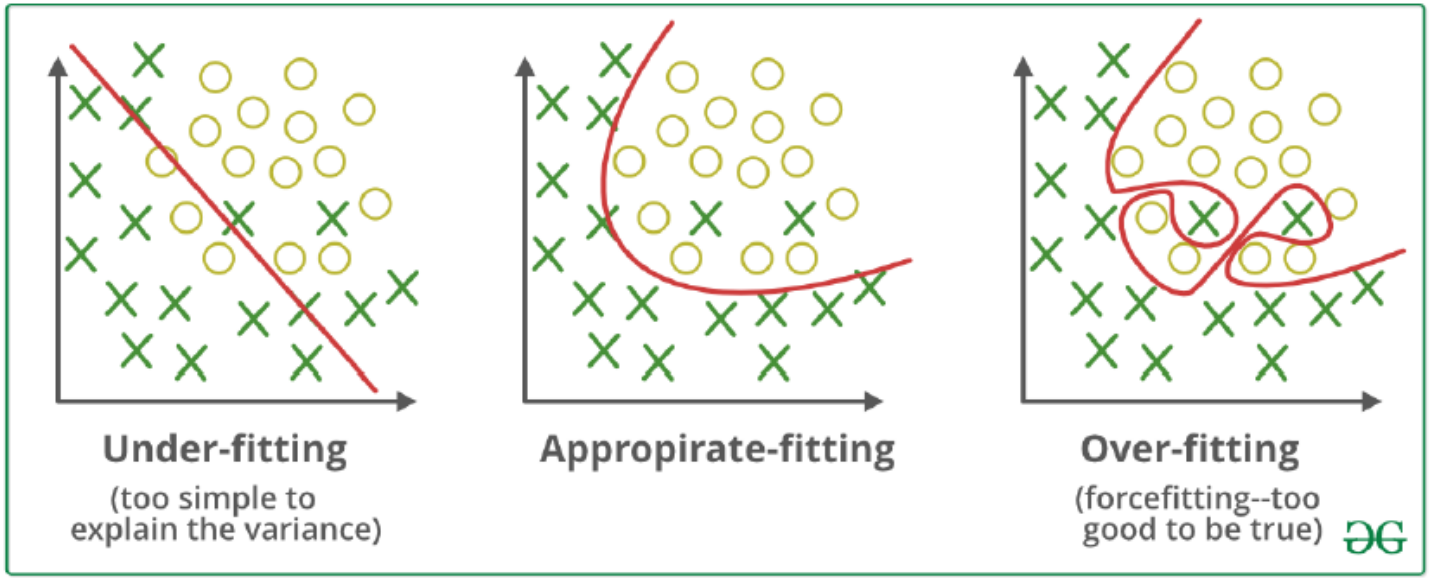


KNN visualization for the U-shaped dataset

https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6

Model fitting

- Overfitting: the model fits the training data closely but generalizes poorly, i.e. has many test errors.
- Underfitting: the model makes many errors even for the training data, i.e. it is unable to describe the data.

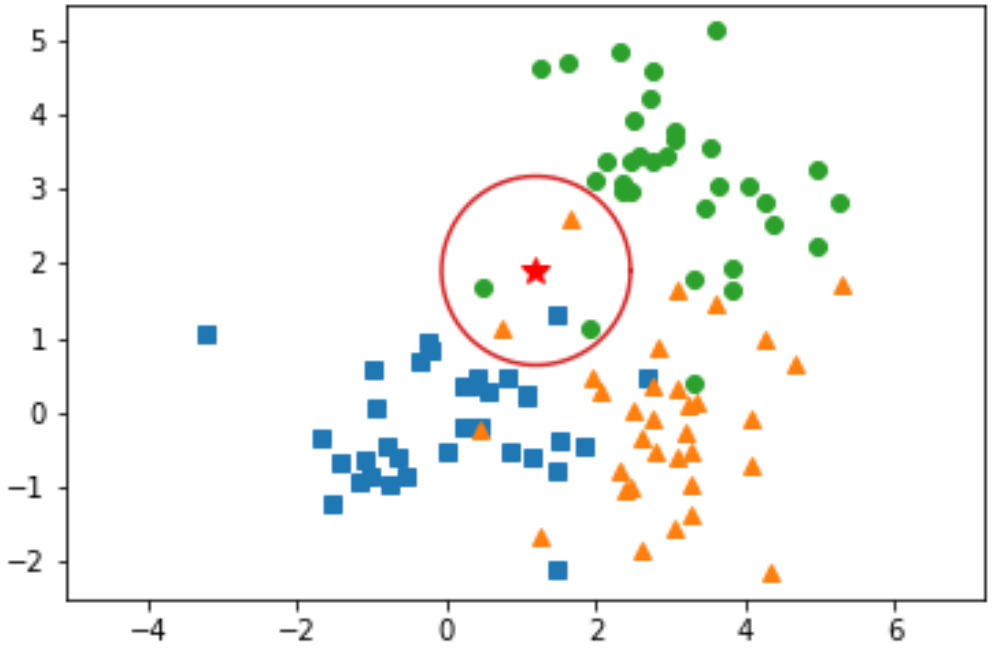


Bias–variance tradeoff

- The bias–variance tradeoff; the balancing of two sources of errors:
- Bias: the impact of assumptions in the model. Oversimplified models have high bias, leading to underfitting.
- Variance: sensitivity to changes in training data. Highly complex models may have high variance leading to overfitting.
- Example for kNN:
  - k=1: very high variance, very low bias.
  - k=N: very low variance, very high bias.

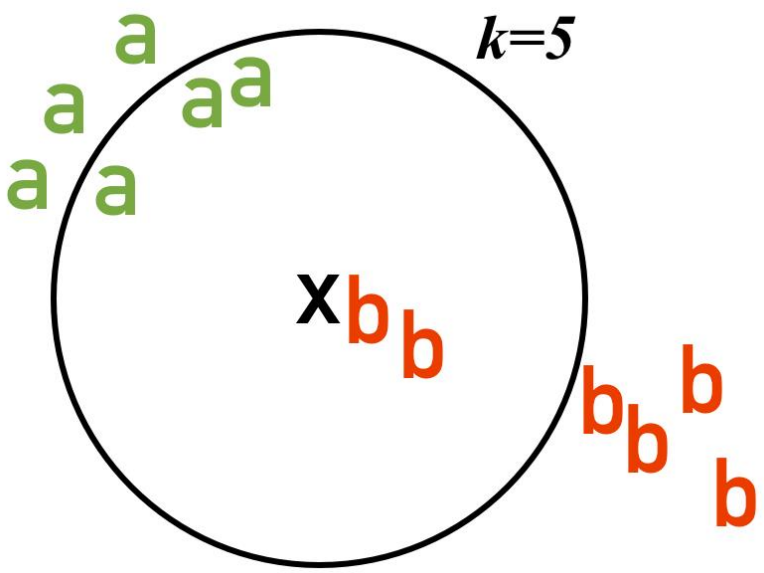
Footnote: More than two classes

- A binary classifier with odd *k* always reaches a decision
- With more than 2 classes, there might be a draw
- One possible way out
  1. Weight points by inverse distance from target,
  2. Sum weighted distances for each class
  3. Choose the class with largest weighted max.



Probabilistic *k*NN

- Sometimes, we are not interested in a hard decision,
  but rather the probability of an item belonging to a class
  - In particular if we are to combine this with other information
- *k*NN can be made probabilistic:
  - The probability of class c is the proportion of the *k* nearest neighbors in c.
- We may here also apply the weighting from last slide



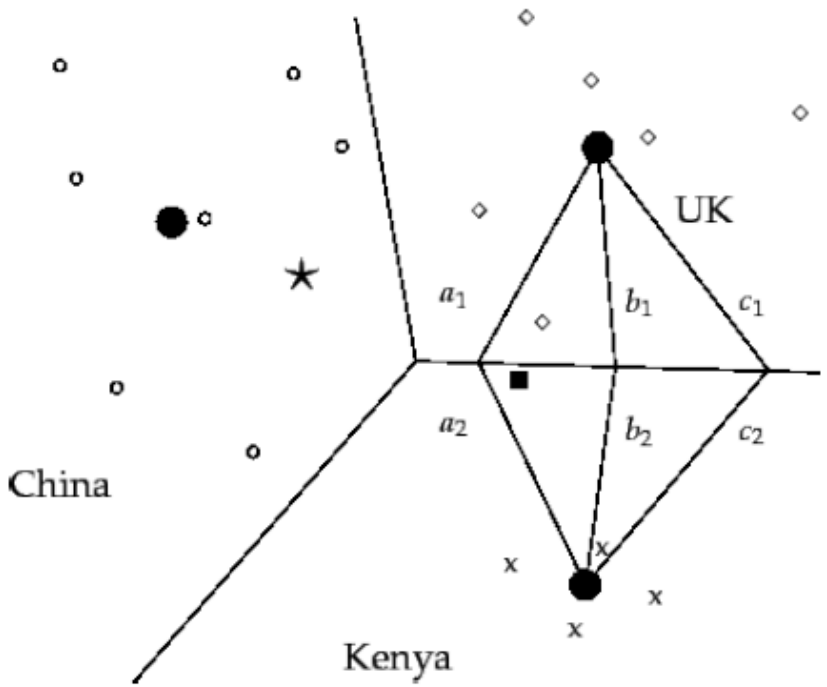
- $P(a|x) = \frac{3}{5}$
- $P(b|x) = \frac{2}{5}$

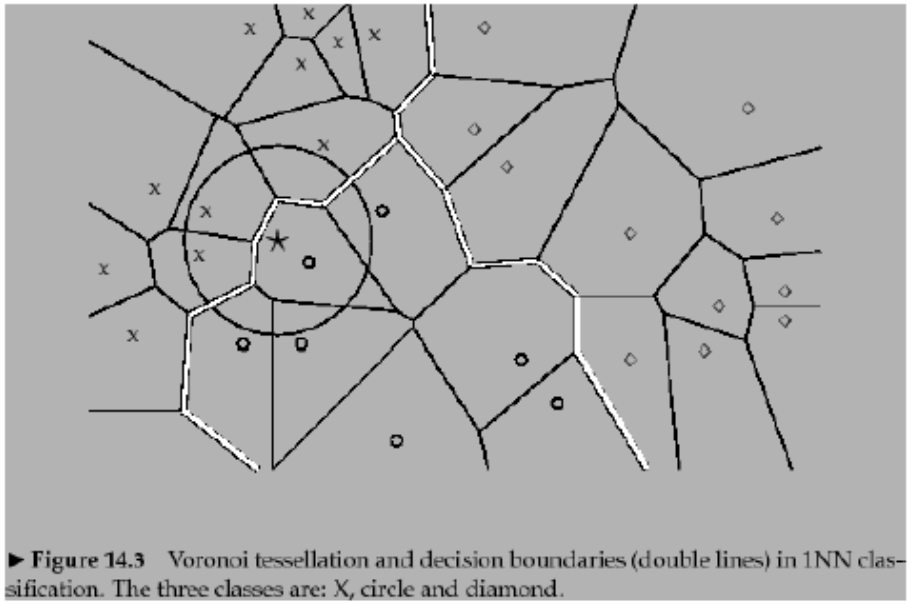
Properties of *k*NN

- Instance-based, no real training
  - it simply memorizes all training examples
  - Fast to "train"
- Inefficient in predicting the label of new instances
  - Since it must consider all the training data each time (= linear in the size of the training set)
- Notice the similarity to retrieving relevant documents for a given query: Both are instances of finding nearest neighbors.
- One parameter: *k*
- The distance measure may influence the result
- The scaling of the axes might influence the result

Connecting the dots

- Recall that the kNN decision boundary is locally linear for each cell in the Voronoi diagram.
- For Rocchio, we only use one example per class, i.e. the centroid, yielding a single Voronoi cell for each entire class.





► **Figure 14.3** Voronoi tessellation and decision boundaries (double lines) in 1NN classification. The three classes are: X, circle and diamond.

A practical note on feature extraction

- Before training and applying a classifier we first have to create the feature vectors to represent our data.
- Sometimes referred to as vectorization.
- The feature types (e.g. the BoW vocabulary) needs to be defined relative to the training set (including parameters like frequency cut-offs, idf-weights, etc).
- When vectorizing test data we must use the same features as in training.
- Vectorization therefore often done in two passes: first defining the feature set based on the training data, then creating the feature vectors. (*Fit* and *transform* in scikit-learn terminology)

Clarifications

- Syllabus: Manning, Raghavan & Schütze (2008), kapittel 14 frem til seksjon 14.3.1.
- Last week we included 14.4 "Linear versus nonlinear classifiers" on the last slide, but to learn more about that you have to attend
  - IN3050, a main topic
  - UN4080 explains why Naive Bayes can be considered a linear classifier

Next lecture

- Evaluating classifiers
- Unsupervised machine learning for class discovery: Clustering
- k-means clustering.
- Reading: *Manning, Raghavan & Schütze (2008)*, section 16, 16.1, 16.2, and 16.4 up until 16.4.1.

  J&M, seksjon 4.7–4.8