CSCI567 Machine Learning (Fall 2017)

Prof. Fei Sha

U of Southern California

Lecture on Aug. 22, 2017

Outline

1. About this Course
2. Overview of machine learning
3. Nearest neighbor classifier
4. Some practical sides of NNC
5. What we have learned

Outline

1. About this Course
   - Administration
   - Syllabus
2. Overview of machine learning
3. Nearest neighbor classifier
4. Some practical sides of NNC
5. What we have learned

Enrollment

- This course has *two* offerings: on-campus and DEN. You only need to *attend one*.
- This course has *two* sections: a lecture and a discussion section. You need to *attend both*.
- Given the current number of enrollments, auditing is *not* permitted.
- If you have not been able to enroll, you will have to wait for open slots. The department setups D-clearance policy.

Teaching staff

- Instructors:

  Lecture: Dr. Fei Sha

  Office: RTH 403

  Office Hours: By Appointments Only

  Discussion: Dr. Michael Shindler and Dr. Parisa Mansourifard

  Office: TBA

  Office Hours: TBA

- TAs:

  Soravit Changpinyo
  Wei-lun Chao
  Guangyu Li
  Zhiyun Lu
  Yimin Yan
  Ke Zhang

Required preparation

- Undergraduate courses in probability and statistics, linear algebra, multivariate calculus
- Programming: Python 3.0 and necessary packages, `git`
- Text processing: LaTeX

You are expected to complete coding assignments independently. This is not an introductory-level CS course thus there is no hand-on training of basic programming skills.

Online learning forums

Blackboard: `http://blackboard.usc.edu`

- Grades will be posted there.
- Do *not* use Blackboard for online discussion — we do *not* monitor discussion boards on it.

Piazza: `http://www.piazza.com`

- Online discussion forums will be setup in Piazza
- We will enroll you in Piazza once you have completed a quiz about the syllabus (Your grade on this quiz does not count toward to your grade for the course)

Textbook and course material

Lectures

Lecture slides will be posted after class — in some cases, we might also be able to post before lectures.

Textbooks

- *No required one*
- Strongly suggest you obtain *one* of the following
  - Machine Learning: A Probabilistic Perspective by Kevin Murphy
  - Elements of Statistical Learning by Hastie, Tibshirani and Friedman
- Optional and supplementary: please see syllabus for other reference pointers

Assigned readings to facilitate your understanding of the lectures

- We mark the corresponding chapters in the 2 books
- There can also be research papers, information from the web, etc.

Key dates in the schedule

- Quiz 1: 10/5
- Quiz 2: 10/31
- Quiz 3: 11/28

Homework assignments

- 5 Homework assignments, composed of
  - Algorithmic components
  - Programming components
- Policy
  - A total of 5 grace days for the semester, requiring to apply formally by filling a form.
  - Late assignments will be penalized
  - Working in group: each needs to write up and submit on his/her own
    We use sophisticated tools to check for potential violations in ethics. Please read and observe the academic honesty and integrity rules in the syllabus.
  - Submission process: you need to get a Github repo.
    You will need to get a Github handle before 8/25
    Other methods of submissions including emails are **not** *accepted*
  - Need to typeset your solutions. Learn how to use LaTeX

Quiz

- Three quizzes
- Length: about 1 hour each.
- You need to take those quizzes physically.

No make-up quiz unless with documented and unanticipated personal/medical urgency, and such reasons will be checked against the University's guidelines

Grade

- 40%: 5 homework assignments
- 60%: Quizzes

Re-grading policy

Only *factual errors* will be corrected.

- Homework and quizzes
  Within one week of releasing grades, submit a form in writing to apply for regrading – you need to be specific about which problem to be considered, and where the mistake in grading happens.
  All re-grading requests will be reviewed by the body of instructors and TAs together — please do not approach us individually for re-grading.
- The final grade
  Follow standard procedures by the University to appeal.
- Curved grading?
  We do not plan so.

Reasons that do not qualify for regrading

- I need to upgrade my grade to maintain/boost my GPA.
- I cannot graduate if my GPA is low or if I have failed this course.
- This is the last course I have taken before I graduate.
- I have done well in other courses / I am a great programmer/theoretician.
- I have a deadline prior to the homework/quiz due date.
- I have a regular job requiring a lot of my attention.
- Exam/homework are not the best way to show my competency in learning.

Rules

- Only factual errors in grading are basis for regrading.

Academic honesty and integrity

Plagiarism and other unacceptable violations

- Neither ethical nor in your self-interest.
- We use software as well as manually examine to detect possible plagiarism.
- Zero-tolerance, *one strike out!*

Things to note specifically

- Cite relevant and related work (published or unpublished)
- Acknowledge others' help
- *Please study the related sections in the syllabus on academic honesty and integrity – a quiz is coming up!*

Teaching philosophy

The nature of this course

- Describe basic concepts and tools
- Describe algorithms and their development with intuition and rigor

Expectation on you

- Hone skills on grasping abstract concepts and thinking critically to solve problems with machine learning techniques
- Solidify your knowledge with hand-on programming assignments
- Prepare you for studying advanced machine learning techniques

Important things for you to do

- Syllabus was just updated as of today. Syllabus (schedules and topics to be covered) will be adjusted, if necessary, as the semester progresses.
- Get a Github handle before 8/25
- Brush up your Python 3.0 programming skills
- Study the syllabus, especially the academic honesty and integrity rules. A quiz is coming up

Any questions?

Outline

1 About this Course

2 Overview of machine learning
- What is machine learning?

3 Nearest neighbor classifier

4 Some practical sides of NNC

5 What we have learned

What is machine learning?

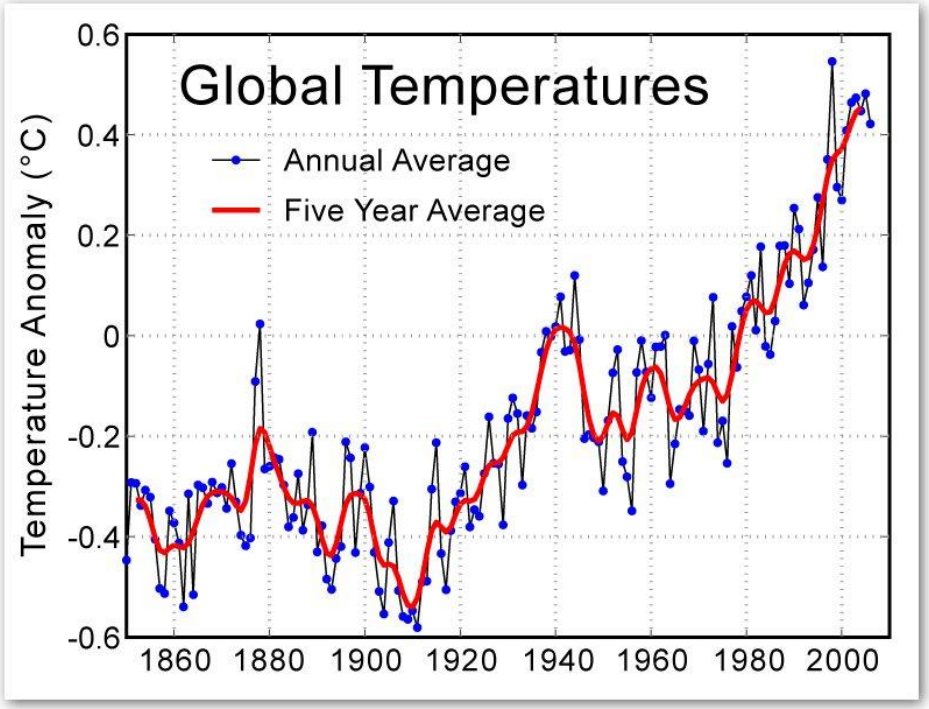
One possible definition[1]

a set of methods that can automatically *detect patterns* in data, and then use the uncovered patterns to *predict future data*, or to perform other kinds of decision making *under uncertainty*

cf. Murphy's book

Example: detect patterns

How the temperature has been changing in the last 140 years?

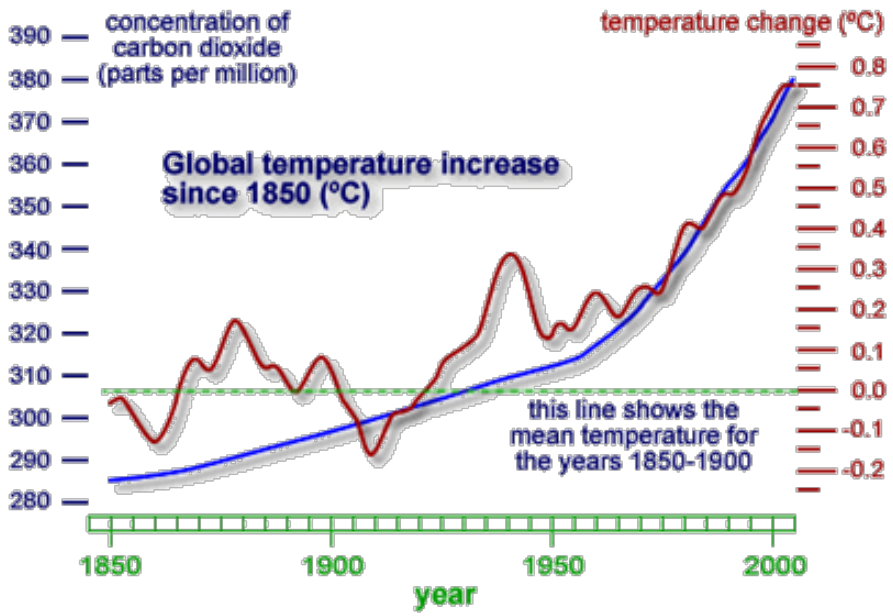


Patterns

- Seems going up
- there seems to be repeated periods of going up and down.

How do we describe the pattern?

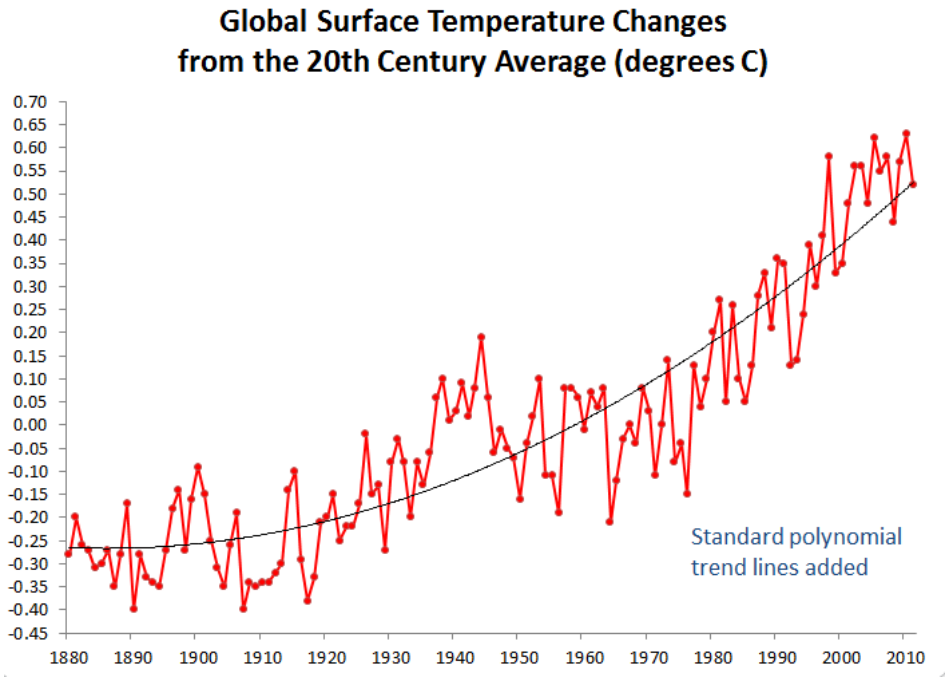
Build a model: fit the data with a polynomial function



- The model is not accurate for individual years
- But collectively, the model captures the major trend (for instance, we are not able to model the pattern of the *repeated up and down* with this polynomial function)

Predicting future

What is temperature of 2010?



- Again, the model is not accurate for that specific year
- But then, it is close to the actual one

What we have learned from this example?

Key ingredients in the machine learning task

- Data
  collected from past observation (we often call them *training data*)
- Modeling
  devised to capture the patterns in the data
  - The model does not have to be true — as long as it is close, it is useful
  - We should tolerate randomness and mistakes — many interesting things are stochastic by nature.
- Prediction
  apply the model to forecast what is going to happen in future

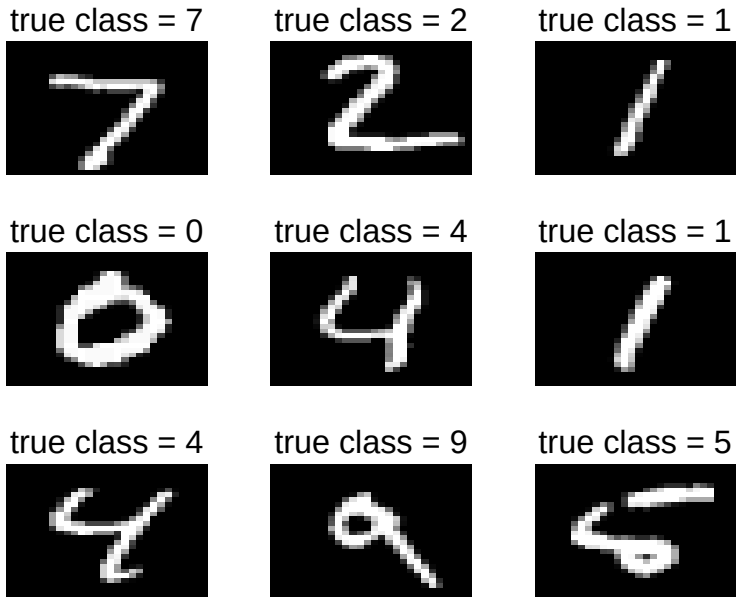
A rich history of applying statistical learning methods

Recognizing flowers (by R. Fisher, 1936)

Types of Iris: setosa, versicolor, and virginica

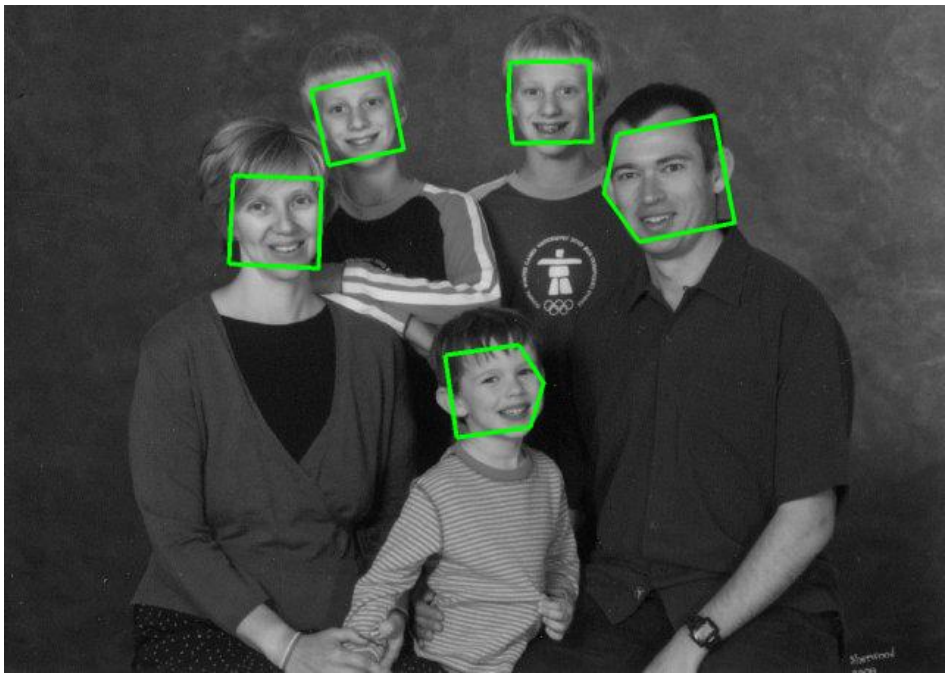
Huge success 20 years ago

Recognizing handwritten zipcodes and cheques (AT&T Labs, circa late 1990s)



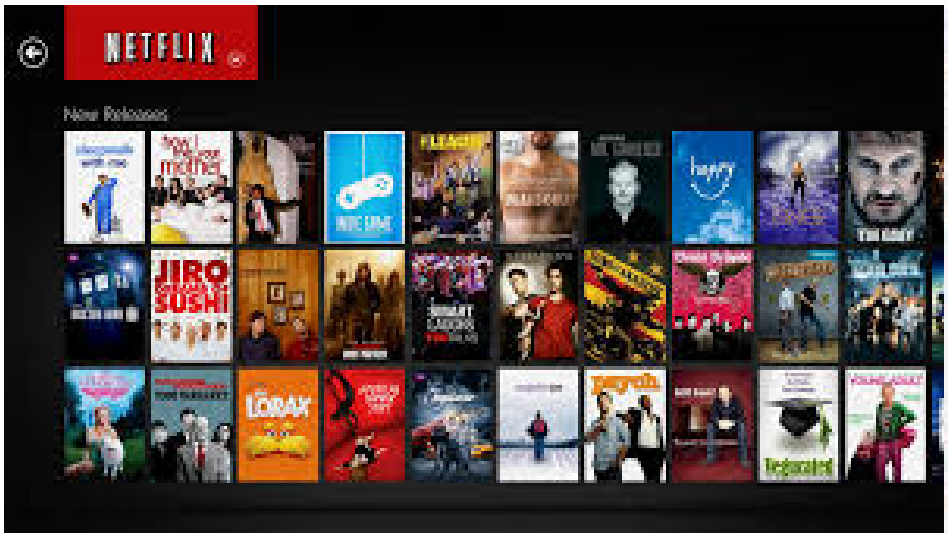
More modern ones, in your social life

Recognizing your friends on Facebook



It might be possible to know about you than yourself

Recommending what you might like



Why is machine learning so hot?

- Tons of consumer applications:
  - speech recognition, information retrieval and search, email and document classification, stock price prediction, object recognition, biometrics, etc
  - Highly desirable expertise from industry: Google, Facebook, Microsoft, Uber, Twitter, IBM, Linkedin, Amazon, $\cdots$
- Enable scientific breakthrough
  - Climate science: understand global warming cause and effect
  - Biology and genetics: identify disease-causing genes and gene networks
  - Social science: social network analysis; social media analysis
  - Business and finance: marketing, operation research
  - Emerging ones: healthcare, energy, $\cdots$

What is in machine learning?

Different flavors of learning problems

- Supervised learning
  Aim to predict (as in the previous list of applications)
- Unsupervised learning
  Aim to discover hidden and latent patterns and explore data
- Reinforcement learning
  Aim to act optimally under uncertainty
- Many other paradigms

The focus and goal of this course

- Supervised learning (before quiz 1)
- Unsupervised learning (after quiz 1)

Outline

1. About this Course
2. Overview of machine learning
3. Nearest neighbor classifier
   - Intuitive example
   - General setup for classification
   - Algorithm
   - How to measure performance of a classifier?
4. Some practical sides of NNC
5. What we have learned

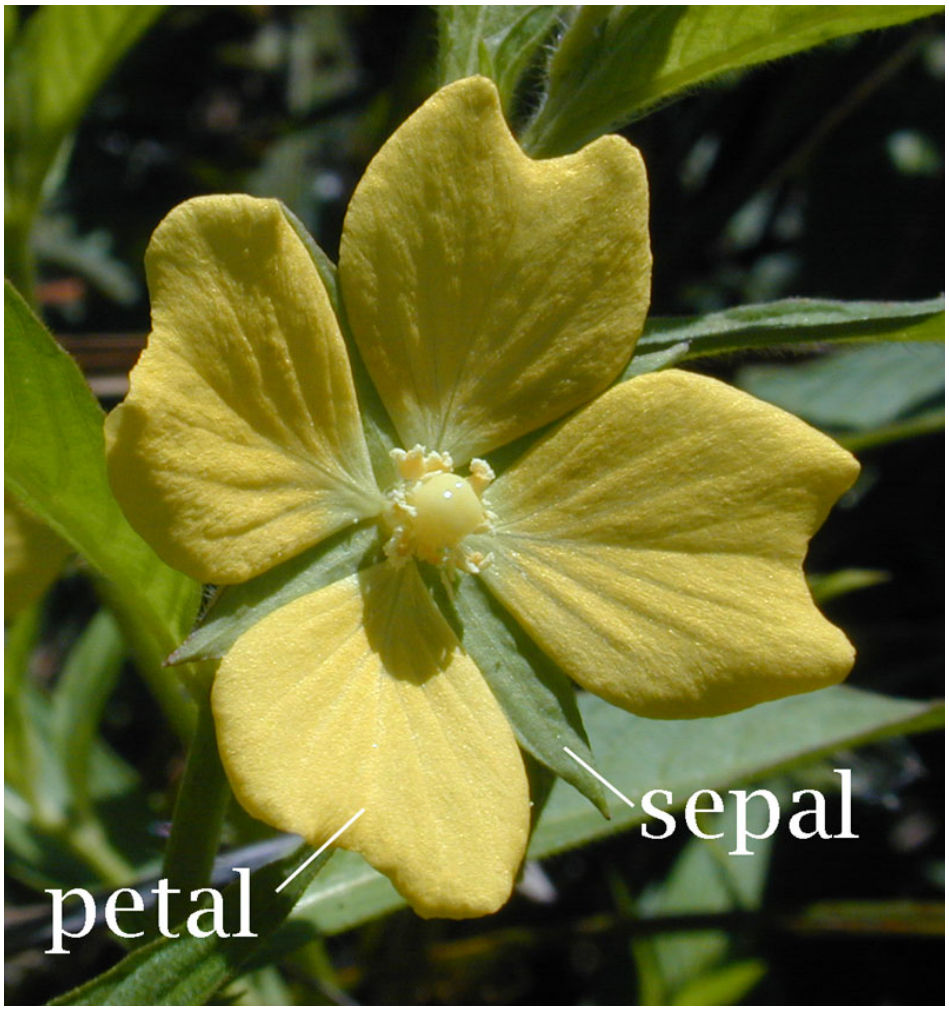
Recognizing flowers

Types of Iris: setosa, versicolor, and virginica



Measuring the properties of the flowers

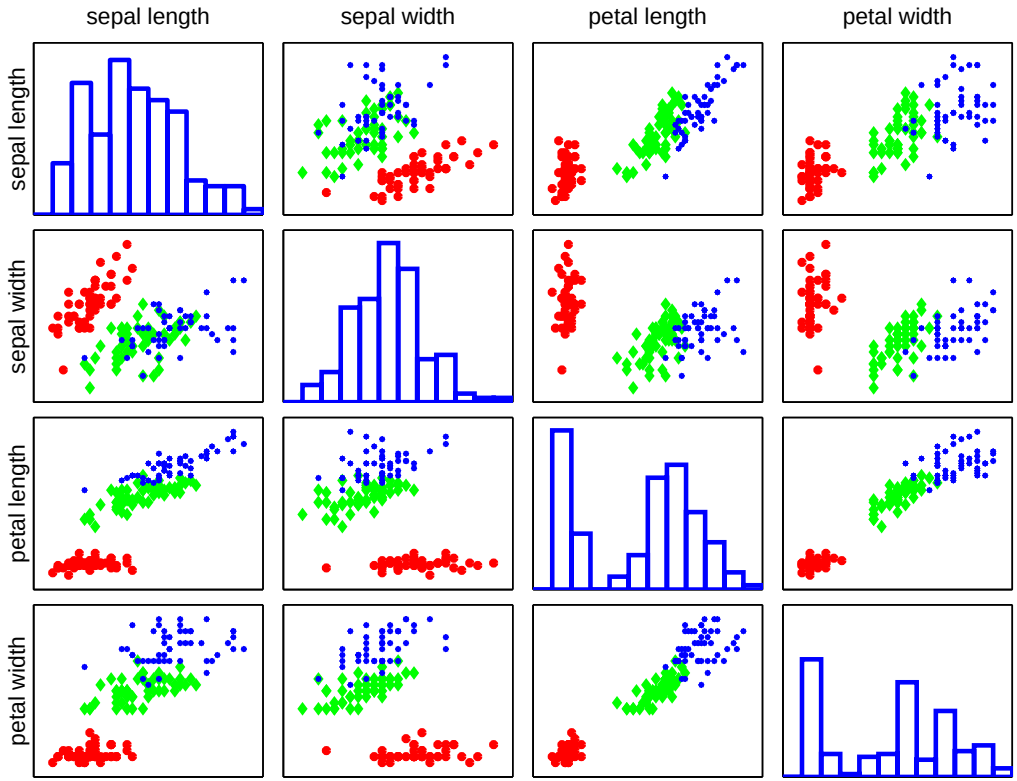
Features and attributes: the widths and lengths of sepal and petal



Pairwise scatter plots of 131 flower specimens

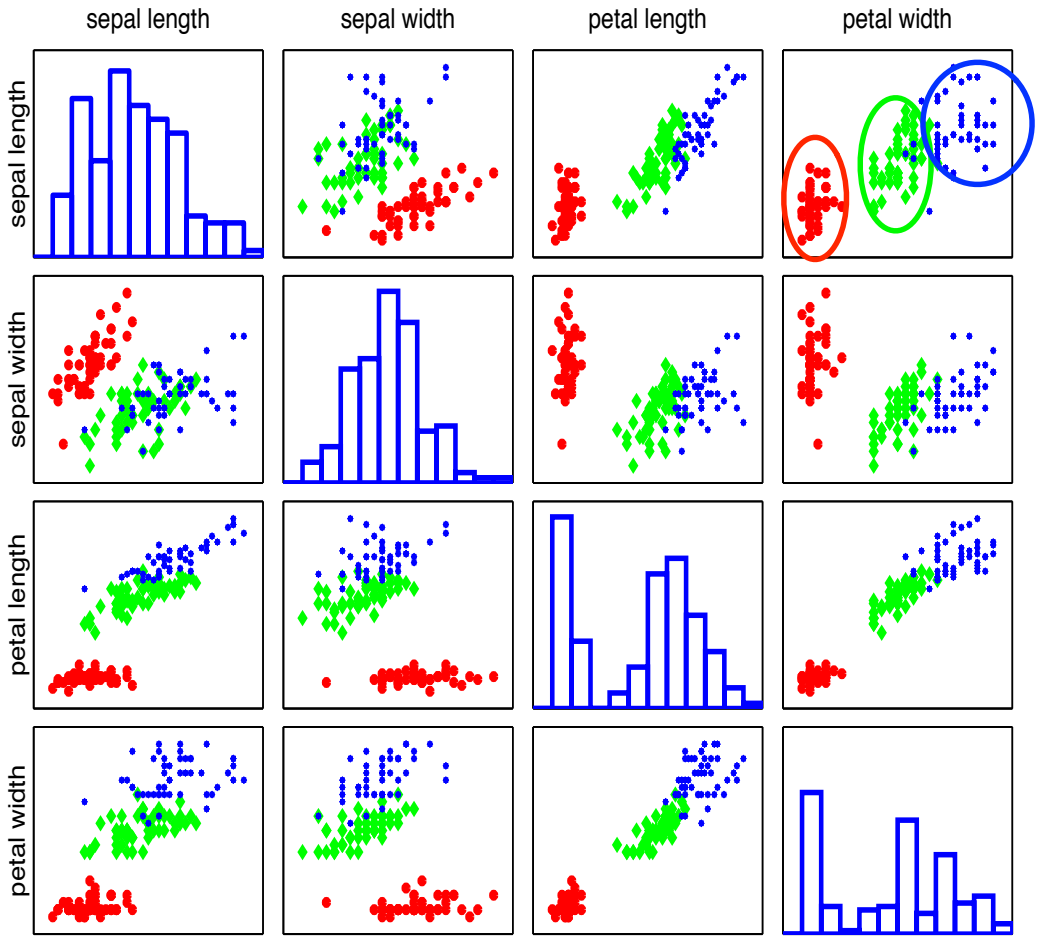
Visualization of data helps to identify the right learning model to use

Each colored point is a flower specimen: setosa, versicolor, virginica



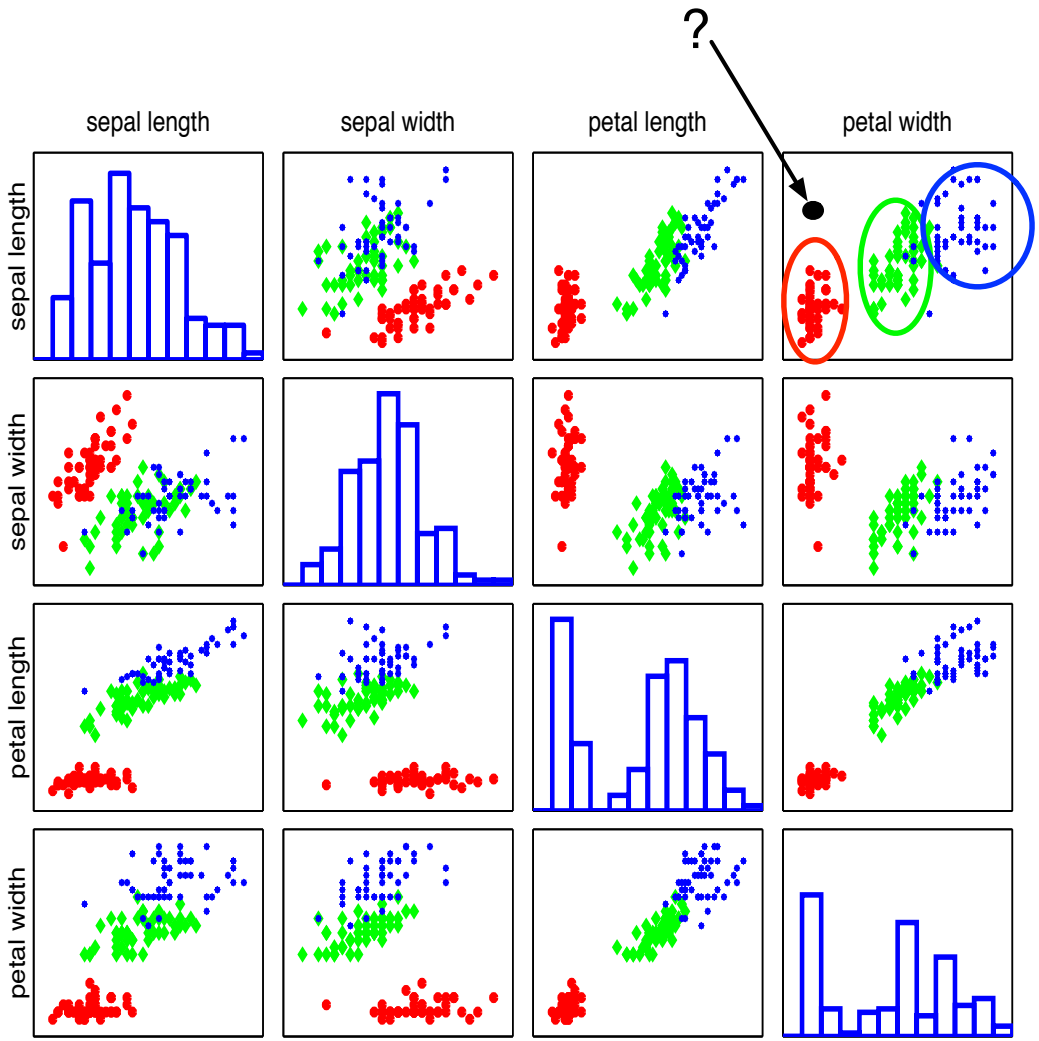
Different types seem well-clustered and separable

Using two features: petal width and sepal length



Labeling an unknown flower type

Closer to red cluster: so labeling it as setosa



Multi-class classification

Classify data into one of the multiple categories

- Input (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Output (label): $y \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$
- Learning goal: $y = f(\boldsymbol{x})$

Special case: binary classification

- Number of classes: $\mathsf{C} = 2$
- Labels: $\{0, 1\}$ or $\{-1, +1\}$

More terminology

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{N}}, y_{\text{N}})\}$
- They are used for learning $f(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{M}}, y_{\text{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Training data and test data should *not* overlap: $\mathcal{D}^{\text{TRAIN}} \cap \mathcal{D}^{\text{TEST}} = \emptyset$

Note that. In real life when the model is used, we do not know what the ground-truth is – we would take whatever the model predicts as if it were the truth

Often, data is conveniently organized as a table

Ex: Iris data (click here for all data)

- 4 features
- 3 classes

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*
4.9	3.1	1.5	0.1	*I. setosa*

Nearest neighbor classification (NNC)

Nearest neighbor

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathsf{nn}(\boldsymbol{x})}$$

where $\mathsf{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$, i.e., the index to one of the training instances,

$$\mathsf{nn}(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2 = \arg\min_{n \in [\mathsf{N}]} \sqrt{\sum_{d=1}^{\mathsf{D}} (x_d - x_{nd})^2}$$

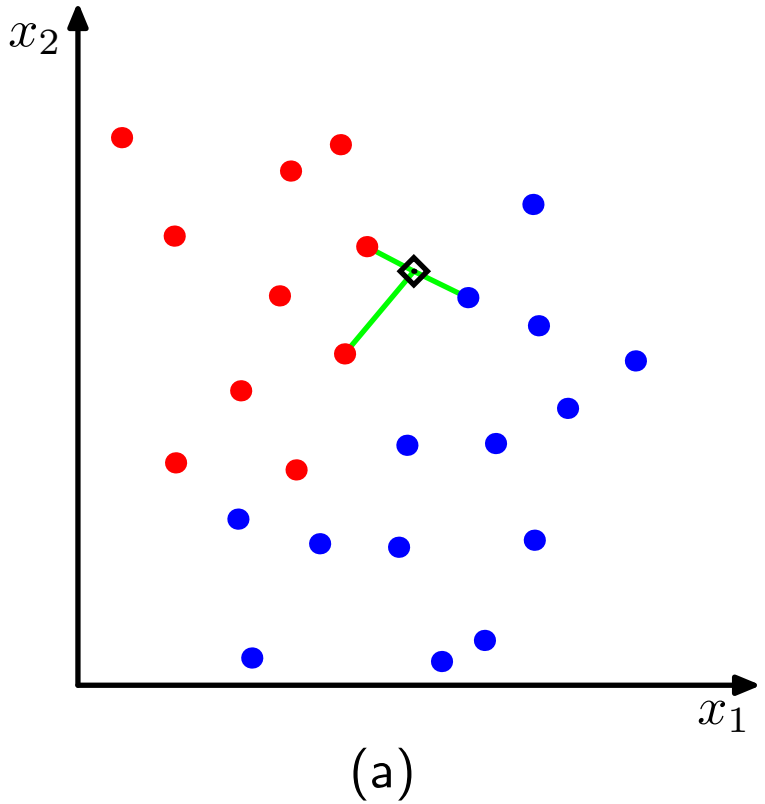
Classification rule

$$y = f(\boldsymbol{x}) = y_{\mathsf{nn}(\boldsymbol{x})}$$

N.B. $\arg\min$ *means finding the argument (in this case* n*) that minimizes the function/expression.* $\| \ \|_2$ *means Euclidean norm, ie, 2-norm, which calculates the distances between two points.*

Visual example

In this 2-dimensional example, the nearest point to $\boldsymbol{x}$ is a red training instance, thus, $\boldsymbol{x}$ will be labeled as red.



(a)

Example: classify Iris with two features

Training data

ID (n)	petal width (x_1)	sepal length (x_2)	category (y)
1	0.2	5.1	setoas
2	1.4	7.0	versicolor
3	2.5	6.7	virginica
⋮	⋮	⋮	

Flower with unknown category

petal width $= 1.8$ and sepal width $= 6.4$

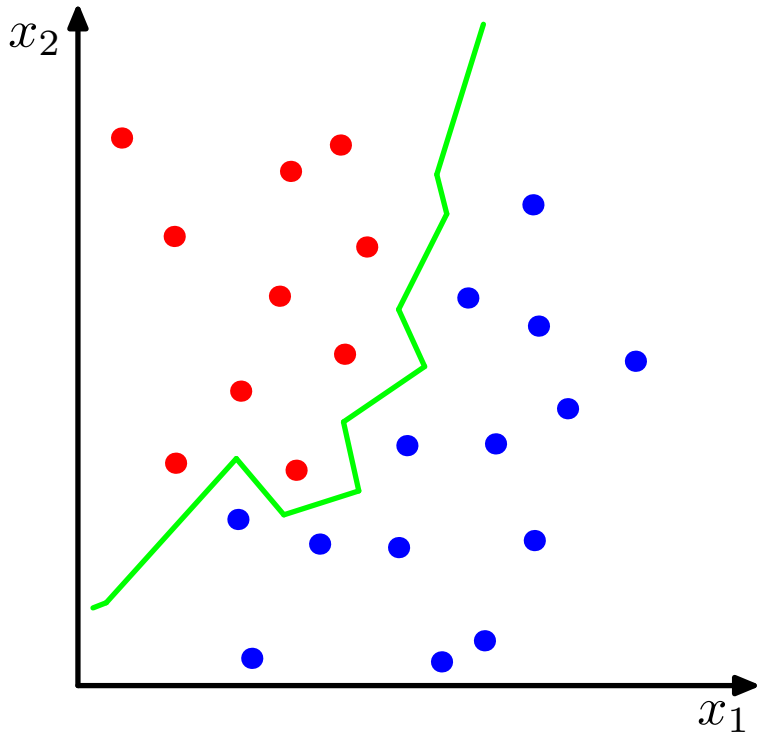
Calculating distance $= \sqrt{(x_1 - x_{n1})^2 + (x_2 - x_{n2})^2}$

ID	distance
1	1.75
2	0.72
3	0.76

Thus, the category is *versicolor* (the real category is *viriginica*)

Decision boundary

For every point in the space, we can determine its label using the NNC rule. This gives rise to a *decision boundary* that partitions the space into different regions.



(b)

How to measure nearness with other distances?

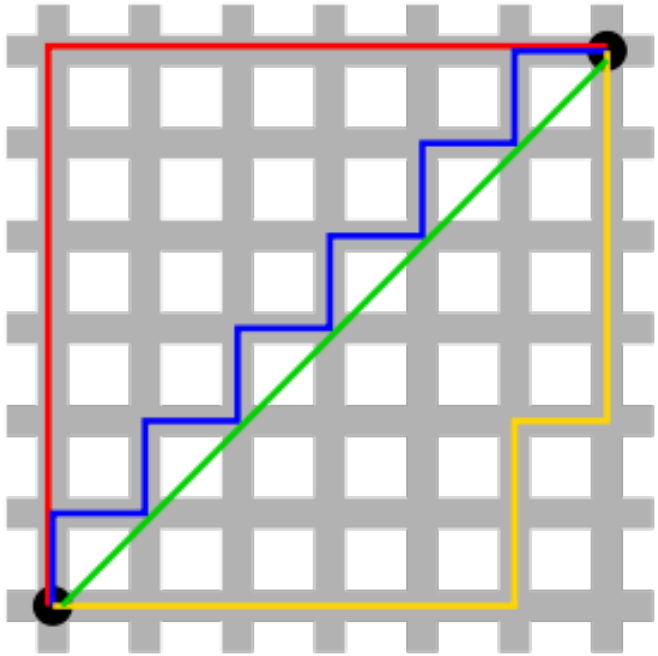
Previously, we use the Euclidean distance

$$\mathsf{nn}(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$$

We can also use alternative distances

E.g., the following L_1 distance (i.e., city block distance, or Manhattan distance)

$$\begin{aligned}\mathsf{nn}(\boldsymbol{x}) &= \arg\min_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_1 \\ &= \arg\min_{n \in [\mathsf{N}]} \sum_{d=1}^{\mathsf{D}} |x_d - x_{nd}|\end{aligned}$$



Green line is Euclidean distance. Red, Blue, and Yellow lines are L_1 distance

K-nearest neighbor (KNN) classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\mathsf{nn}_1(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 2nd-nearest neighbor: $\mathsf{nn}_2(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}] - \mathsf{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 3rd-nearest neighbor: $\mathsf{nn}_2(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}] - \mathsf{nn}_1(\boldsymbol{x}) - \mathsf{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$

The set of K-nearest neighbor

$$\mathsf{knn}(\boldsymbol{x}) = \{\mathsf{nn}_1(\boldsymbol{x}), \mathsf{nn}_2(\boldsymbol{x}), \cdots, \mathsf{nn}_K(\boldsymbol{x})\}$$

Let $\boldsymbol{x}(k) = \boldsymbol{x}_{\mathsf{nn}_k(\boldsymbol{x})}$, then

$$\|\boldsymbol{x} - \boldsymbol{x}(1)\|_2^2 \leq \|\boldsymbol{x} - \boldsymbol{x}(2)\|_2^2 \cdots \leq \|\boldsymbol{x} - \boldsymbol{x}(K)\|_2^2$$

How to classify with K neighbors?

Classification rule

- Every neighbor votes: suppose y_n (the true label) for $\boldsymbol{x}_n$ is c, then
  - vote for c is 1
  - vote for $c' \neq c$ is 0

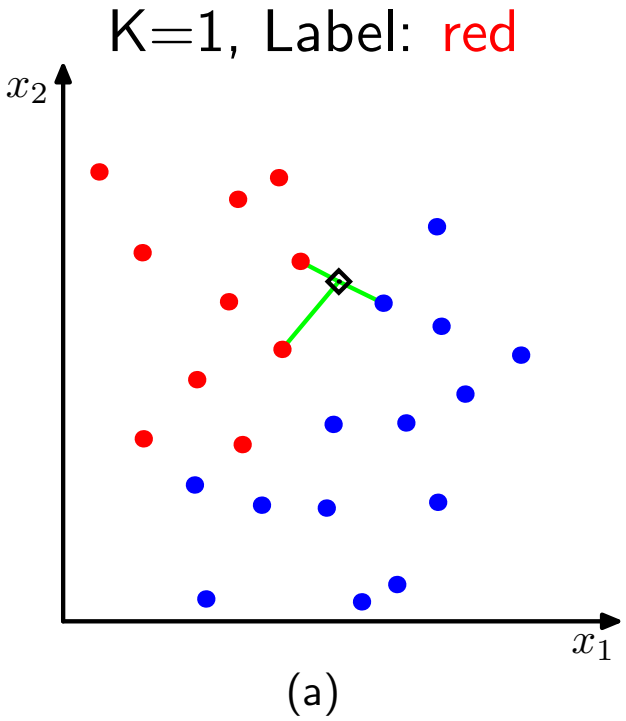
  We use the *indicator function* $\mathbb{I}(y_n == c)$ to represent.
- Aggregate everyone's vote on a class label c

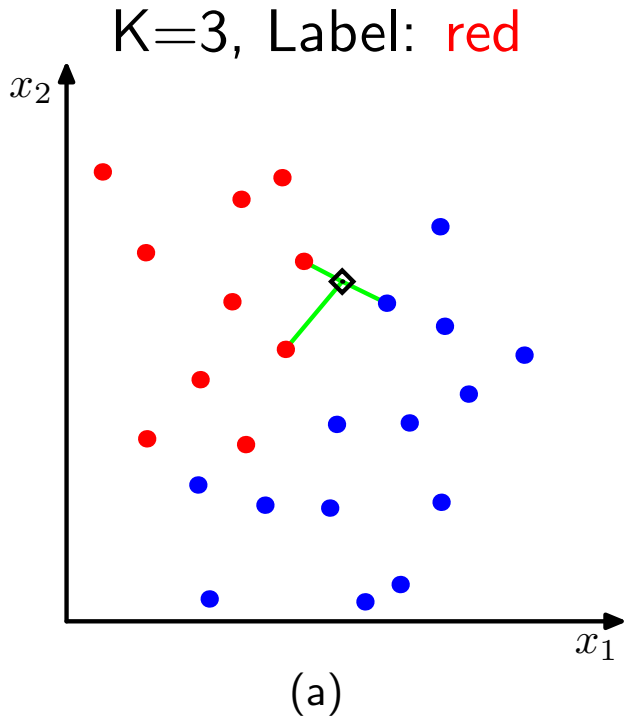
$$v_c = \sum_{n \in \text{knn}(\boldsymbol{x})} \mathbb{I}(y_n == c), \quad \forall \quad c \in [\text{C}]$$

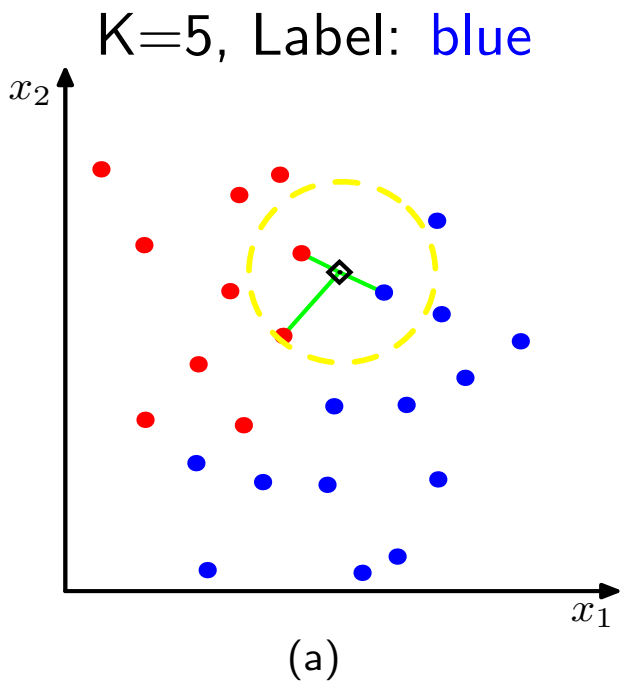
- Label with the majority

$$y = f(\boldsymbol{x}) = \arg\max_{c \in [\text{C}]} v_c$$

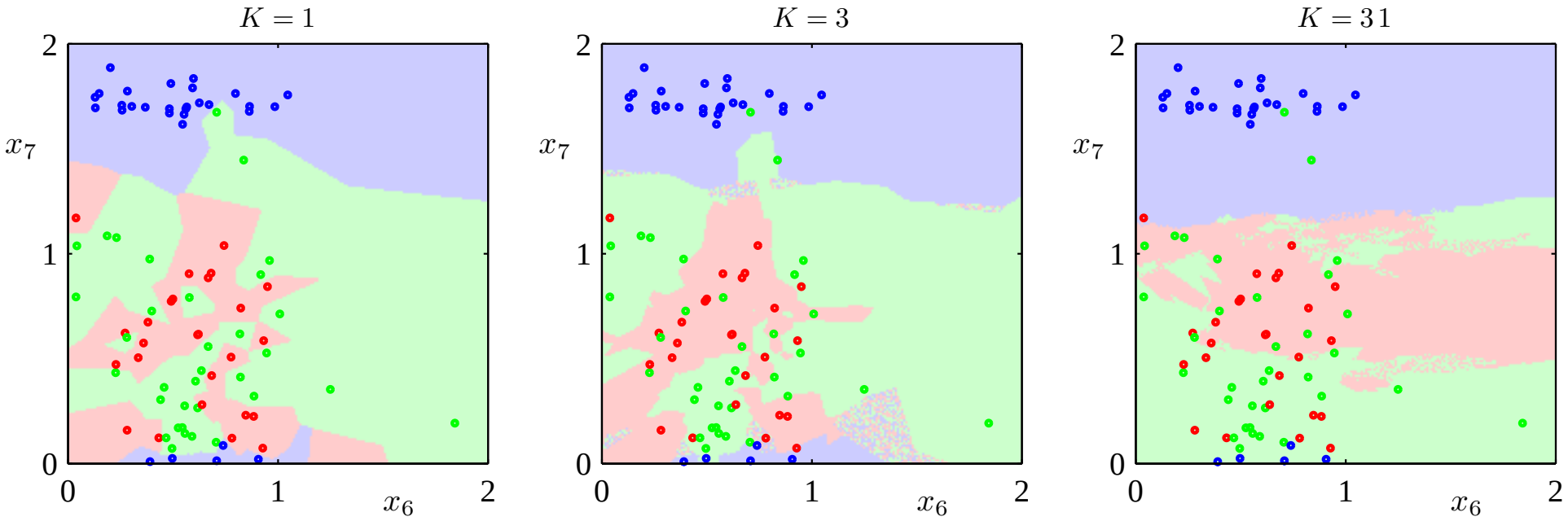
Example







How to choose an optimal K?



When K increases, the decision boundary becomes smooth.

Is NNC doing the right thing for us?

Intuition

We should compute accuracy — the percentage of data points being correctly classified, or the error rate — the percentage of data points being incorrectly classified.

Two versions: which one to use?

- Defined on the training data set

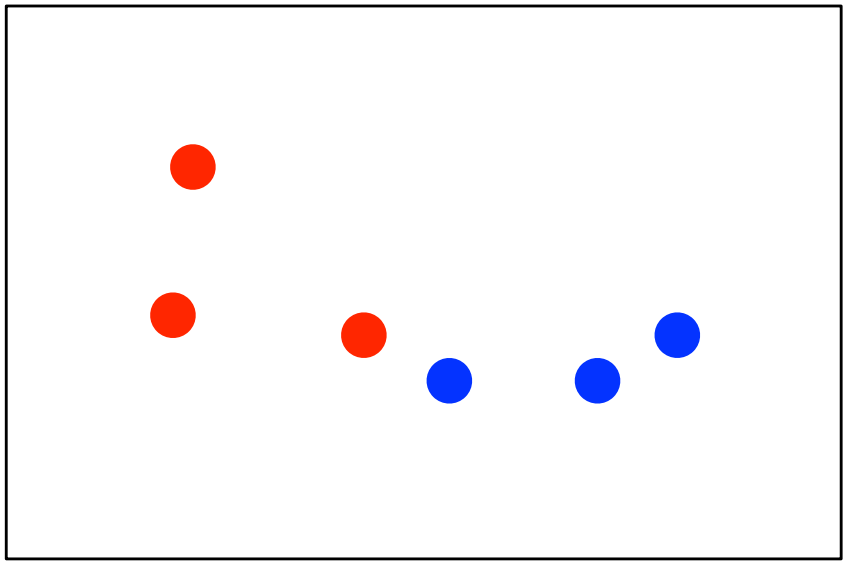
$$A^{\text{TRAIN}} = \frac{1}{\mathsf{N}} \sum_{n} \mathbb{I}[f(\boldsymbol{x}_n) == y_n], \quad \varepsilon^{\text{TRAIN}} = \frac{1}{\mathsf{N}} \sum_{n} \mathbb{I}[f(\boldsymbol{x}_n) \neq y_n]$$

- Defined on the test (evaluation) data set

$$A^{\text{TEST}} = \frac{1}{\mathsf{M}} \sum_{m} \mathbb{I}[f(\boldsymbol{x}_m) == y_m], \quad \varepsilon^{\text{TEST}} = \frac{1}{\mathsf{M}} \sum_{M} \mathbb{I}[f(\boldsymbol{x}_m) \neq y_m]$$

Example

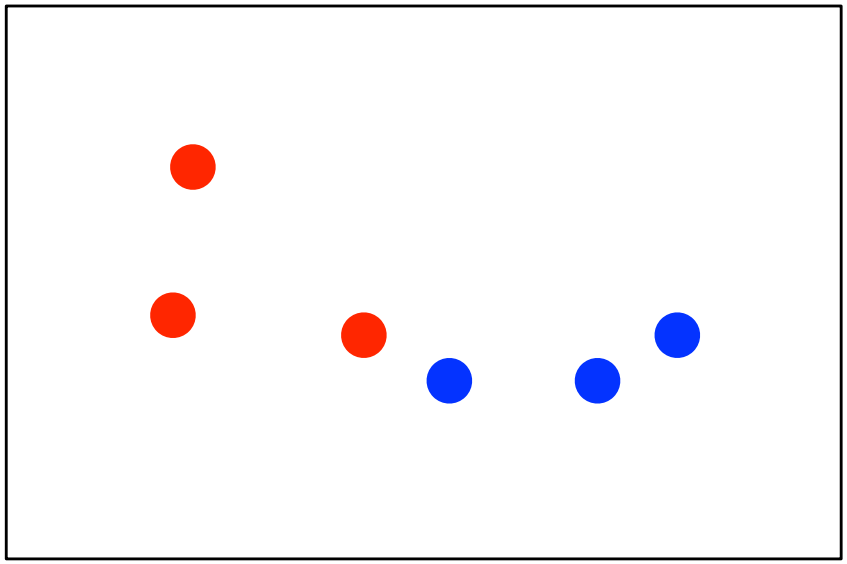
Training data



What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

Example

Training data

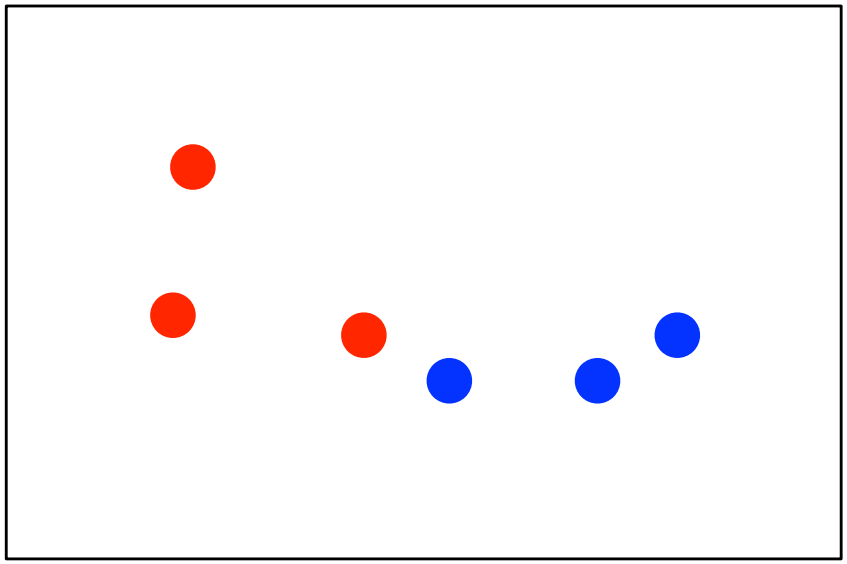


What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

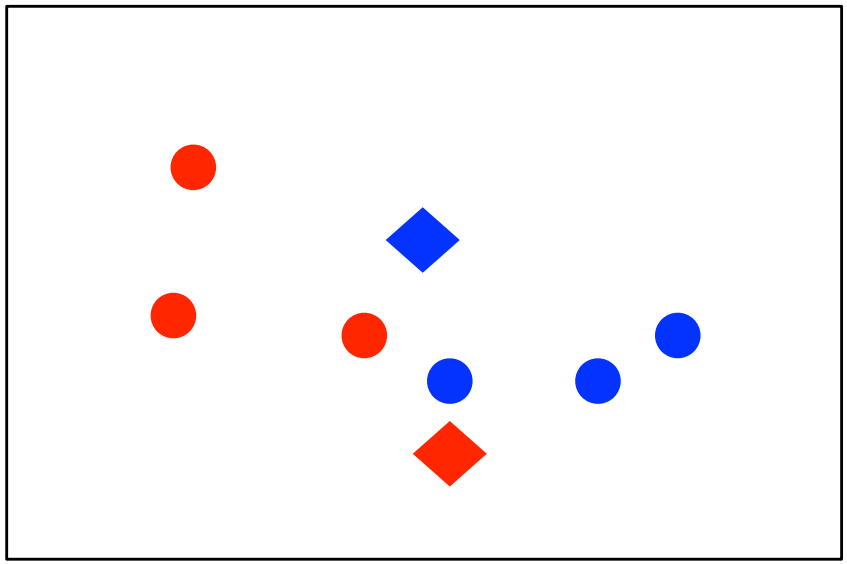
$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

Example

Training data



Test data



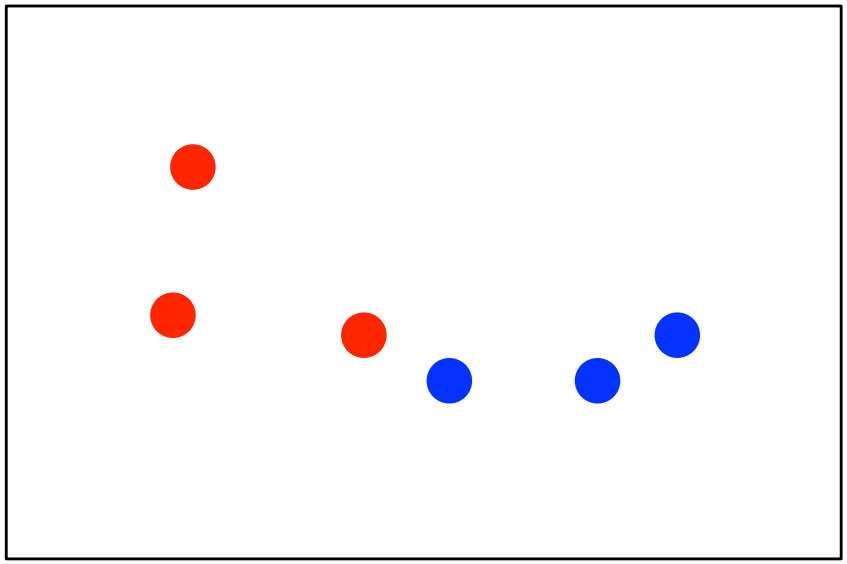
What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

What are A^{TEST} and $\varepsilon^{\text{TEST}}$?

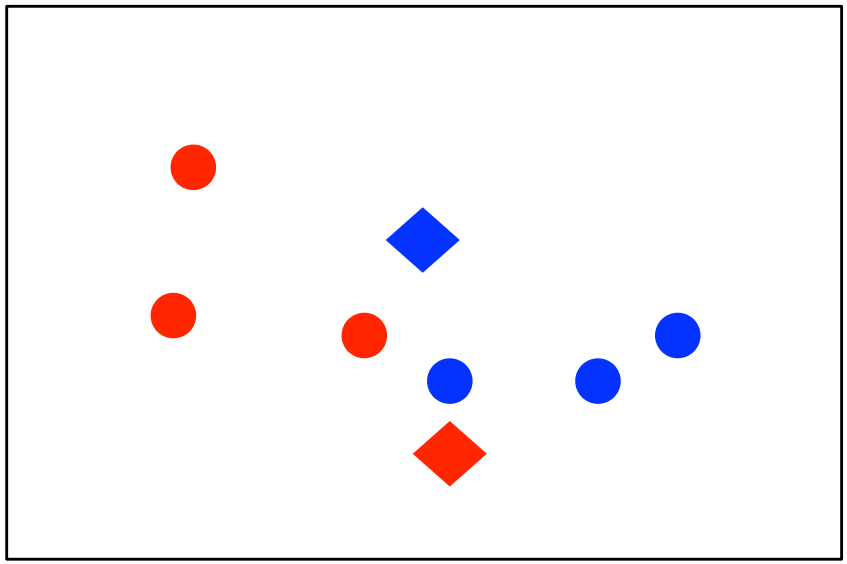
$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

Example

Training data



Test data



What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

What are A^{TEST} and $\varepsilon^{\text{TEST}}$?

$$A^{\text{TEST}} = 0\%, \quad \varepsilon^{\text{TEST}} = 100\%$$

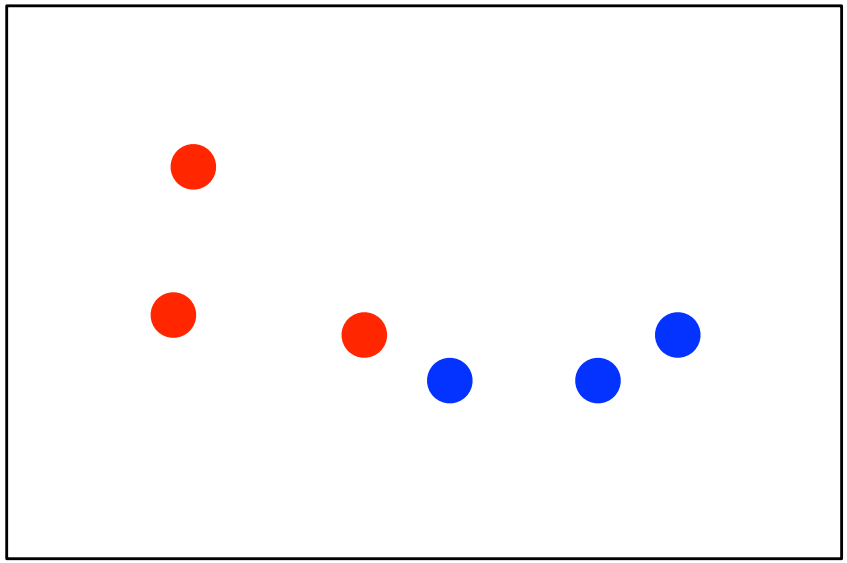
NB. In this case, for every training data point, its nearest neighbor in the training dataset is itself.

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will *not be itself*. So the error rate would not become 0 necessarily.

Training data



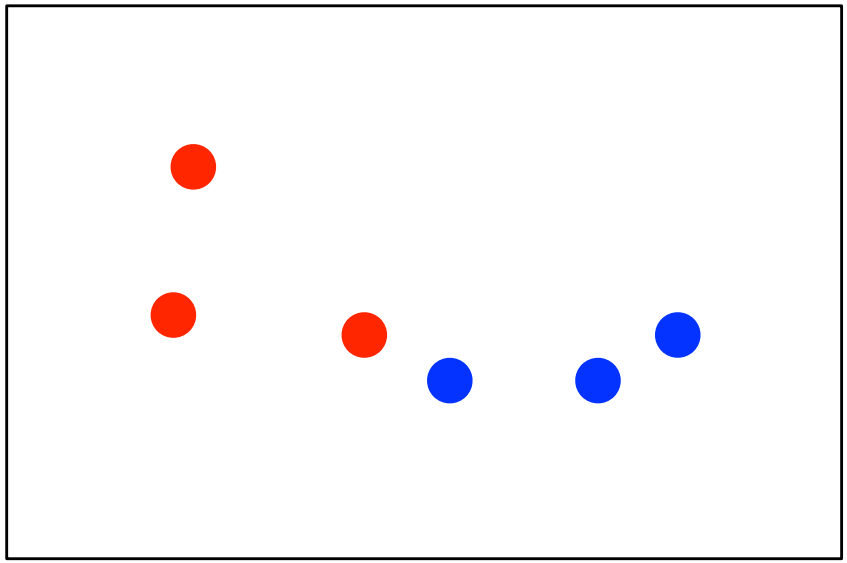
What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will *not be itself*. So the error rate would not become 0 necessarily.

Training data



What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 66.67\% (\text{i.e.}, 4/6)$$
$$\varepsilon^{\text{TRAIN}} = 33.33\% (\text{i.e.}, 2/6)$$

Outline

1. About this Course
2. Overview of machine learning
3. Nearest neighbor classifier
4. Some practical sides of NNC
   - How to tune to get the best out of it?
   - Preprocessing data
5. What we have learned

Hypeparameters in NNC

Two practical issues about NNC

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance), for example, from the following generalized distance measure

$$\|\boldsymbol{x} - \boldsymbol{x}_n\|_p = \left(\sum_d |x_d - x_{nd}|^p \right)^{1/p}$$

for $p \geq 1$.

Those are not specified by the algorithm itself — resolving them requires empirical validation and are task/dataset-specific.

Tuning by using a validation dataset

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{N}}, y_{\text{N}})\}$
- They are used for learning $f(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{M}}, y_{\text{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Development (or validation) data

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{L}}, y_{\text{L}})\}$
- They are used to optimize hyperparameter(s).

Training data, validation and test data should *not* overlap!

Recipe

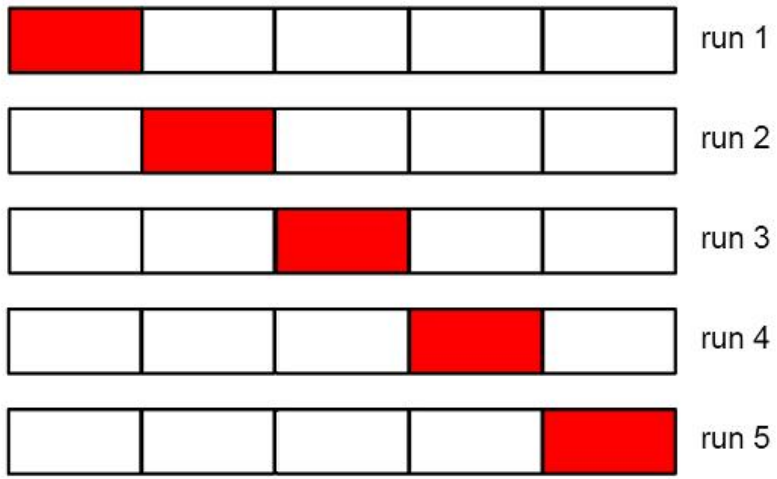
- for each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$
  - Evaluate the performance of the model on $\mathcal{D}^{\text{DEV}}$
- Choose the model with the best performance on $\mathcal{D}^{\text{DEV}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Cross-validation

What if we do not have validation data?

- We split the training data into S equal parts.
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that *on average*, the model performing the best

S $= 5$: 5-fold cross validation



Special case: when S = N, this will be leave-one-out.

Recipe

- Split the training data into S equal parts. Denote each part as $\mathcal{D}_s^{\text{TRAIN}}$
- for each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - for every $s \in [1, \text{S}]$
    - Train a model using $\mathcal{D}_{\setminus s}^{\text{TRAIN}} = \mathcal{D}^{\text{TRAIN}} - \mathcal{D}_s^{\text{TRAIN}}$
    - Evaluate the performance of the model on $\mathcal{D}_s^{\text{TRAIN}}$
  - Average the S performance metrics
- Choose the hyperparameter corresponding to the best averaged performance
- Use the best hyerparamter to train on a model using all $\mathcal{D}^{\text{TRAIN}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Yet, another practical issue with NNC

Distances depend on units of the features!

(Show how proximity can be changed due to change in features' scale; Draw on screen or blackboard)

Preprocess data

Normalize data so that the data look like from a normal distribution

- Compute the means and standard deviations in each feature

$$\bar{x}_d = \frac{1}{\mathrm{N}} \sum_n x_{nd}, \qquad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Many other ways of normalizing data — you would need/want to try different ones and pick them using (cross)validation

Mini-summary

Advantages of NNC

- Computationally, simple and easy to implement – just computing the distance
- Theoretically, has strong guarantees "doing the right thing"

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(\mathrm{N}D)$ for labeling a (unseen) data point
- We need to "carry" the training data around. Without it, we cannot do classification. This type of method is called *nonparametric*.
- Choosing the right distance measure and K can be involved.

Outline

1 About this Course

2 Overview of machine learning

3 Nearest neighbor classifier

4 Some practical sides of NNC

5 What we have learned

Summary so far

- Described a simple learning algorithm called Nearest Neighbor Classification
  - Used intensively in practical applications — you will get a taste of it in your homework
  - Discussed a few practical aspects, such as tuning hyperparameters, with (cross)validation