ATM4171 Introduction to Machine Learning for Atmospheric and Earth System Research

Kai Puolamäki, kai.puolamaki@helsinki.fi (mailto:kai.puolamaki@helsinki.fi)

11-22 May 2020

This Jupyter Notebook conains slides. I will use only part of them during the lectures. I was planning to incorporate material to this slideset during the course, i.e., you might want to re-download it every day.

In [1]:

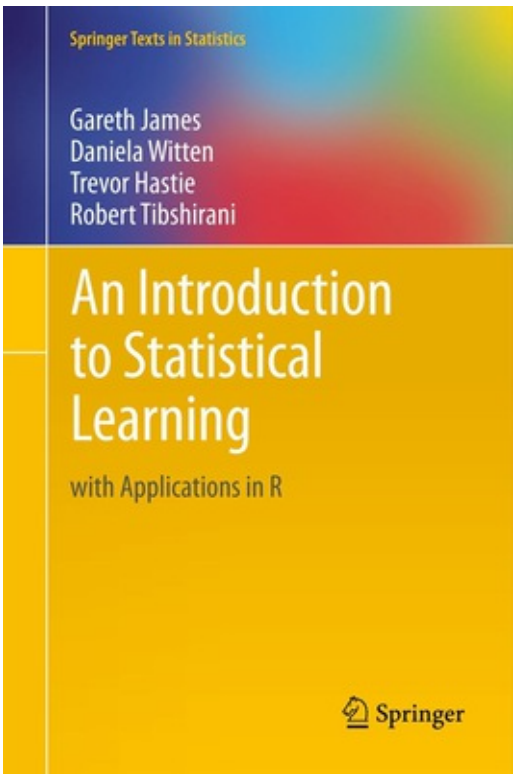
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import pystan
import matplotlib.pyplot as plt
import arviz as az
import seaborn as sns
from sympy import *
import random
import statsmodels.api as sm

import time

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

## We don't really use Keras in this course except during the 1st lecture.
## The stupid TensorFlow generates lots of warnings. :(
#from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

```
Using TensorFlow backend.
/usr/local/anaconda3/lib/python3.7/site-packages/tensorflow/python/fra
mework/dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will b
e understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorflow/python/fra
mework/dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will b
e understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorflow/python/fra
mework/dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will b
e understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorflow/python/fra
mework/dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will b
e understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorflow/python/fra
mework/dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will b
e understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorflow/python/fra
mework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will b
e understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

```
/usr/local/anaconda3/lib/python3.7/site-packages/tensorboard/compat/te
nsorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, i
t will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorboard/compat/te
nsorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, i
t will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorboard/compat/te
nsorflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, i
t will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorboard/compat/te
nsorflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, i
t will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorboard/compat/te
nsorflow_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, i
t will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/anaconda3/lib/python3.7/site-packages/tensorboard/compat/te
nsorflow_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, i
t will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:

```
np.random.seed(42)
```



Staff

- Kai Puolamäki
- Anna Shcherbacheva
- Please use ml-inar2020@helsinki.fi (mailto:ml-inar2020@helsinki.fi) for all email correspondance!
- We'll mostly be using Moodle and Slack for communications
- For organisation see https://moodle.helsinki.fi/mod/page/view.php?id=1883181 (https://moodle.helsinki.fi/mod/page/view.php?id=1883181)

Related courses

- Advanced Course in Machine Learning (period IV)
- Statistical Data Science (period II)
- Computational Statistics I-II (periods I-II)
- Probabilistic Graphical Models (period III)
- Introduction to Data Science (period I)
- Deep Learning (period II)
- Many seminars also have a strong machine learning flavour!

Contents of the course

- Ingredients of machine learning
  - What is machine learning? (overview)
  - tasks, model, data, statistics, algorithms, optimization
  - evaluation of algorithms
- Supervised learning
  - regression, classification
- Unsupervised learning
  - clustering, dimensionality reduction

Contents of the course (another view)

- tasks
  - supervised learning (regression & classification)
  - unsupervised learnign (clustring & dimensionality reduction)
- models
  - linear models
  - probabilistic modelling
  - algorithmic models (kNN, tree-based models etc.)
- statistics and evaluation
  - how to evaluate errors and confidence
- algorithms and optimization
  - greedy algorithms
  - tree algorithms

Elements of ML task

- Approach 1: download some software, run, have results!
- Approach 2: understand what you are doing

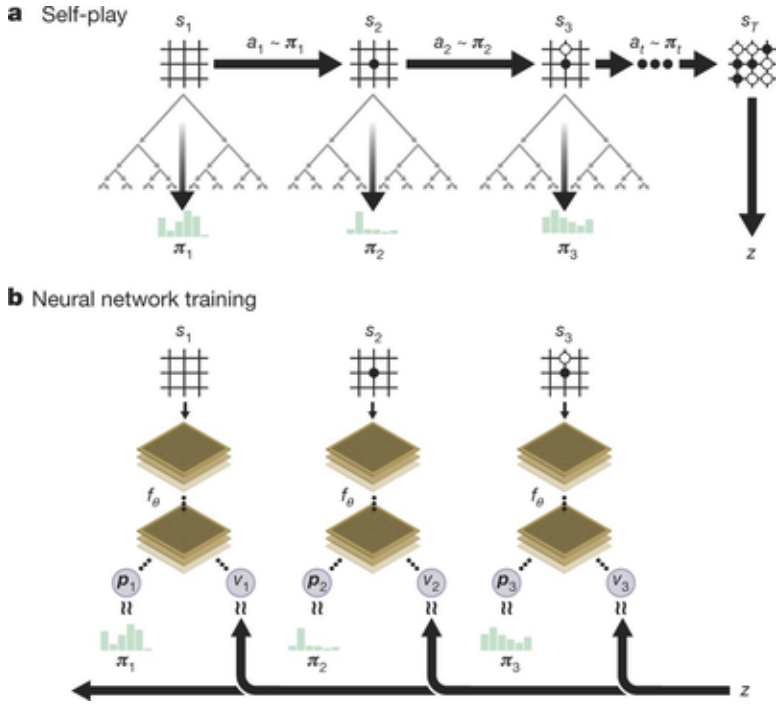
What is machine learning?

- **machine** = computer and computer progrem (in this course)
- **learning** = improving performance on a given task, based on experience / examples
- In other words:
  - instead of programming explicit rules, computer learns from examples.
  - often, computer will reach better performance than the programmer ever could!

Example 1: tic tac toe

- *Option A:* The programmer writes explicit rules, e.g., "if the opponent has two in a row, and the third position is free, place your mark there', etc
- *Option B:* Go through the game tree, choose optimally
- *Option C:* Let the computer try out various strategies by playing against itself and others, and noting which strategies lead to winning and which to losing (=**machine learning**)

Example 2: alpha go zero



- Figure from Silver et al. (2017) Nature. https://doi.org/10.1038/nature24270 (https://doi.org/10.1038/nature24270)

Example 3: spam filter

- **Classify** email messages as spam or ham (=not spam) based on message features
- Example: SpamAssassin https://spamassassin.apache.org/ (https://spamassassin.apache.org/)
  - Uses rule outputs and features and (historically) Naive Bayes classifier, nowadays perceptron to adjust rule weights
  - Has to work in *adversarial world* where the spammers are adapting (trying to circumvent the filter)
- Analogous problem: detecting malware

Example 3: regression

- Predict precipitation
- Figures from Eronen, Puolamäki et al. (2010) Evolutionary Ecology Research http://www.evolutionary-ecology.com/issues/v12/n02/ggar2538.pdf (http://www.evolutionary-ecology.com/issues/v12/n02/ggar2538.pdf) http://www.evolutionary-ecology.com/issues/v12/n02/hhar2539.pdf (http://www.evolutionary-ecology.com/issues/v12/n02/hhar2539.pdf)

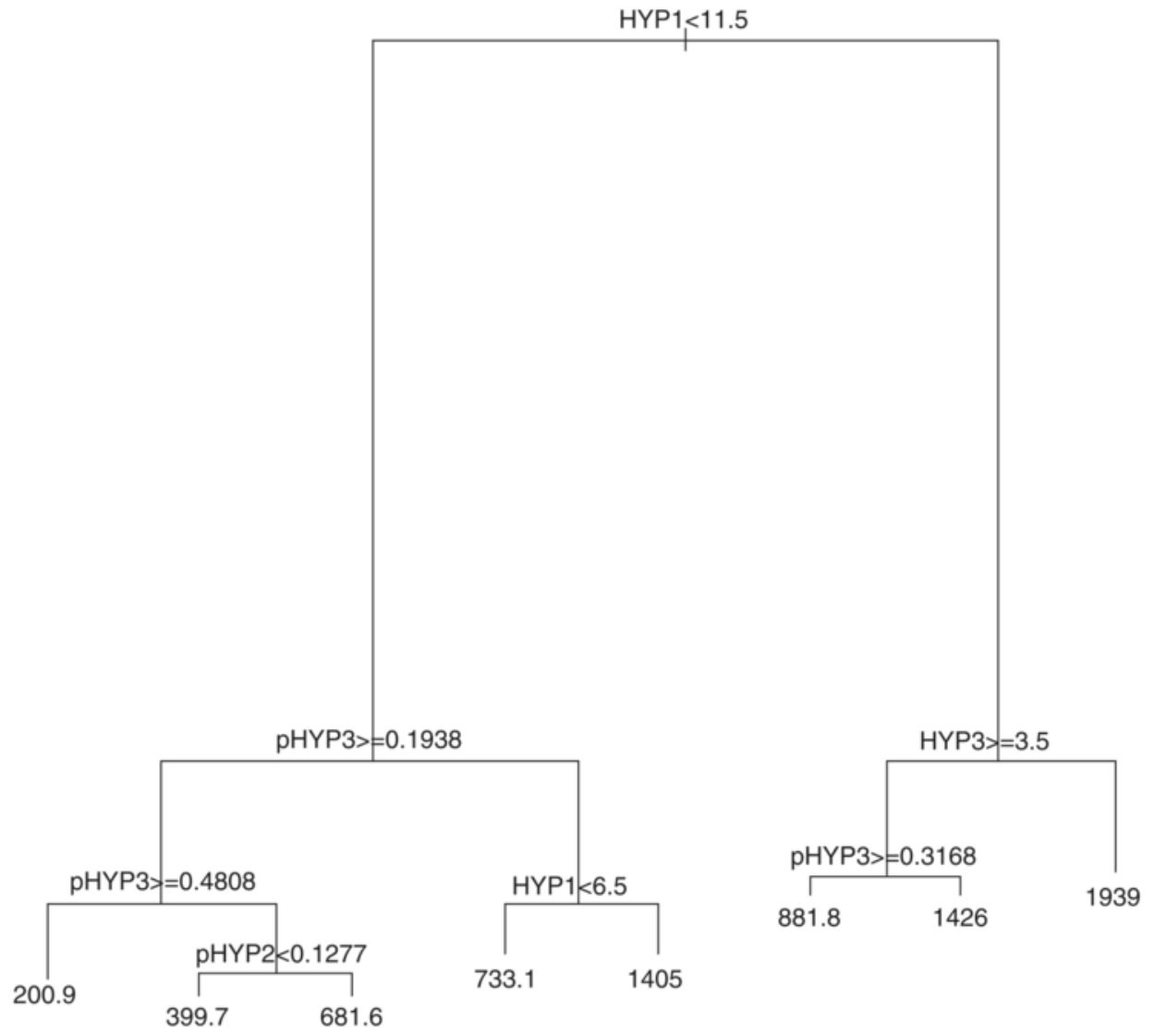
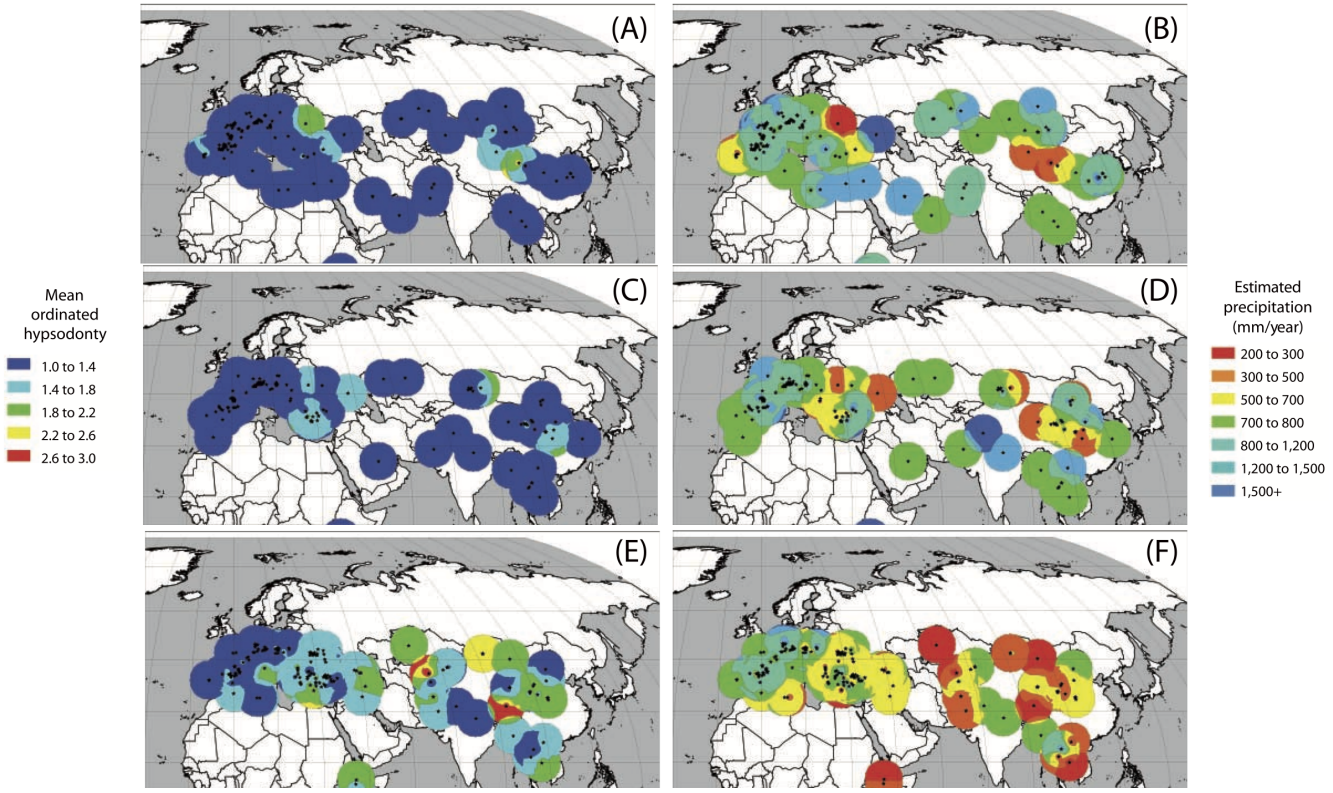
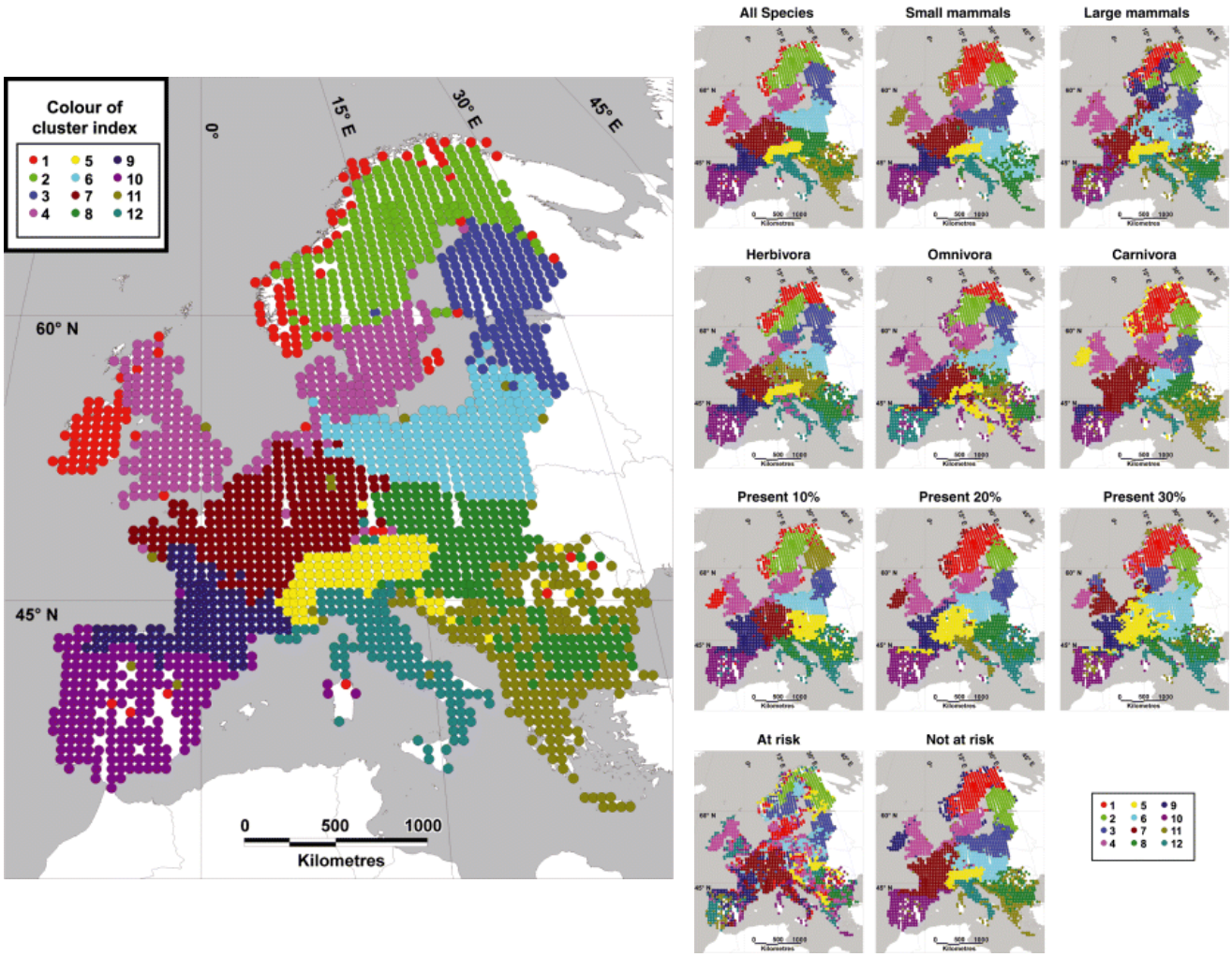


Fig. 1. Decision tree for annual precipitation using hypsodonty alone as regressor (see online Appendix 1 for other decision trees generated: evolutionary-ecology.com/data/2538A1.pdf).



Example 4: clustering

- Question: Can we find ecological communities?
- Question: What explains the communities?
- The 50 × 50 km map grids were grouped into clusters. Map grids within a cluster should occupy similar mammals.
- Figures from Heikinheimo et al. (2007) J. of Biogeography https://doi.org/10.1111/j.1365-2699.2006.01664.x (https://doi.org/10.1111/j.1365-2699.2006.01664.x)



Example 5: parable of big data

- Machine learning is powerful
- Consequence: you can mess up spectacularly and not notice it
- Lots of potential pitfalls:
  - ML can find unexpected relations which may lead to unexpected results
  - Predictive power vs. understandability vs. generalisability of results
  - More complex machinery easier it is to break (subtle and hard-to-spot mistakes can lead to major errors)

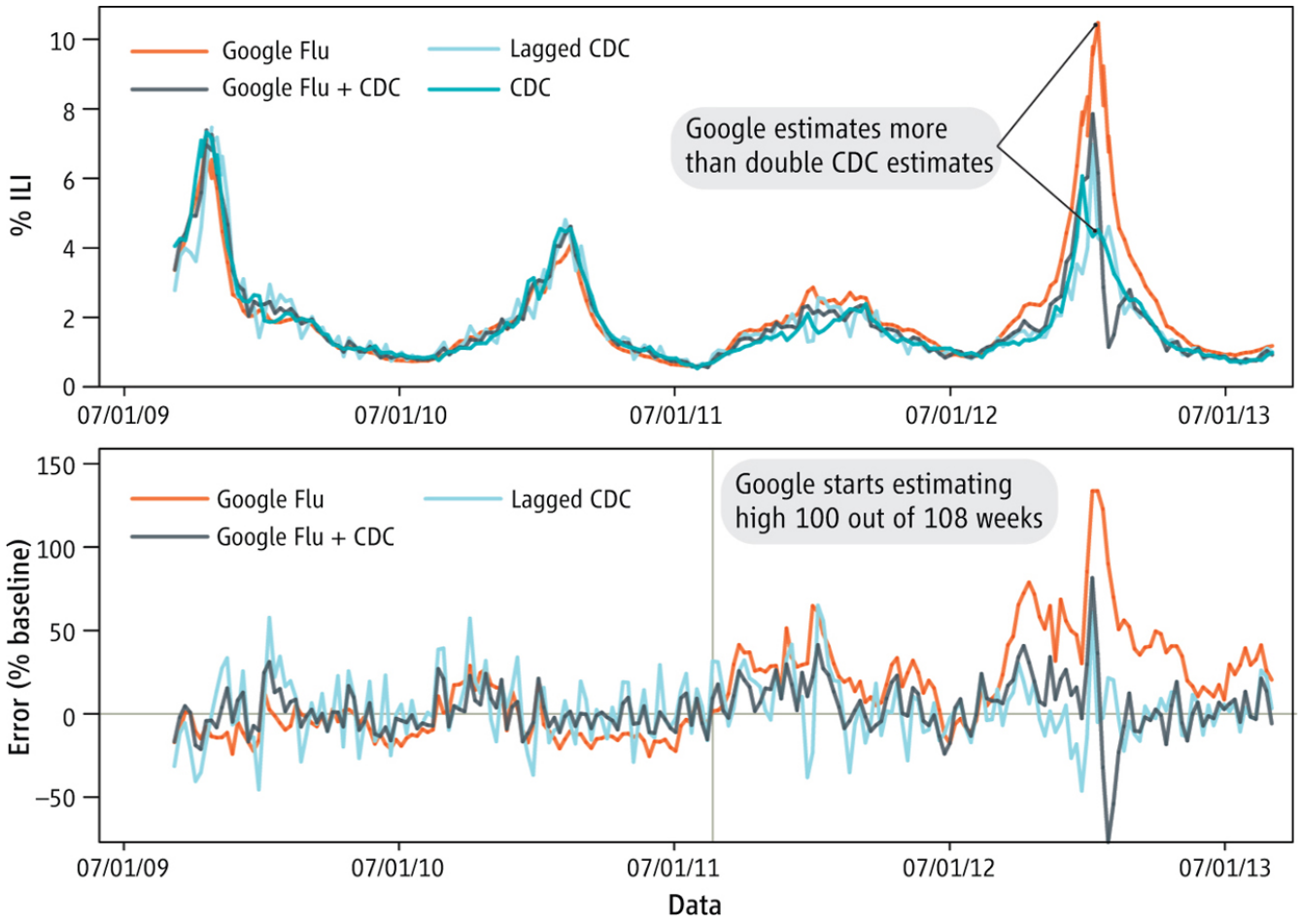


Figure from Lazer et al. (2014) Science https://doi.org/10.1126/science.1248506 (https://doi.org/10.1126/science.1248506)

Figure 5: Adversarial examples generated for AlexNet [9].(Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an *"ostrich, Struthio camelus"*. Average distortion based on 64 examples is 0.006508. Plase refer to `http://goo.gl/huaGPb` for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved.

Figure from Szegedy et al. (2013) https://arxiv.org/abs/1312.6199 (https://arxiv.org/abs/1312.6199)

Tools

- Machine learning software is already quite sophisticated
- It is relatively easy to do complex tasks that 10 years ago required PhD in computer science

- Example: Keras, TensorFlow, CNN, and mnist images (following example at https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py (https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py))
- MNIST: train CNN using 60000 28x28 pixel handrwritten digits 0-9. Classification task: given a new digit, try to guess the number 0-9 it is supposed to be.

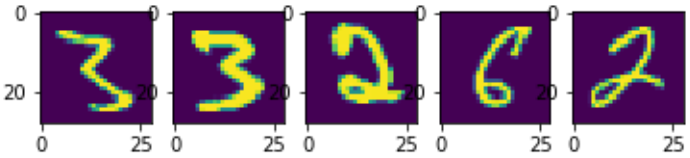
In [3]:

```
batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [48]:

```
fig, axs = plt.subplots(1,5)
for i in range(5):
    axs[i].imshow(x_train[np.random.choice(x_train.shape[0],1)[0],:,:,0])
plt.show()
```



In [5]:

```
if False:
    # It is a "known feature" that TensorFlow does not work well with Jupyter notebo
    # convert class vectors to binary class matrices
    y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)

    # define a simple convolutional neural network (CNN) structure that classifies r
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3),
                     activation='relu',
                     input_shape=input_shape))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])
```

In [6]:

```
# fit model
if False:
    start_time = time.time()
    # It is a "known feature" that TensorFlow does not work well with Jupyter notebo
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,verbose=1,
              validation_data=(x_test[:5000,:,:,:], y_test[:5000,:]))
    end_time = time.time()

    score = model.evaluate(x_test[5000:,:,:,:], y_test[5000:,:], verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])
    print("CNN: accuracy = %.4f time = %.2f" %
          (np.mean(np.argmax(model.predict(x_test[5000:,:,:,:]),axis=1)==np.argmax(y
```

```
Epoch 11/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.0293 -
accuracy: 0.9906 - val_loss: 0.0391 - val_accuracy: 0.9870
Epoch 12/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.0265 -
accuracy: 0.9917 - val_loss: 0.0424 - val_accuracy: 0.9854
Test loss: 0.01308044140580314
Test accuracy: 0.9959999918937683
CNN: accuracy = 0.9960 time = 767.21
```

Results

- CNN: training time 13 minutes, accuracy 0.996 (fraction of numbers in test set classified correctly)
- The same task with some other classifiers:

- logistic regression: training time 8 s, accuracy 0.925
- random forest: training time 37 s, accuracy 0.969
- k nearest neighbours (kNN): training time 1 s, accuracy 0.967
- support vector machine (SVM): training time 33 s, accuracy 0.955

- Why neural networks often have good accuracies on images?
  - trained on large annotated data sets
  - relatively low level features (image pixels)

In [7]:

```
x_train1 = x_train.reshape(x_train.shape[0],-1)
y_train1 = y_train
i = np.random.choice(x_train1.shape[0],10000,replace=False)
x_train2 = x_train1[i,:] # subsample 10000 data points (for kNN and SVC)
y_train2 = y_train1[i]
x_test1 = x_test.reshape(x_test.shape[0],-1)
y_test1 = y_test
```

In [8]:

```
start_time = time.time()
clf = LogisticRegression(random_state=42).fit(x_train1, y_train1)
end_time = time.time()
print("logistic regression: accuracy = %.4f time = %.2f" % (np.mean(clf.predict(x_te
```

```
logistic regression: accuracy = 0.9256 time = 15.98

/usr/local/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/
_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as show
n in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic
-regression (https://scikit-learn.org/stable/modules/linear_model.html
#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [9]:

```
start_time = time.time()
clf = RandomForestClassifier().fit(x_train1, y_train1)
end_time = time.time()
print("random forest: accuracy = %.4f time = %.2f" % (np.mean(clf.predict(x_test1)==
```

```
random forest: accuracy = 0.9695 time = 39.33
```

In [10]:

```
j = np.random.choice(y_test1.shape[0],1000,replace=False)
start_time = time.time()
clf = KNeighborsClassifier().fit(x_train2, y_train2)
end_time = time.time()
print("kNN: accuracy = %.4f time = %.2f" % (np.mean(clf.predict(x_test1[j,:])==y_tes
```

```
kNN: accuracy = 0.9500 time = 0.69
```

In [11]:

```
start_time = time.time()
clf = SVC().fit(x_train2, y_train2)
end_time = time.time()
print("SVC: accuracy = %.4f time = %.2f" % (np.mean(clf.predict(x_test1[j,:])==y_tes
```

```
SVC: accuracy = 0.9600 time = 32.68
```

What you really want to do?

- Estimate missing values
- Understand some process
- Explore data
- ...

Related disciplines

- Artificial intelligence (AI)
  - ML is a central technology that underlies AI
- Artificial neural networks, computational neuroscience
  - Modern ML grew from NN boom during the past millenea
- Pattern learning
  - recognizing objects and patterns, typically implements ML techniques

Related disciplines (cont.)

- Data mining
- Statistics
- Algorithms (computer science)
- Optimization

Deep learning

- “Family” of machine learning methods.
- Has become incredibly popular in the past few years.
- Yields very good results, e.g., in computer vision and speech recognition tasks.
- Heavily based on classical work on artificial neural network methods. (Fundamentally not really that novel.)
- Basic principles of ML covered in this course also apply to deep learning!

Availability of data

- These days it is very easy to
  - collect data (sensors are cheap, much information digital)
  - store data (hard drives are big and cheap)
  - transmit data (essentially free on the internet)
- Result: everyone has large quantities of data
  - how to deal with it?

Resources: datasets

- UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html (http://www.ics.uci.edu/~mlearn/MLRepository.html)
- Statlib: http://lib.stat.cmu.edu/ (http://lib.stat.cmu.edu/)
- Our own smartsmear... https://avaa.tdata.fi/web/smart/smear (https://avaa.tdata.fi/web/smart/smear)

Resources: ML journals

- Journal of Machine Learning Research www.jmlr.org (http://www.jmlr.org)
- Machine Learning
- Neural Computatio Neural Networks
- IEEE Trans on Neural Networks and Learning Systems
- IEEE Trans on Pattern Analysis and Machine Intelligence
- Journals on Statistics/Data Mining/Signal
- Processing/Natural Language
- Processing/Bioinformatics/...

Resources: ML conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Uncertainty in Artificial Intelligence (UAI)
- Computational Learning Theory (COLT)
- International Conference on Artificial Neural Networks (ICANN)
- International Conference on AI & Statistics (AISTATS)
- International Conference on Pattern Recognition (ICPR)
- ...

Chapter 1: data and the tools

Data wrangling challenges

- Claim: 80% of the work on data mining project is about data understanding and data preparation
- See, e.g., Cross Industry Standard Process Data Mining (CRISP-DM), https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining (https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining)

Data wrangling challenges

- **DP** Data parsing, e.g., converting csv's or tables
- **DD** Obtaining (or inferring) a data dictionary: basic types + semantics
- **DI** Data integration: Combining data from multiple sources
- **ER** Entity resolution: Recognising that two distinct pieces of information in the data concern the same entity. Includes deduplication and record linkage
- **FV** Format variability: e.g. for dates, but also for variability in names (e.g. IBM, I.B.M.).
- **SV** Coping with structural variability in the data, e.g. wide vs tall format. Also variation over time.
- **MD** Identifying and repairing missing data
- **AD** Anomaly detection and repair

List: Chris Williams

"Tools" for machine learning

- One of the major developments during last decades is introduction of software tools to implement AI methods
  - PhD no longer needed to run a complex algorithms (deep learning etc.)
  - PhD may still be needed to understand what happens
- Scientific publications often include open source libraries to implement the methods
- Take-away:
  - use these tools
  - contribute to these tools
  - recognise more the general computational problems and solve them instead of separately. tacking specific problems
- Examples: Keras for deep learning, Stan for probabilistic reasoning, R for statistical analysis etc.

Statistics on tools

- Top-3 data science tools (all used by over half of data scientists):
  - SQL
  - Python
  - R
  - Source: Data Science Salary Survey 2017, https://www.oreilly.com/data/free/files/2017-data-science-salary-survey.pdf (https://www.oreilly.com/data/free/files/2017-data-science-salary-survey.pdf)
- The list might be different after 10 years
  - You should be able to switch between tools and recognize their strengths and weaknesses!

Problem

Example of how to extract data (see aktia.zip).

Looking at the data and communicating results

- Look at the data
- Understand what you try to do
  - Apply the dummy method
  - Apply the absolutely simplest non-trivial method
  - Only then try the advanced apporach!
- In data science communication is the key
  - Write a report of the above which includes figures and explanation of what you have done
  - Include your name on the report
  - It is not enough just to show random plots to confused audience!

Ingredients of machine learning

- *Task* is what an end user actually wants to do.
- *Computational problem* is a (hopefully general) mathematical definition of the task
  - usually includes some *performance measure* (to be optimised somehow)
- *Model* is a (hopefully good) solution to the computational problem.
- *Data* consists of objects in the domain the user is interested in, with perhaps some additional information attached.
- *Machine learning algorithm* produces a model based on data.
- *Features* are how we represent the objects in the domain.

Task

- Task is an actual data processing problem some end user needs to solve.
- Examples were given in earlier (playing go, finding groups of mammals, estimating flu epidemics...)
- Typically, a task involves getting some input and then producing the appropriate output.
- For example, in hand-written digit recognition, the input is a pixel matrix representing an image of a digit, and the output is one of the labels ’0’, ..., ’9’.
- Machine learning is a way to find a solution for this data processing problem when it’s too complicated or poorly understood for a programmer (or an application specialist) to figure out.

Supervised learning

Many common tasks belong to the area of *supervised learning* where we neeed to produce some target value (often called *label*):

- binary classification: divide inputs into two categories
  - *Example:* classify e-mail messages into spam and non-spam
- multiclass classification: more than two categories
  - *Example:* classify an image of a digit into one of the classes '0', ..., '9'

Supervised learning

- multilabel classification: multiple classes, of which more than one may match simultaneously
  - *Example:* classify news stories based on their topics
- regression: output is a real-valued
  - *Example:* predict the value of a house based on its size, location etc.

Supervised learning as a computational problem

Task: predict a value y given some *covariates* x given a (training) data which contains pairs of (x, y).

- *Training set*, $D_{tr} = \{(x_i, y_i)\}_{i=1}^{n}$ of n points drawn i.i.d. from fixed but unknown distribution F.
- *Hypothesis class*, the set of models (here functions) $\hat{y} = f(x)$ to predict y, given x.
- *Loss function*, $L(\hat{y}, y)$ difference between the output of model $\hat{y}$ and the desired output y.

Computational problem: given the training set D_{tr}, the hypothesis class, and a loss function, find a function $\hat{f}$ such that the expected loss on the data drawn from F is minimized, i.e.,

$$\hat{f} = \arg\min_f E_F(L(f(x), y))).$$

Unsupervised learning

In unsupervised learning, there is no specific target value of interest.

Examples of unsupervised learning tasks:

- *clustering: partition the given set of data into *clusters* so that elements that belong to same cluster are similar (in terms of some given similarity measure)
- *dimensionality reduction:* if the data is high-dimensional (each data point is described by a large number of variables), find an alternative lower-dimensional representation that retains as much of the structure as possible
- *association rules:* given shopping cart contents of different customers, find product combinations often bought together

Unsupervised learning as a computational problem

- *clustering (k-means):* Given a constant k and data $D = \{X_i\}_{i=1}^{n}$, where $X_i \in \mathbb{R}^d$, find k *cluster centroids* $\overline{X}_j$, where $j \in \{1, \ldots, k\}$, such that the loss
  $$\sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} \left|X_i - \overline{X}_j\right|^2/n$$
  is minimised.
  - Each the data item X_i will be assigned to cluster indexed by $\arg\min_{j\in\{1,\ldots,k\}} \left|X_i - \overline{X}_j\right|^2$.

Unsupervised learning as a computational problem

- *dimensionality reduction (PCA):* Find a unit vector $V \in \mathbb{R}^d$ such that the variance
  $$\sum_{i=1}^{n} \left(X_i^T V\right)^2/n$$
  is maximised.
  - Data item $X_i \in \mathbb{R}^d$ is projected into $P_i \in \mathbb{R}$ by $P_i = X_i^T V$.

Semisupervised learning

In semisupervised learning, we have target values only for a subset of the input examples.

- Unsupervised learning can be used as pre-processing to help supervised learning, e.g., *image classification*:
  - Internet has as many non-classified images as we could ever want
  - less easy to *label* the data (assign the correct class to each image)
- (one possible) solution: use the unlabelled images to find a low-dimensional representation, then use a smaller set of labelled images to solve the hopefully easier lower-dimensional classification problem

Reinforcement learning

In reinforcement learning, the learning algorithm **can interact with its environment** and learn from rewards it receives.

- Main idea: A learning *agent* is in some state s, and carries out action a, which takes the agent to state s'. (e.g., carries out some specific move in Chess)
- After taking some number of steps the agent receives a *reward*. (e.g., takes the queen of the opponent)
- The agent aims to *learn a policy* that maps states to actions so that it's total reward is maximised.
- Can be thought of as a type of supervised learning, where the target value is specified indirectly through the rewards, possibly with a temporal delay.
- Interesting stuff, but not covered in this course.

Predictive vs. descriptive model

- **Predictive model:** our goal is generalization, i.e., predict outcomes in future data
- **Descriptive model:** no guarantees about generalizaton to future data, we want to understand the data (a.k.a. exploratory analysis, data mining)
- Distinction between supervised vs. unsupervised learning is whether the target values are available to the learning algorithm: both can be either predictive or descriptive
- Keep in mind: **Look at the data first!** It is very easy to screw things up by blindly fitting models to data without slowing down and taking a closer look.

Computational problem

- Usually, task can be expressed as a instance of a **computational problem**
- Computational problems are more generic than tasks
- Often, the computational problem is some kind of a optimisation problem
- In this course we focus on a couple of common and generic computational problems
- When faced with a practical ML problem, always think the corresponding computational problem!
  - Then choose a model and make the algorithm to solve the computational problem (not practical task!)
- Wrong way:
  - Start hacking the model and algorithm and hope the best!