Machine Learning and AI

Jour 1 : fondamentaux avancés

Jeff Abrahamson

July 2024

Welcome

Course structure

Two full days:

- 11, 12 July

Course structure

- Email `jeff@p27.eu`
- Github: `https://github.com:JeffAbrahamson/ML-diva-beapp`

Course structure

- Day 1 - ML theory
- Day 2 - ML Ops and Practices

Supervised

Unsupervised

Reinforcement

Machine learning is not magic

Machine learning is mathematics

Mostly, it's these maths:

- Probability
- Statistics
- Linear algebra
- Optimisation theory
- Differential calculus

Unless you want to, we'll skip the maths. For a certain definition of "skip".

Probability

events

- Independence
- Dependence
- Bayes Theorem

Statistics

What is Statistics

1. Identify a question or problem.
2. Collect relevant data on the topic.
3. Analyze the data.
4. Form a conclusion.

What is Statistics

1. Identify a question or problem.
2. Collect relevant data on the topic.
3. Analyze the data.
4. Form a conclusion.

Sadly, sometimes people forget 1.

What is Statistics

1. Identify a question or problem.
2. Collect relevant data on the topic.
3. Analyze the data.
4. Form a conclusion.

Statistics is about making 2–4 efficient, rigorous, and meaningful.

OpenIntro Statistics, 2nd edition, D. Diez, C. Barr, M. Çetinkaya-Rundel, 2013.

What is data science?

(Exercise: Is this the same question as the last slide?)

1. Define the question of interest
2. Get the data
3. Clean the data
4. Explore the data
5. Fit statistical models
6. Communicate the results
7. Make your analysis reproducible

What is data science?

(Exercise: Is this the same question as the last slide?)

1. Define the question of interest
2. Get the data
3. Clean the data
4. Explore the data
5. Fit statistical models
6. Communicate the results
7. Make your analysis reproducible

What the public thinks.

What is data science?

(Exercise: Is this the same question as the last slide?)

1. Define the question of interest
2. Get the data
3. Clean the data
4. Explore the data
5. Fit statistical models
6. Communicate the results
7. Make your analysis reproducible

Where we spend most of our time.

What is data science?

(Exercise: Is this the same question as the last slide?)

1. Define the question of interest
2. Get the data
3. Clean the data
4. Explore the data
5. Fit statistical models
6. Communicate the results
7. Make your analysis reproducible

The easiest part to forget.

What is data science?

https://simplystats.github.io/2015/03/17/data-science-done-well-looks-easy-and-that-is-a-big-problem-for-data-scientists/

Anecdote

Some properties of anecdote:

- is data
- haphazardly collected
- is generally not representative
- sometimes result of selective retention
- does not accumulate to be representative
- might be true (by chance)
- is ok to use as hypothesis, but be clear that hypothesis is anecdote

Study Types

- Observational
- Experimental

Study Types

- Observational
- Experimental

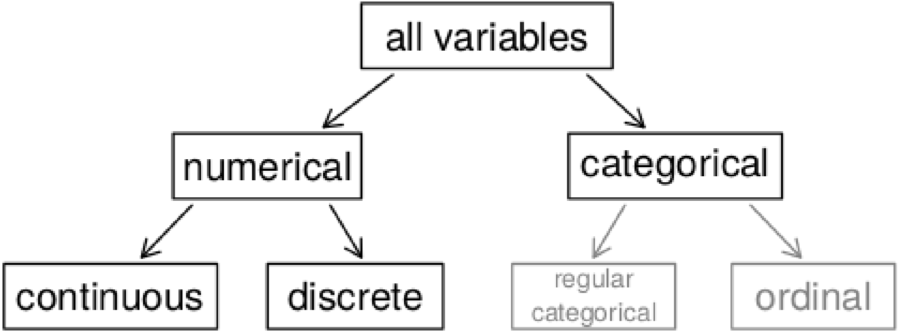
What can go wrong?

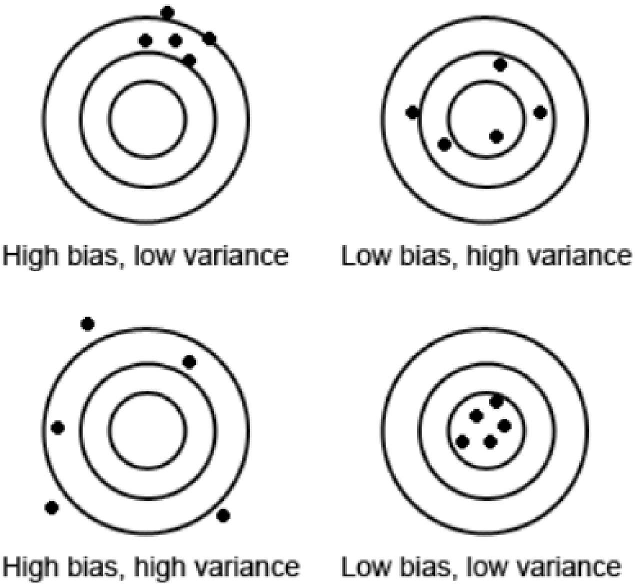
- Forgetting that association $\neq$ causation
- Not random
- Confounding variables

Variables and statistics

- Input: features ("explanatory variables")
- Output: response variable
- Training set (learn parameters)
- Test Set (check learned parameters)
- Validation set (check learned hyperparameters)
- Cross validation (and jack knife and . . .)
- Bias and variance (picture coming up)

Variable types



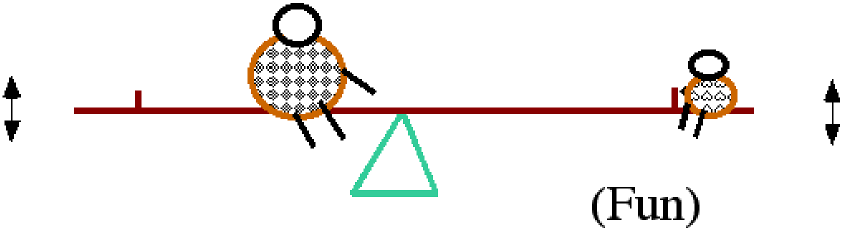

High bias, low variance
Low bias, high variance
High bias, high variance
Low bias, low variance

Mean

- Weighted and unweighted
- Centroid to physicists

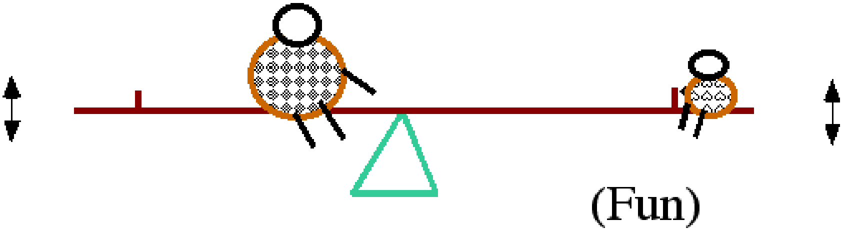
Mean

- Weighted and unweighted
- Centroid to physicists



Mean

- Weighted and unweighted
- Centroid to physicists

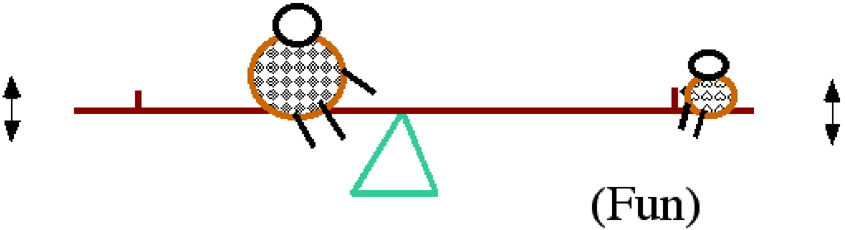


$$\mu = E(X) = \sum w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

http://telescopes.stardate.org/images/research/teeter-totter/TT4.gif

Mean

- Weighted and unweighted
- Centroid to physicists



$$\mu = E(X) = \sum \Pr(X = x_i) x_i$$

http://telescopes.stardate.org/images/research/teeter-totter/TT4.gif

Mean

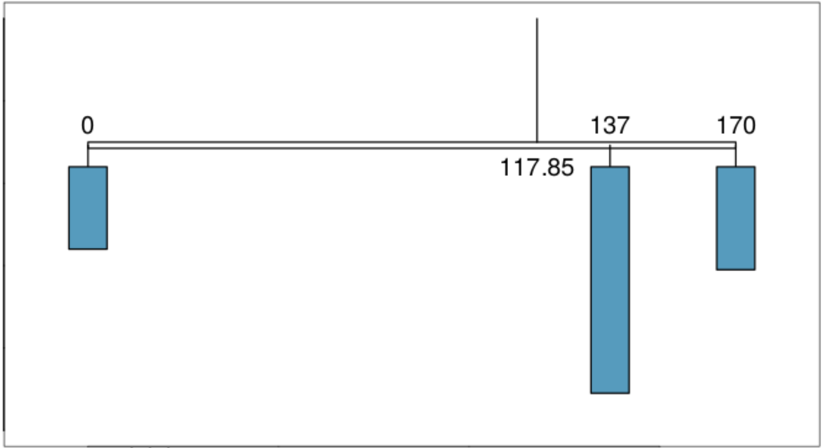
- Weighted and unweighted
- Centroid to physicists

$$\mu = E(X) = \int xf(x)\,\mathrm{d}x$$

http://telescopes.stardate.org/images/research/teeter-totter/TT4.gif

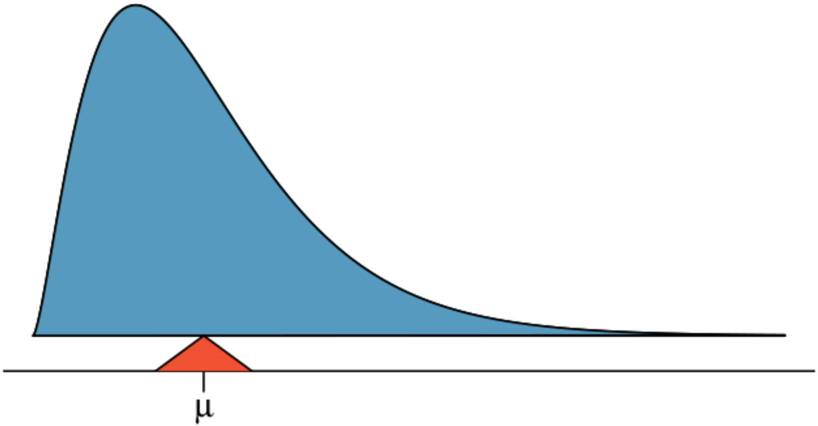
Mean

- Weighted and unweighted
- Centroid to physicists



Mean

- Weighted and unweighted
- Centroid to physicists



Population statistics

Mean is just a sum.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

This is a special case ($w_i = 1$) of the weighted mean:

$$\mu = \frac{1}{\sum w_i} \cdot \sum_{i=1}^{N} w_i x_i$$

Population statistics

Deviation is distance from mean.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Deviation of $x_i = \mu - x_i$

Population statistics

Variance is the mean square of deviations

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

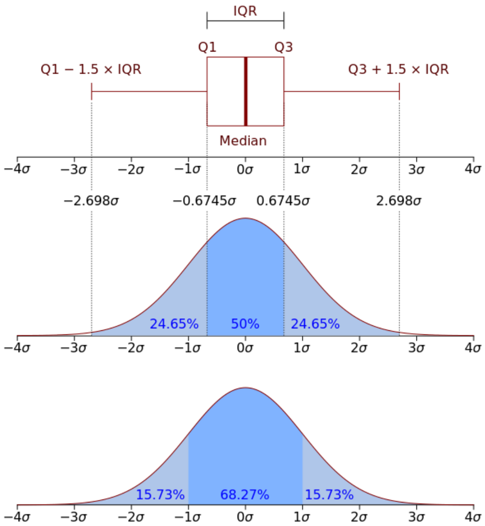
$$\mathrm{Var}(X) = \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(\mu - x_i)^2$$

Population statistics

Standard deviation is square root of variance

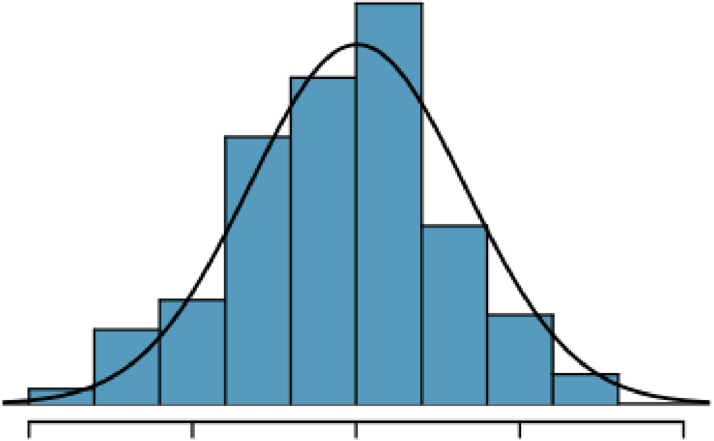
$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma = \sqrt{\mathrm{Var}(X)} = \sqrt{\sigma^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mu - x_i)^2}$$


IQR
Q1
Q3
Q1 − 1.5 × IQR
Q3 + 1.5 × IQR
Median
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ
−2.698σ
−0.6745σ
0.6745σ
2.698σ
24.65%
50%
24.65%
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ
15.73%
68.27%
15.73%
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ

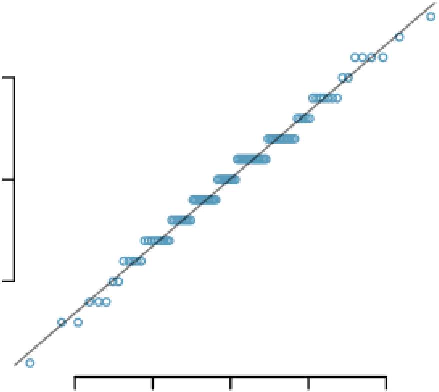
Evaluating Normal Approximations

Easy technique 1: visually compare to normal plot.



Evaluating Normal Approximations

Easy technique 2: normal probability plot.

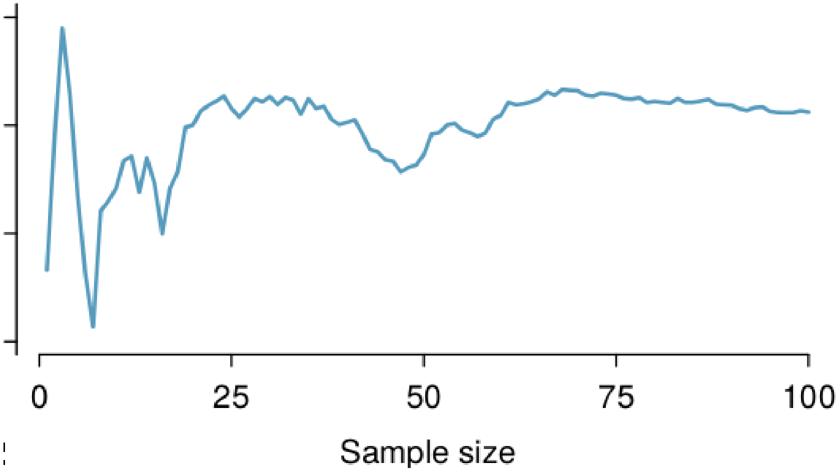


Also known as a quantile-quantile plot.

$$\overline{x} \neq \mu$$

Running mean. Sequence of partial sums (divided by number in sum).

Running mean. Sequence of partial sums (divided by number in sum).



Inference Concepts

Running mean. Sequence of partial sums (divided by number in sum).

Sampling variation. Change of $\overline{x}$ from one sample to the next.

Inference Concepts

Running mean. Sequence of partial sums (divided by number in sum).

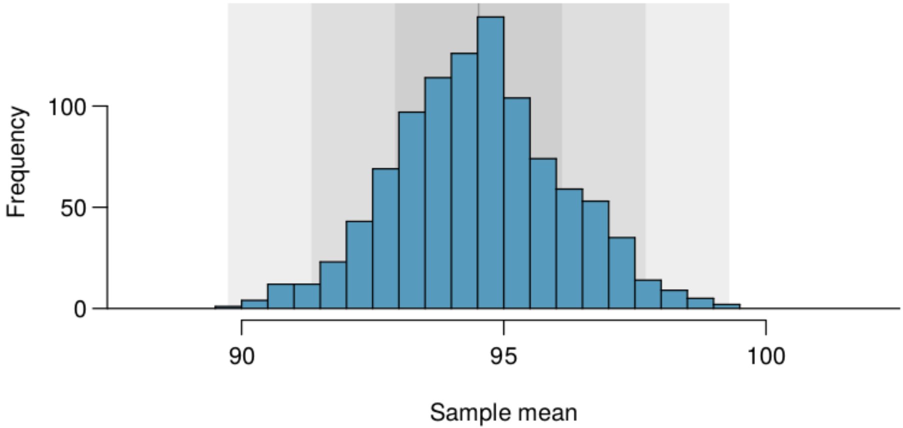
Sampling variation. Change of $\overline{x}$ from one sample to the next.

Sampling distribution. The distribution of possible point samples of a fixed size from a given population.

Inference Concepts

- Law of large numbers
- Central limit theorem

Sampling distribution



Confidence intervals

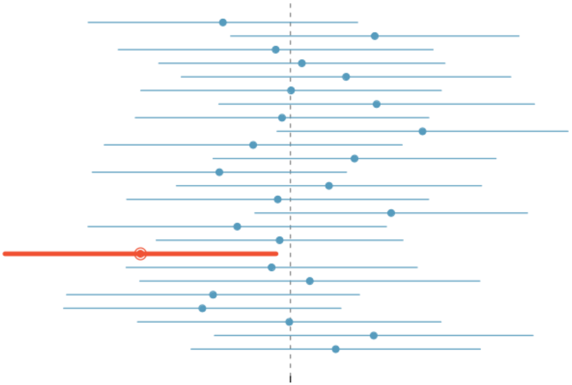
Sample *n* points, choose an interval around the sample mean.

A 95% confidence interval means if we sample repeatedly, about 95% of the samples will contain the population mean.

Confidence intervals

Sample *n* points, choose an interval around the sample mean.

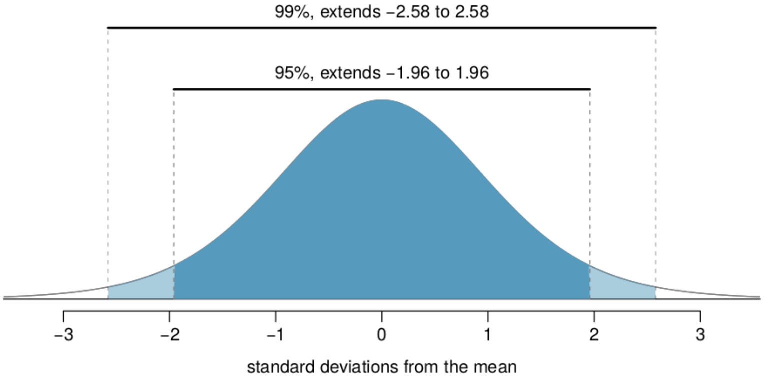
A 95% confidence interval means if we sample repeatedly, about 95% of the samples will contain the population mean.



Confidence intervals

Sample n points, choose an interval around the sample mean.

A 95% confidence interval means if we sample repeatedly, about 95% of the samples will contain the population mean.



Linear Algebra

Vector Space

Curse of Dimensionality

Linear algebra: basics

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^n$$

This is all mostly for convenience (not getting lost). Remember weighted averages?

$$\mu = w^T \cdot x$$

Linear algebra: basics

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

$$= \begin{Bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{Bmatrix} \in \mathbb{R}^{n\times n}$$

Linear algebra: basics

$$u + v = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{pmatrix}$$

Linear algebra: basics

$$\alpha v = \begin{pmatrix} \alpha v_1 \\ \alpha v_2 \\ \vdots \\ \alpha v_n \end{pmatrix} \qquad (\alpha \in \mathbb{R})$$

Linear algebra: basics

$$\| v \| = \sqrt{v_1^2 + \cdots + v_n^2}$$

Linear algebra: basics

$$u \cdot v = u_1 \cdot v_1 + \cdots + u_n \cdot v_n$$

$$= \| u \| \| v \| \cos \theta$$

Linear algebra: basics

$$C = A + B \iff c_{ij} = a_{ij} + b_{ij}$$

$$C = AB \iff c_{ij} = \sum_k a_{ik} b_{kj}$$

$$A = B^T \iff a_{ij} = b_{ji}$$

$$AA^{-1} = A^{-1}A = \mathrm{diag}(1)$$

Linear algebra: transformations

$$Ax = y \qquad f = T_A : \mathbb{R}^n \to \mathbb{R}^n$$

$$x = A^{-1}Ax = A^{-1}y \qquad f^{-1} = T_{A^{-1}} : \mathbb{R}^n \to \mathbb{R}^n$$

Linear algebra: transformations

B is a basis for V iff any of these conditions are met:

- B is a minimal generating set of V
- B is a maximal set of linearly independent vectors
- Every vector $v \in V$ can be expressed in a unique way as a sum of $b_i \in B$

(The conditions are equivalent.)

Linear algebra: transformations

B is a basis for V iff any of these conditions are met:

- B is a minimal generating set of V
- B is a maximal set of linearly independent vectors
- Every vector $v \in V$ can be expressed in a unique way as a sum of $b_i \in B$

(The conditions are equivalent.)

Bases are not unique.

Linear algebra: transformations

Eigenvectors, eigenvalues:

$$Av = \lambda v$$

Linear algebra: transformations

Eigenvectors, eigenvalues:

$$Av = \lambda v$$

$$Av = \lambda 1 v \iff (A - \lambda 1)v = 0$$

Linear algebra: transformations

Eigenvectors, eigenvalues:

$$Av = \lambda v$$

Some matrices are diagonalisable. Then

$$A = Q \Lambda Q^{-1} \qquad \text{with } \Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & \lambda_n \end{bmatrix}$$

$$\text{and } Q = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix}$$

video time

Features

Features and Modeling

Vector spaces

Vector spaces

Features are dimensions

Feature extraction

Feature engineering

Feature extraction

Feature engineering

Synthetic features

Feature Engineering

1. Brainstorm
2. Pick some
3. Make them
4. Evaluate
5. Repeat

This is most often done with neural nets these days, but the technique is still useful.

One of K = one-hot encoding

Text features

Bag of words

- Corpus (documents)
- Vocabulary (set of unique words)
- Words

Text features

Bag of words

- Order doesn't matter
- Stop words
- Stemming *(racinisation, désuffixation)*
- Lemmatisation *(transformer en lemme)*

Image features

- Corners, edges (rotation invariant, but scaling can hide)
- More complex: scale space or RNN
- Point matching is easy

Since 2012 or so, this is mostly done by neural networks.

Image features

Problems

- Illumination
- Scale
- Rotation
- Skew (perspective)
- Data size (matrices not sparse)

python

Useful tools

- virtualenv
- pip
- ipython
- jupyter notebook
- `conda.pydata.org`

Notes

- `pip install -r requirements.txt`
- ipython offers tab completion (vs python)
- jupyter notebook opens in a browser, caches cell output but not cell state

pandas

```
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
```

`Dataframe` has many constructors. For example,

```
In [5]: pd.DataFrame({ 'A' : 1.,
                      'B' : pd.Timestamp('20161209'),
                      'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
                      'D' : np.array([3] * 4, dtype='int32'),
                      'E' : pd.Categorical(["test","train","test","train"]),
                      'F' : 'hello' })
Out[5]:
   A          B  C  D      E      F
0  1 2016-12-09  1  3   test  hello
1  1 2016-12-09  1  3  train  hello
2  1 2016-12-09  1  3   test  hello
3  1 2016-12-09  1  3  train  hello

In [6]:
```

Viewing data

```
In [16]: dates = pd.date_range('20161209', periods=4, freq='1w')

In [17]: df = pd.DataFrame(np.random.randn(4,5), index=dates,
                           columns=list('ABCDE'))

In [18]: df.head()
Out[18]:
                   A         B         C         D         E
2016-12-11 -1.303610 -1.235823  0.621914  0.379340 -0.326934
2016-12-18 -1.218197 -1.113826  0.546314 -0.255001 -0.135573
2016-12-25 -0.124625  0.337268 -0.406295  0.587049 -0.904906
2017-01-01 -0.283182 -0.866213  0.051509  0.693037 -0.661055

In [19]:
```

Basic data exploration

```
In [19]: df.describe()
Out[19]:
              A         B         C         D         E
count  4.000000  4.000000  4.000000  4.000000  4.000000
mean  -0.732403 -0.719648  0.203361  0.351106 -0.507117
std    0.614672  0.721194  0.478728  0.424558  0.342755
min   -1.303610 -1.235823 -0.406295 -0.255001 -0.904906
25%   -1.239550 -1.144325 -0.062942  0.220755 -0.722018
50%   -0.750689 -0.990019  0.298912  0.483195 -0.493995
75%   -0.243543 -0.565343  0.565214  0.613546 -0.279094
max   -0.124625  0.337268  0.621914  0.693037 -0.135573

In [20]:
```

Select a column (series)

```
In [20]: df.loc[dates[1]]
Out[20]:
A   -1.218197
B   -1.113826
C    0.546314
D   -0.255001
E   -0.135573
Name: 2016-12-18 00:00:00, dtype: float64

In [21]:
```

Select a range

```
In [21]: df.loc[:, ['A', 'C']]
Out[21]:
                   A         C
2016-12-11 -1.303610  0.621914
2016-12-18 -1.218197  0.546314
2016-12-25 -0.124625 -0.406295
2017-01-01 -0.283182  0.051509

In [22]:
```

Boolean selection criteria

```
In [23]: df[df.D > 0]
Out[23]:
                   A         B         C         D         E
2016-12-11 -1.303610 -1.235823  0.621914  0.379340 -0.326934
2016-12-25 -0.124625  0.337268 -0.406295  0.587049 -0.904906
2017-01-01 -0.283182 -0.866213  0.051509  0.693037 -0.661055

In [24]:
```

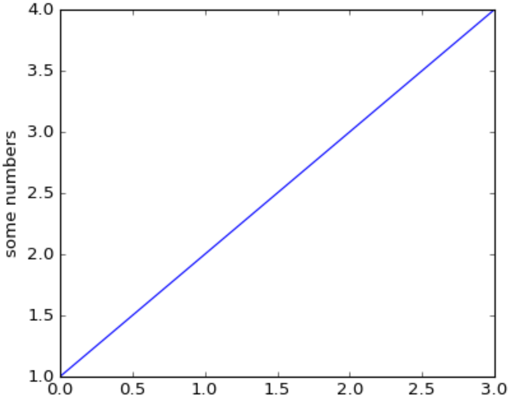
Recommended

`http://www.gregreda.com/2013/10/26/intro-to-pandas-data-structures/`

Plotting

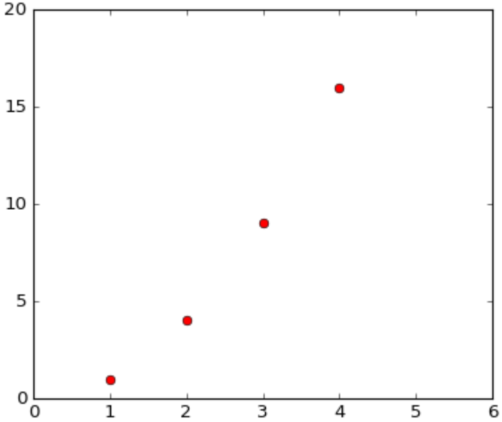
Draw a line

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



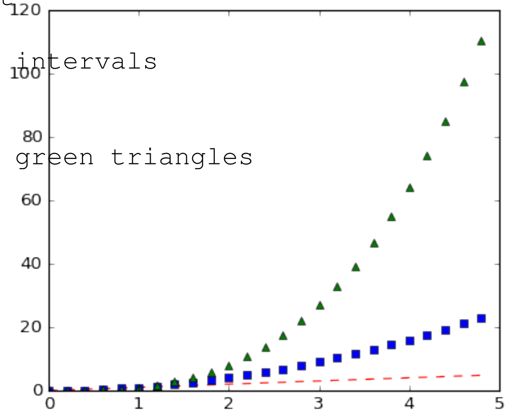
Draw a line

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.ylabel('some numbers')
plt.show()
```



Draw a line

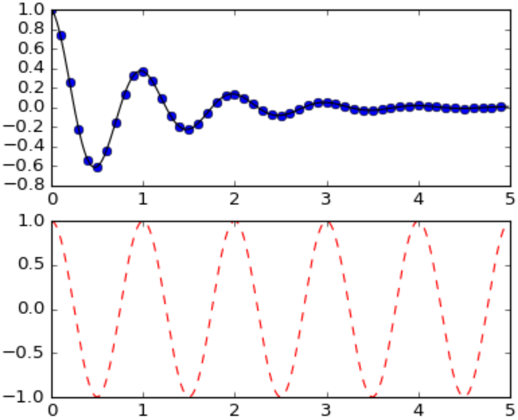
```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t,
         'r--', t,
         t**2, 'bs',
         t, t**3, 'g^')
plt.show()
```



Draw two curves

```
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

Plotting

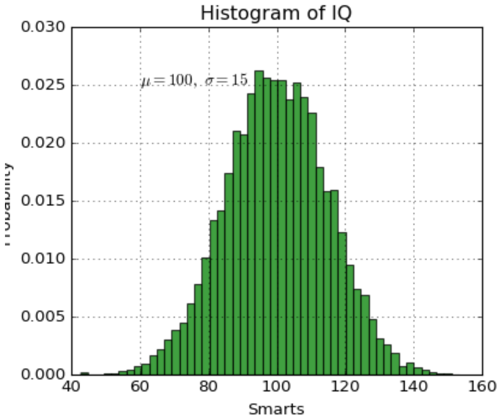


Plotting

Draw two curves

```
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

Plotting

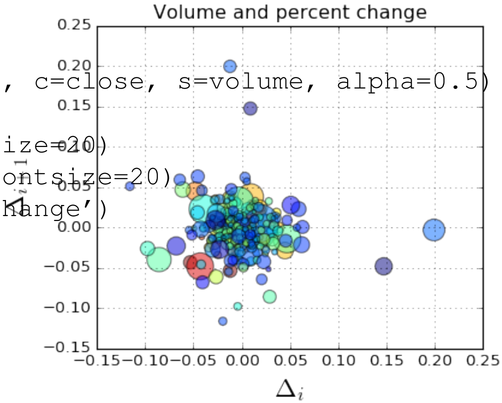


Histogram of IQ
$\mu = 100,\ \sigma = 15$
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140
160
Smarts

Scatter plot

http://matplotlib.org/mpl_examples/pylab_examples/scatter_demo2.py

```
import numpy as np
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.scatter(delta1[:-1], delta1[1:], c=close, s=volume, alpha=0.5)

ax.set_xlabel(r'$\Delta_i$', fontsize=20)
ax.set_ylabel(r'$\Delta_{i+1}$', fontsize=20)
ax.set_title('Volume and percent change')

ax.grid(True)
fig.tight_layout()

plt.show()
```



Plotting

http://matplotlib.org/users/pyplot_tutorial.html

http://matplotlib.org/users/beginner.html

Linear and Logistic Regression

Linear models

Problem: $\{(x_i, y_i)\}$.

Given x, predict $\hat{y}$.

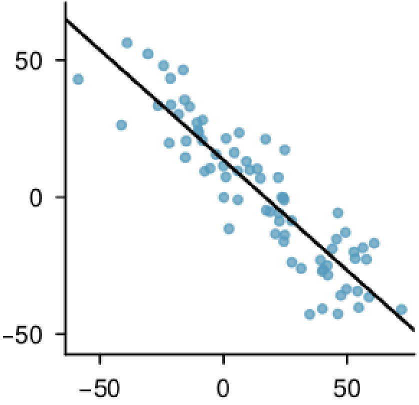
Linear models

x: **explanatory** or **predictor** variable.

y: **response** variable.

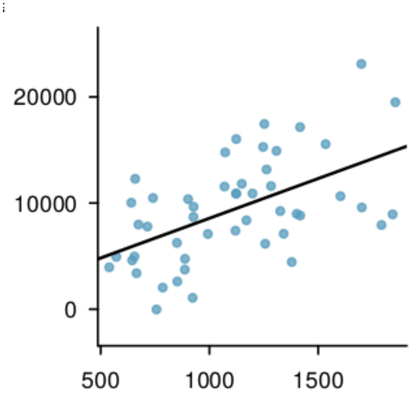
For some reason, we believe a linear model is a good idea.

Example:

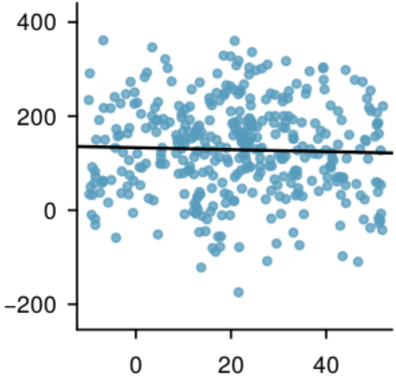


Linear models

Example:

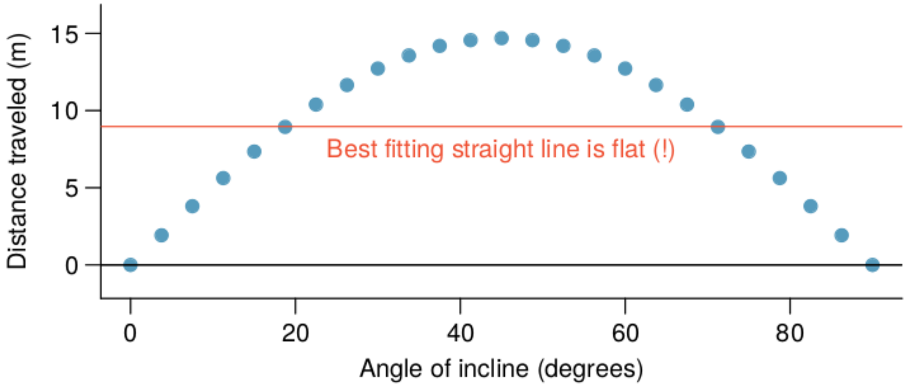


Example:



Linear models

Example:



Residuals

What's left over.

$$\text{data} = \text{fit} + \text{residual}$$

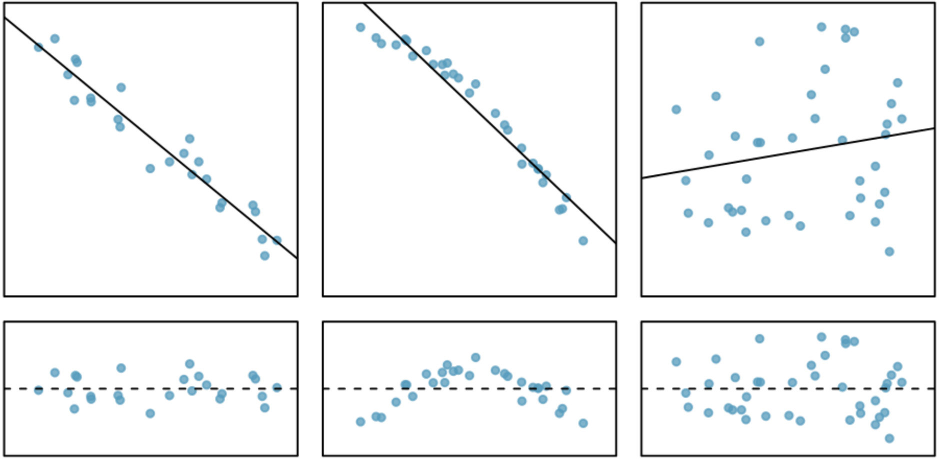
Residuals

What's left over.

$$y_i = \hat{y}_i + e_i$$

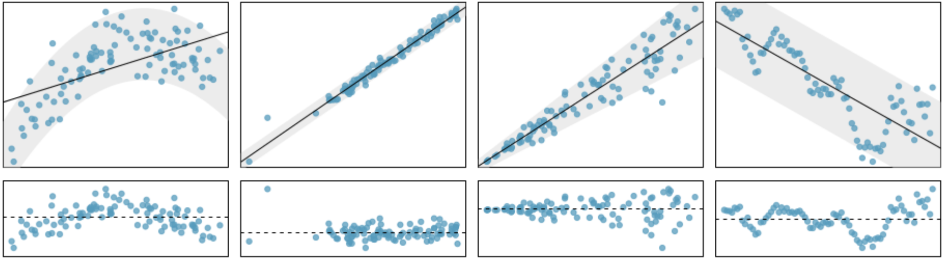
Residuals

What’s left over.



Residuals

What's left over.



Residuals

What's left over.

Goal: small residuals.

$$\sum \mid e_i \mid$$

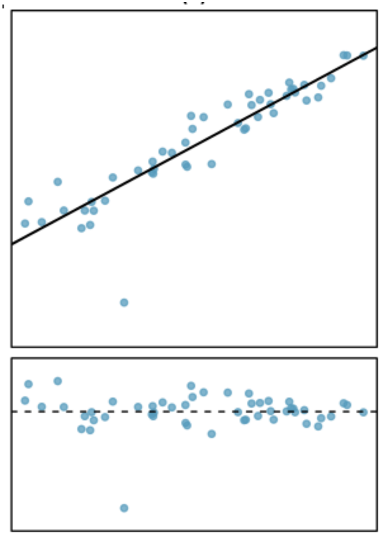
Residuals

What's left over.

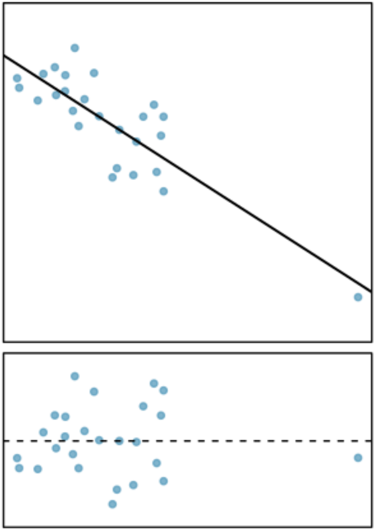
Goal: small residuals.

$$\sum e_i^2$$

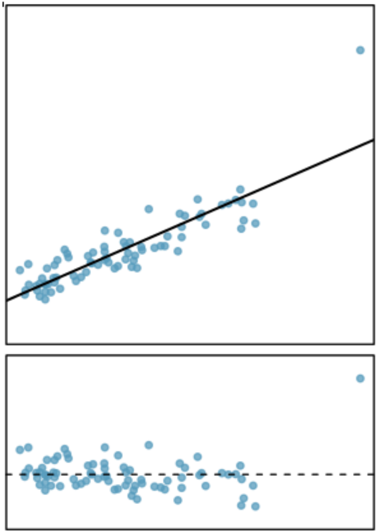
Outliers



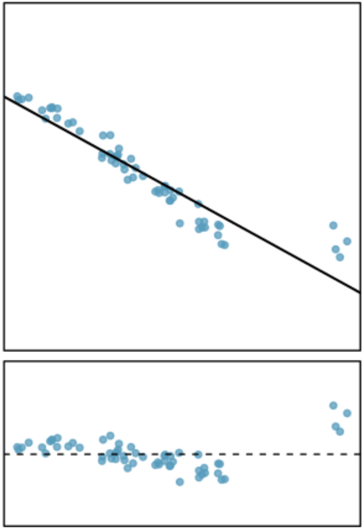
Outliers



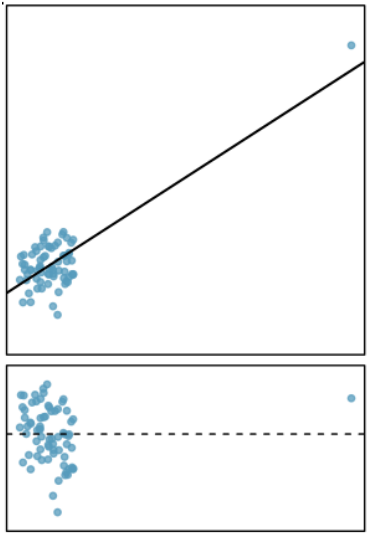
Outliers



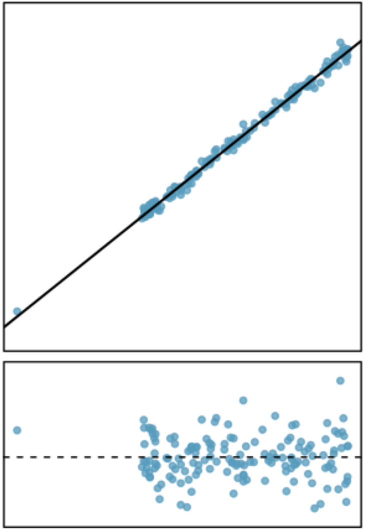
Outliers



Outliers



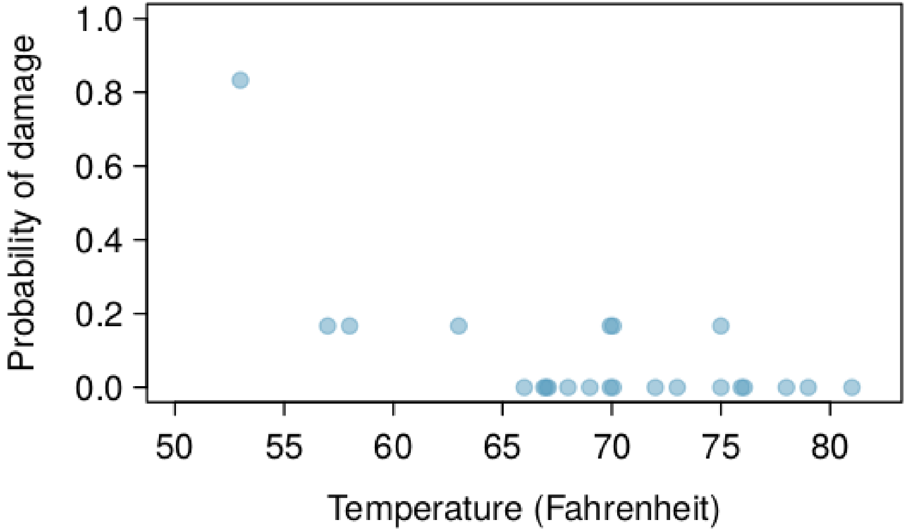
Outliers



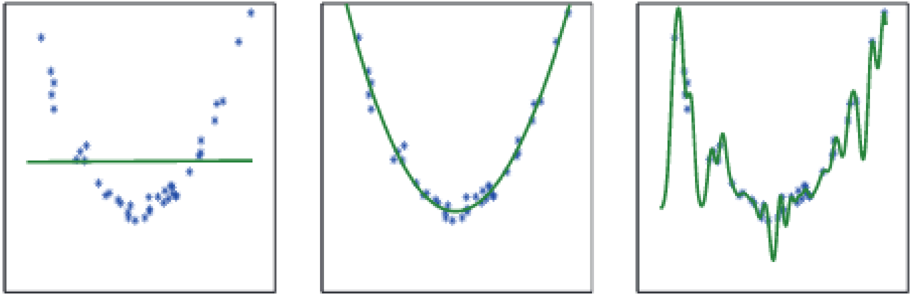
Outliers

Don’t ignore outliers.

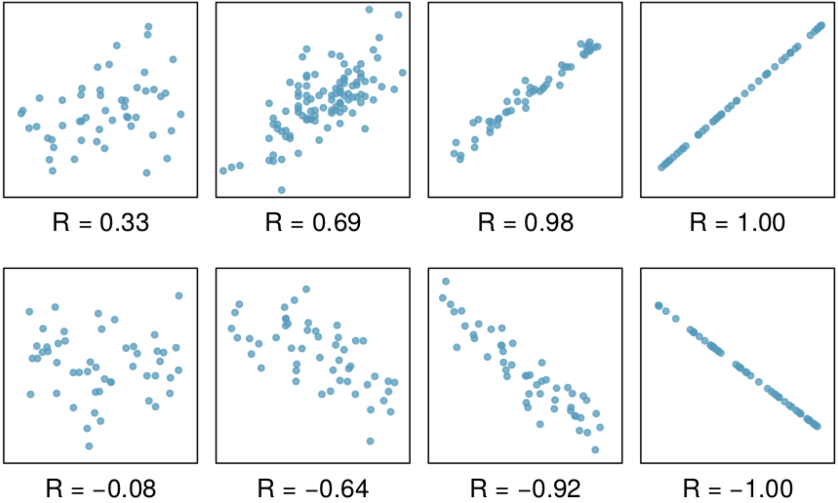
Outliers


1.0
0.8
0.6
0.4
0.2
0.0
Probability of damage
50
55
60
65
70
75
80
Temperature (Fahrenheit)

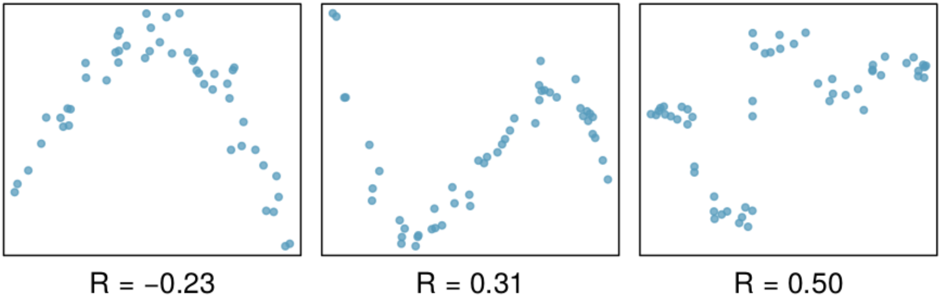
Underfitting, overfitting



Correlation

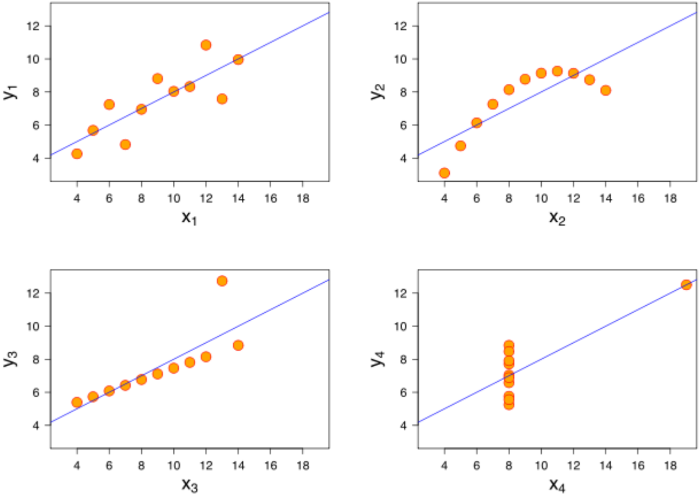


Correlation



Correlation

Anscombe's Quartet



Hypothesis (model)

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Cost function

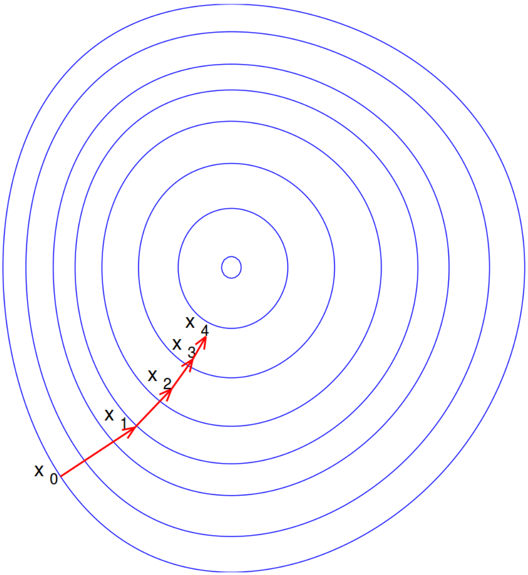
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Gradient descent

$$\begin{cases} \theta_0 & \leftarrow \theta_0 - \alpha \dfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \theta_1 & \leftarrow \theta_1 - \alpha \dfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \end{cases}$$

Gradient descent

$$\begin{cases} \theta_0 & \leftarrow \theta_0 - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) \\ \theta_1 & \leftarrow \theta_1 - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) \end{cases}$$


x 0
x 1
x 2
x 3
x 4

Hypothesis again

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \theta_1 x_1 \\
&= \theta_0 + \sum_{i=1}^{1} \theta_i x_i \\
&= [\theta_0, \theta_1] \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \\
&= \theta^T x
\end{aligned}
$$

Hypothesis (multiple regression)

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \sum_{i=1}^{n} \theta_i x_i \\
&= [\theta_0, \cdots, \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \\
&= \theta^T x
\end{aligned}
$$

Hypothesis (multiple regression)

$$
\begin{aligned}
h_\theta(x) &= \theta^T x \\
&= \theta^T x^{(1)}
\end{aligned}
$$

Hypothesis (multiple regression)

$$X = \begin{bmatrix} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & \cdots & | \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(m)} \end{bmatrix}$$

Hypothesis (multiple regression)

$$
\begin{aligned}
h_\theta(X) &= \theta^T X \\
&= \left[h_0(x^{(1)}), h_0(x^{(2)}), \cdots, h_0(x^{(m)})\right] \\
&= \theta^T X
\end{aligned}
$$

Hypothesis (multiple regression)

or $X\theta$ if row vectors. . .

Cost function (multiple regression)

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$
$$= \frac{1}{2m}(X\theta - Y)^T(X\theta - Y)$$

Gradient descent (multiple regression)

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x_j^{(i)}$$

$$\text{for } j = 1, \cdots, n$$

Gradient descent (multiple regression)

$$\theta \leftarrow \theta - \nabla J(\theta)$$

$$\text{where } \nabla = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \\ \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_n} \end{bmatrix}$$

Linear regression

- Continuous output
- Normal residues
- Predict $\hat{y}$ for x given $\{(x_i, y_i)\}$

Logistic regression

- Binary output
- Classification

Logistic regression

- Have: continuous and discrete inputs
- Want: class (0 or 1)

Probabilistic inspiration

$h_\theta(x) = .75 \iff$ event has 75% of being true

Probabilistic inspiration

$$h_\theta(x) = \Pr(y = 1 \mid x; \theta) = 0.75$$

Probabilistic inspiration

So this must be true:

$$\Pr(y = 0 \mid x; \theta) + \Pr(y = 1 \mid x; \theta) = 1$$

Probabilistic inspiration

$$\text{Set } y = 1 \iff h_\theta(x) = \Pr(y = 1 \mid x; \theta) > \frac{1}{2}$$

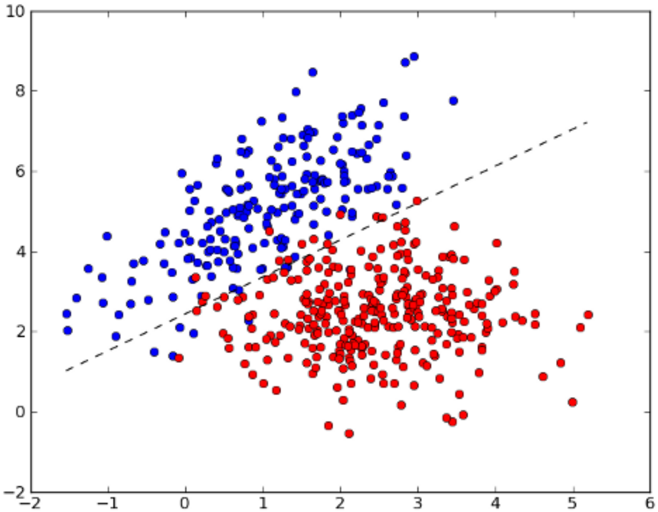
Probabilistic inspiration

Math review:

- $z = (\theta^T x)$
- $\theta^T x \geqslant 0 \iff h_\theta \geqslant 0.5$
- $\theta^T x \geqslant 0 \iff$ predict $y = 1$

Logistic Regression

Logistic Regression



Logistic (sigmoid, logit) function

$$g(z) = \frac{1}{1 + e^{-z}}$$

Logistic (sigmoid, logit) function

$$g(z) = \frac{1}{1 + e^{-z}}$$

Exercise: plot this

Cost function in logistic regression

In linear regression, we had

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost function in logistic regression

In linear regression, we had

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y)^2$$

Cost function in logistic regression

In linear regression, we had

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x), y)$$

Cost function in logistic regression

Here's a convex cost function:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Cost function in logistic regression

Here's a convex cost function:

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Exercise: Plot this (cost vs y).

Cost function in logistic regression

Here's a convex cost function:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x), y)$$

Cost function in logistic regression

Here's a convex cost function:

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = y \cdot \log(h_\theta(x)) + (1 - y) \cdot \log(1 - h_\theta(x))$$

Gradient descent

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x_j^{(i)}$$

$$\text{for } j = 1, \cdots, n$$

null hypothesis

true positive, true negative

false positive, false negative

type I error

(incorrect rejection of null hypothesis)

type II error

(failure to reject null hypothesis)

sensitivity

100% sensitivity = no false negatives

specificity

100% specificity = no false positives

Precision

$$P = \frac{TP}{TP + FP}$$

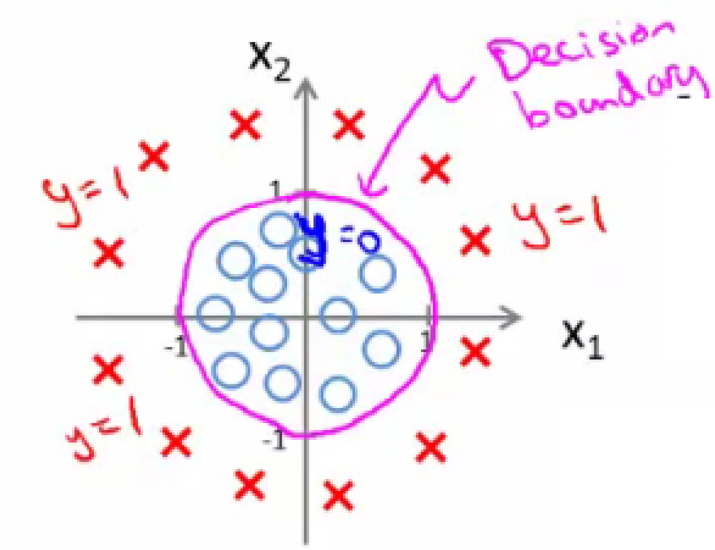
Recall

$$R = \frac{TP}{TP + FN}$$

F1 score

$$F1 = \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Non-linear decision boundaries



Non-linear decision boundaries

OvA = OvR

OvO

Non-linear decision boundaries

One vs All = One vs Rest

One vs One

One vs Rest, One vs One

- OvR (OvA): compute k classifiers
- OvO: compute $k(k-1)/2$ classifiers

The missing point: the classifiers give scores, not just in/out answers.

One vs Rest, One vs One

One vs Rest:

Accept the judgement of the classifier with the highest score.

One vs Rest, One vs One

One vs One:

Classifiers vote. Accept the class that gets the most votes. Advantage: Reduces multi-class classification to single-class classification.

Disadvantage: Classifier scores aren't necessarily comparable. For example, classes may have very different numbers of members.

Hyperparameters

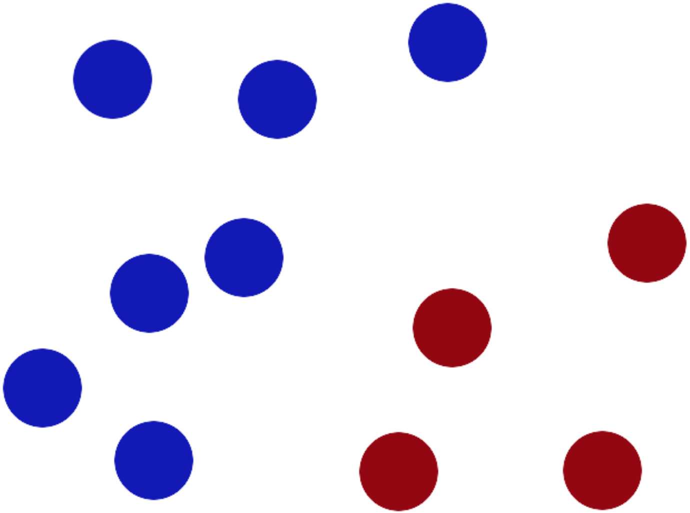
- The word hyperparameter is not well-defined.
- In most contexts, it is the parameters of the underlying distribution
- In training, we learn the parameters of the model
- We choose the hyperparameters to govern the training
- So we may want to experiment to learn the distribution parameters that best optimise our learned model's performance

Testing

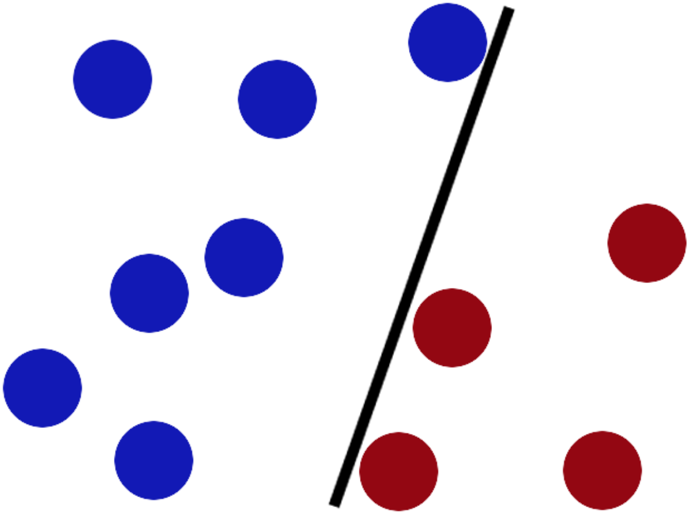
- Set aside (partition) data for testing (e.g., 70% / 30%)
- Learn on training set, test on testing set
- When searching hyperparameters, set aside again (e.g., 60% / 20% / 20%)

Support Vector Machines

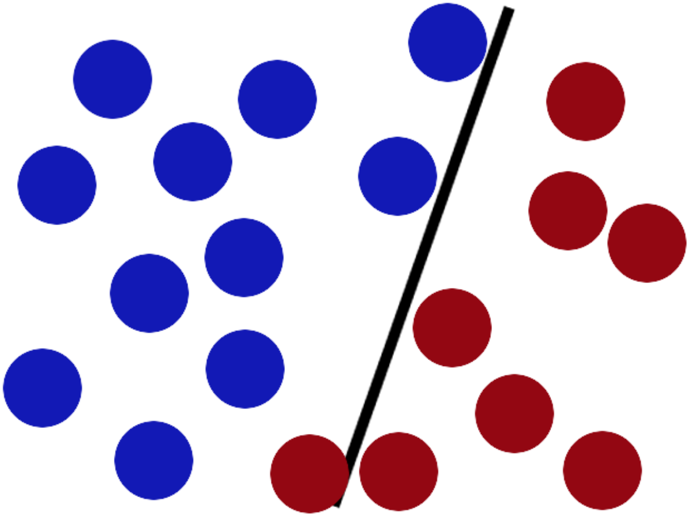
The simple explanation



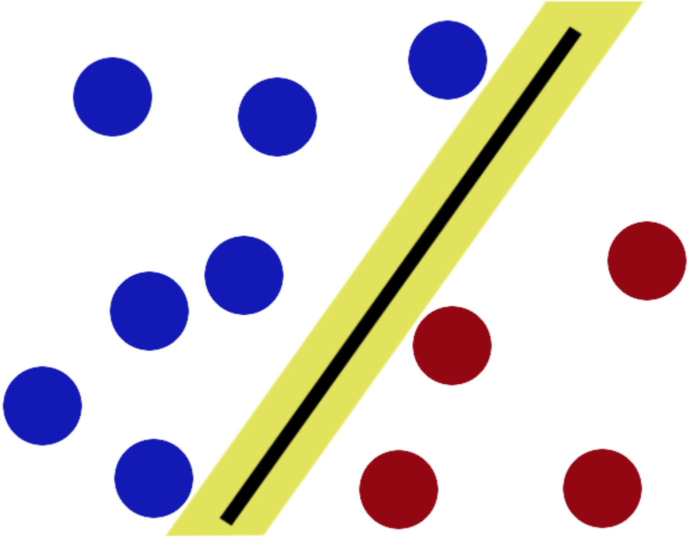
The simple explanation



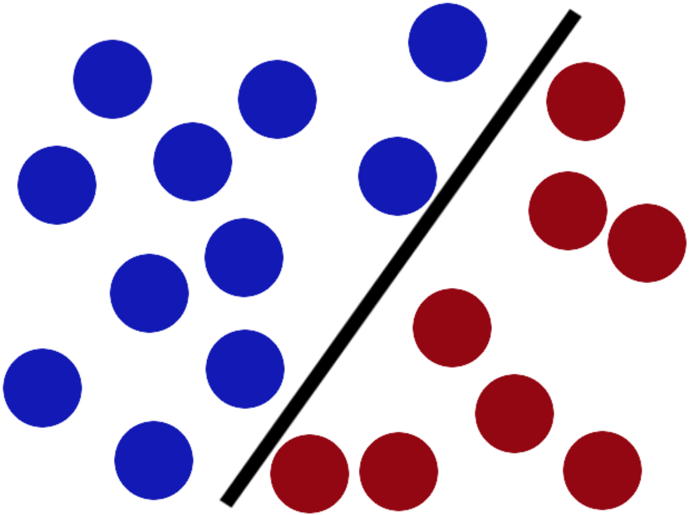
The simple explanation



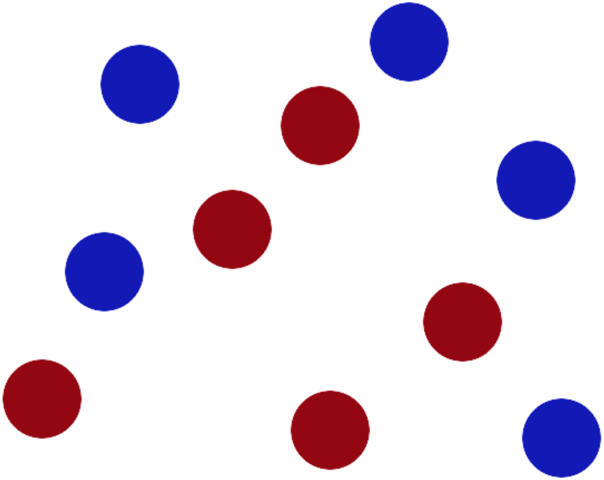
The simple explanation



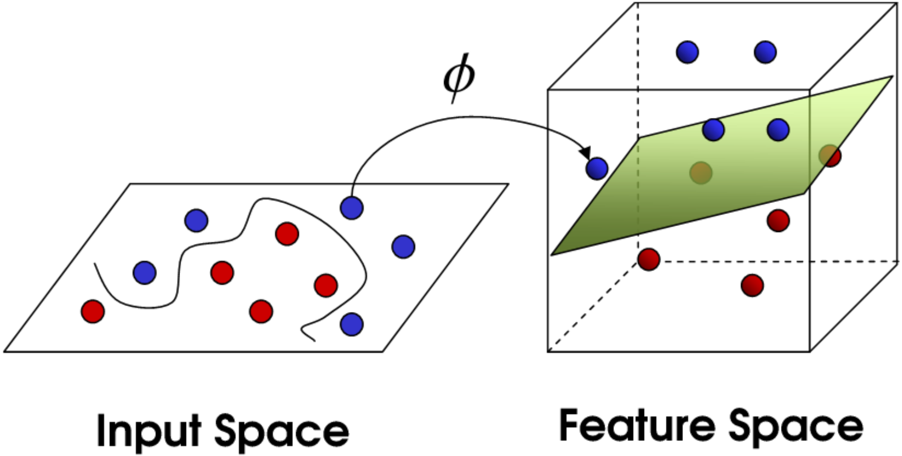
The simple explanation



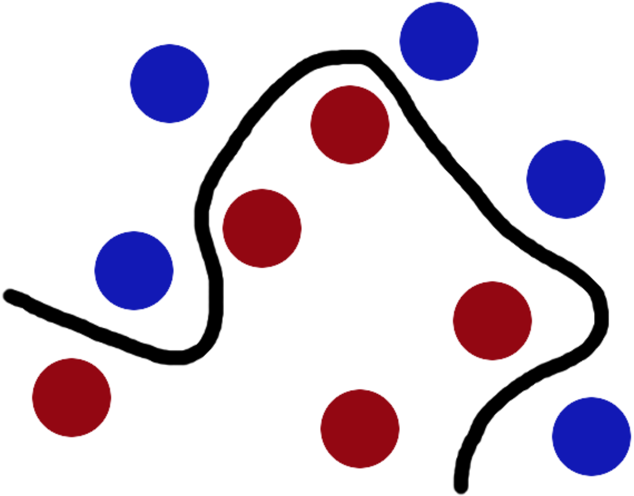
The simple explanation



The simple explanation



The simple explanation

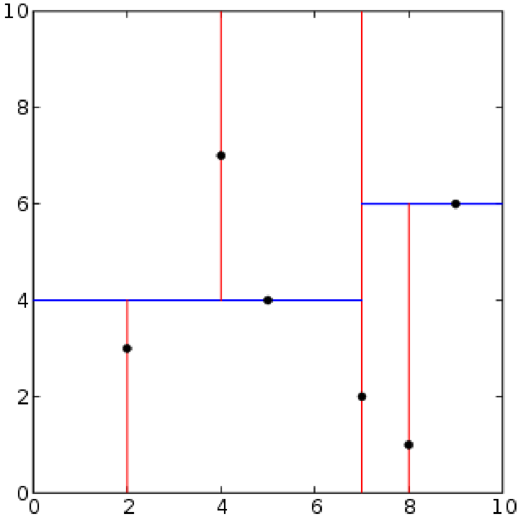


video time

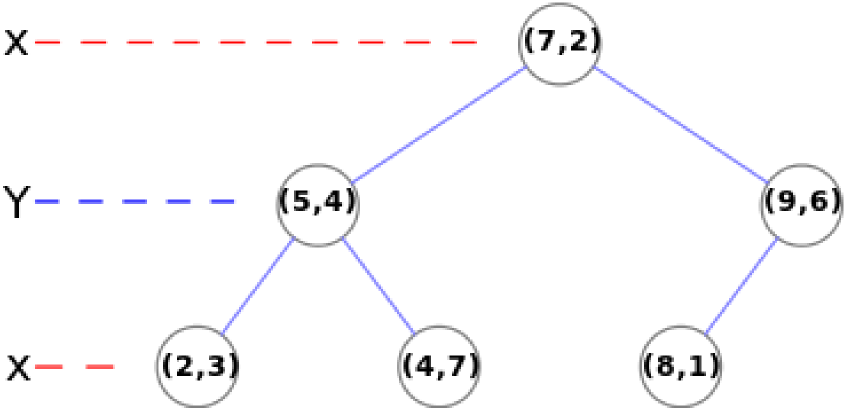
CART

Classification and Regression Trees

kd-tree



kd-tree



Random subspace methods

Draw k hyperplanes at random.

- With text, dot product, so random planes through origin
- With nearly everything else, random planes period

Result: $\log(n)$ search time

Decision Trees

Variations

- Classification tree
- Regression tree

CART = classification and regression trees

What can go wrong?

Ensemble methods

- Bagging
- Random forest
- Boosted trees *(gradient boosted trees)*
- Rotation forest

Boostrap aggregating = bagging

Boostrap

A family of statistical methods using sampling with replacement.

(Also: an example of an ensemble method.)

Boostrap aggregating = bagging

- Increase stability
- Increase accuracy
- Reduce variance
- Avoid overfitting

A type of model averaging (ensemble method).

Boostrap aggregating = bagging

- Training set D of size n
- Sample D *with replacement* to create $D_1, \ldots, D_k$ of size n
- Expect $1 - 1/e \approx 63.2\%$ repeats

Boostrap aggregating = bagging

- Training set D of size n
- Sample D *with replacement* to create $D_1, \ldots, D_k$ of size n
- Expect $1 - 1/e \approx 63.2\%$ repeats

- Train k models
- Average (regression) or vote (classification)

Boostrap aggregating = bagging

Do not confuse with

- Boosting (and AdaBoost)
- Boostrap (statistics)
- Cross validation

Random subspace method

attribute bagging = feature bagging

Random subspace method

Bagging (bootstrap aggregation) = resampling to create more data sets, train models on different samples

Attribute bagging = project to create more data sets, train models on different samples

Random forests

Combine bagging with random subspace method

Topic: Images

Signal processing

in 2 or 3 dimensions

Images

Details that can matter:

- Illumination
- White balance
- Resolution
- Camera settings (e.g., depth of field)
- Sensor noise
- Compression technology

Images

Challenges:

- Segmentation
- Area of interest detection
- Perspective shifting

Images

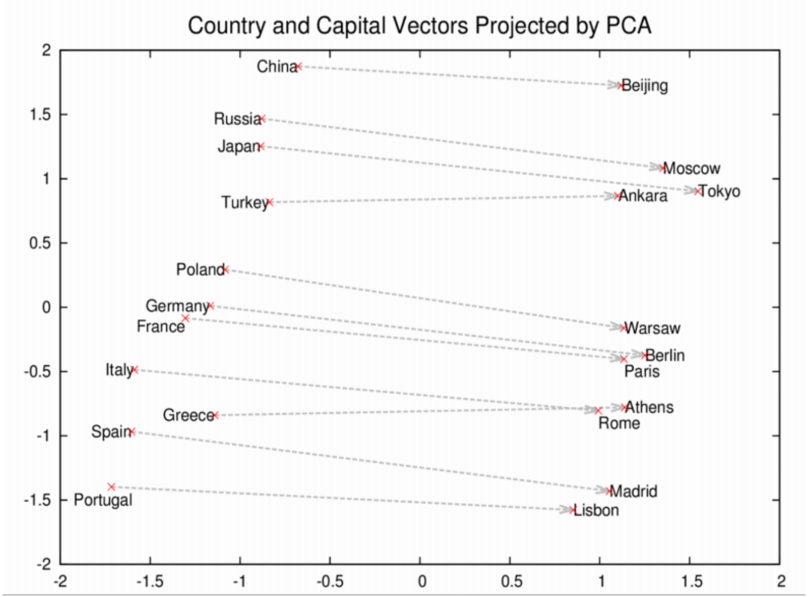
Applications:

- Agriculture: fruit ripening, automated harvesting
- Security: detecting specific people
- Security: detecting accidents (e.g., falls)
- Art: counterfeit detection
- Medicine: assisted surgery
- Image search

Images

Image search (at first):

- Texture
- Colour
- Shape, simple objects


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Athens
Rome
Spain
Portugal
Madrid
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

	Term	Similarity
	"shift"	0.933104
	"gown"	0.887743
	"skirt"	0.881672
	"bandage"	0.880162
	"midi"	0.869786

Similar to ‘dress’

Eddie Bell @ Lyst

PCA

Principle component analysis

Analyse en composantes principales

Motivation

Remember the Curse of Dimensionality?

Principle

- Linear transformations have axes
- Find them (eigenvectors of the covariance matrix)
- Pick the biggest ones

Principle

- Linear transformations have axes
- Find them (eigenvectors of the covariance matrix)
- Pick the biggest ones

Fitting an *n*-dimensional ellipsoid to the data

Uses

- Exploratory data analysis
- Compression

Also known as

- Discrete Kosambi-Karhunen–Loève transform (KLT) (signal processing)
- Hotelling transform (multivariate quality control)
- Proper orthogonal decomposition (POD) (ME)
- Singular value decomposition (SVD), Eigenvalue decomposition (EVD) (linear algebra)
- Etc.

History

- Invented by Karl Pearson in 1901
- Invented (again) and named by Harold Hotelling in 1930's
- Also known as...

Also known as

- It’s a long list, every field uses a different name…

In addition to PCA…

There are newer methods that are sometimes better.

- Sometimes they are faster.
- Sometimes the projection makes more sense.

t-SNE is the most common you'll encounter.

Topic: Face Recognition

Eigenfaces

- Sirovich and Kirby (1987)
- Turk and Pentland (1991)

Turk, Matthew A and Pentland, Alex P. Face recognition using eigenfaces. Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on 1991.

Eigenfaces

Want: a low-dimensional representation of a face

Plan: cluster simplified faces

Eigenfaces

Viewed as compression:

- Use PCA on face images to form a set of basis features
- Use eigenpictures to reconstruct original faces

Eigenfaces

Eigenfaces algorithm

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a random vector with observations $x_i \in \mathbb{R}^d$.

Compute

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

OpenCV

Eigenfaces algorithm

Compute the covariance matrix S:

$$
\begin{aligned}
S_{i,j} &= \mathbf{Cov}(x_i, x_j) \\
&= \mathbf{E}[(x_i - \mu_i)(x_j - \mu_j)^T]
\end{aligned}
$$

$$S = (S_{i,j})$$

Eigenfaces algorithm

Compute the eigenvectors of S:

$$Sv_i = \lambda_i v_i \qquad i = 1, 2, \ldots, n$$

Sort the eigenvectors in decreasing order.

We want the k principal components, so take the first k.

Eigenfaces algorithm

Compute the eigenvectors of S:

$$Sv_i = \lambda_i v_i \qquad i = 1, 2, \ldots, n$$

Sort the eigenvectors in decreasing order.

We want the k principal components, so take the first k.

This is PCA.

Eigenfaces algorithm

The k principal components of the observed vector x are then given by

$$y = W^T(x - \mu)$$

where

$$W = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_k \\ | & | & & | \end{bmatrix}$$

Eigenfaces algorithm

The reconstruction from the PCA basis is then

$$x = Wy + \mu$$

Eigenfaces algorithm

So the plan is this:

- Project all training samples in the PCA subspace
- Project the query into the PCA subspace
- Find the nearest neighbour to the projected query image among the projected training images

Eigenfaces algorithm

Eigenfaces algorithm

Some advantages:

- Easy, relatively inexpensive
- Recognition cheaper than preprocessing
- Reasonably large database possible

Eigenfaces algorithm

Some problems:

- Need controlled environment
- Needs straight-on view
- Sensitive to expression changes
- If lots of variance is external (e.g., lighting)...

Topic: Handwriting Recognition

Choices

- Online
- Offlne

Choices

- Get path information
- Get time data
- Get pressure information
- Only get image

Major techniques

- Clustering (not great performance)
- SVM (until 2006 or so)
- Convolutional neural networks

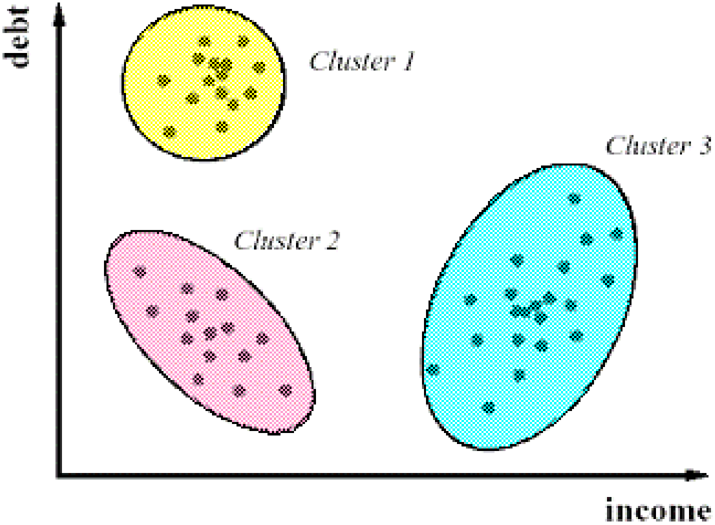
Clustering

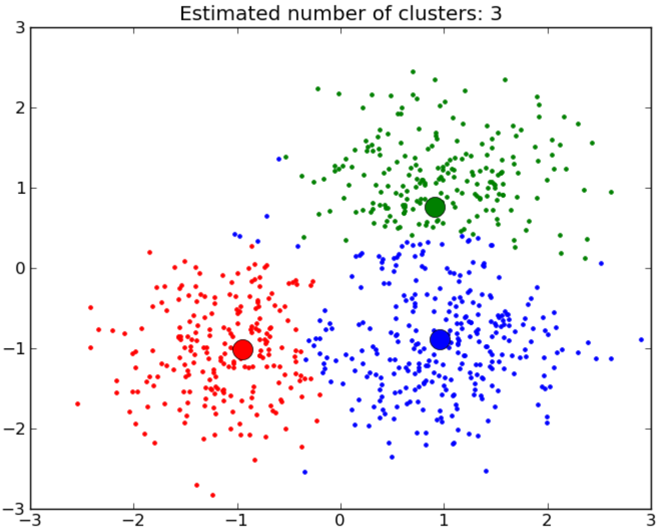
The Problem

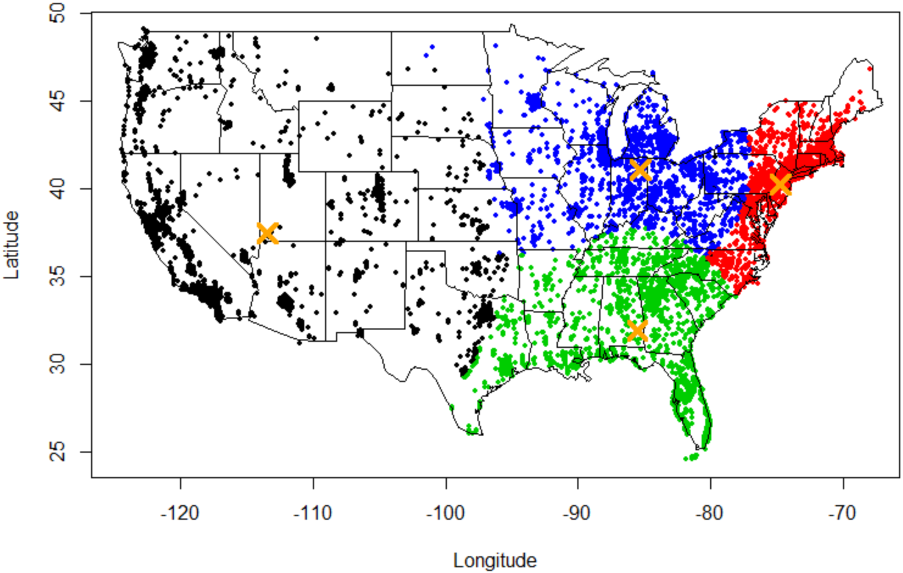
Have points $d = \{d_1, \ldots, d_n\}$.

Have number of clusters k.

Want: an assignment of points to clusters

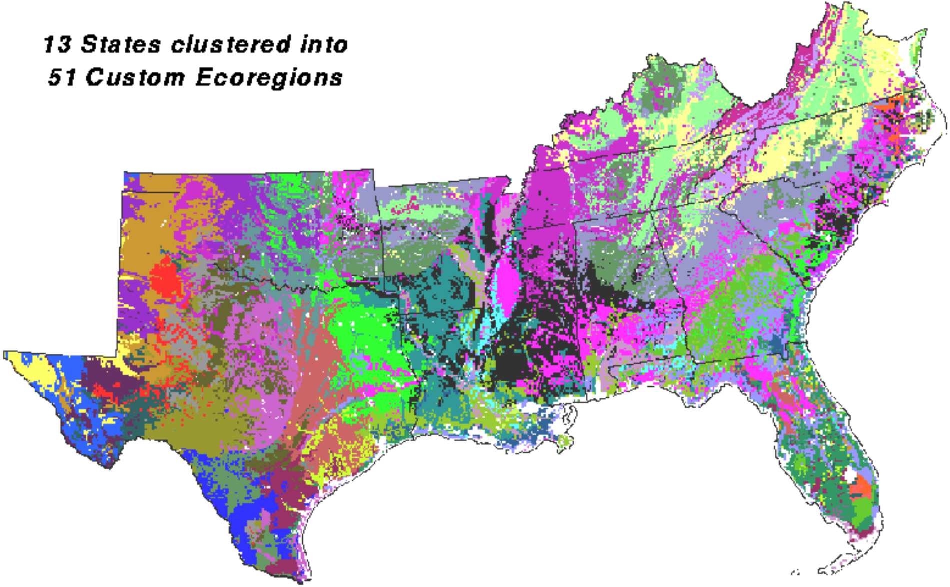

debt
Cluster 1
Cluster 2
Cluster 3
income


Estimated number of clusters: 3
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3



Diva / Beapp

13 States clustered into
51 Custom Ecoregions



The Algorithm

1. Assign points to clusters at random
2. Repeat until stable:
   1. Compute centroids of each cluster
   2. Assign points to nearest centroid

Cost function

$$\text{cost} = \sum_i \sum_j |x_j - \mu_i|$$

Silhouette coefficient

Points $d = \{d_1, \ldots, d_n\}$

Clusters $K = \{c_1, \ldots, c_k\}$.

Cluster c_{d_i} is the centroid of d_i.

Silhouette coefficient

Points $d = \{d_1, \ldots, d_n\}$

Clusters $K = \{c_1, \ldots, c_k\}$.

Cluster c_{d_i} is the centroid of d_i.

Let a_i be the average dissimilarity of d_i to all points in its cluster.

Let b_i be the least average dissimilarity of d_i to any cluster other than k_{d_i}

Silhouette coefficient

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

Silhouette coefficient

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

$$s_i = \begin{cases} 1 - a_i/b_i & \text{if } a_i < b_i \\ 0 & \text{if } a_i = b_i \\ b_i/a_i - 1 & \text{if } a_i > b_i \end{cases}$$

Silhouette coefficient

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

$$s_i = \begin{cases} 1 - a_i/b_i & \text{if } a_i < b_i \\ 0 & \text{if } a_i = b_i \\ b_i/a_i - 1 & \text{if } a_i > b_i \end{cases}$$

So $s_i \in [-1, 1]$

Silhouette coefficient

s_i near 1 $\iff$ d_i well clustered

s_i near 0 $\iff$ d_i on the border between two clusters

s_i near -1 $\iff$ d_i poorly clustered

Silhouette coefficient

Consider $\overline{s_i}$ over $i \in c_j$ for cluster c_j

Silhouette coefficient

Consider $\overline{s_i}$

video time

Topic: Anomaly Detection

Introduction to Anomaly Detection

- Supervised
- Unsupervised

Introduction to Anomaly Detection

Supervised anomaly detection:

- Training data: normal, abnormal
- Train a classifier

So reduced to existing problem of supervised classification.

Introduction to Anomaly Detection

Unsupervised anomaly detection:

- Mostly, this is clustering
- Increasingly, this is neural networks in advanced applications

Applications:

- Intrusion detection (physical or electronic)
- Fraud detection
- Health monitoring (people, animals, machines)

Introduction to Anomaly Detection

Techniques:

- Density: kNN, local outlier factor
- SVM
- Clustering: k-Means

kNN techniques and variations

- Voronoi diagrams
- aNN

LOF

- Measure average density using kNN
- Points with low local density are suspect outliers
- There is no good thresholding technique

k-Means

Examples

ping times

httpd response times

Examples

single/multiple host access abuse (DOS/DDOS)

Examples

bank card fraud

Examples

spam

Topic: Anomaly Detection (not time)

Introduction to Anomaly Detection

- Supervised
- Unsupervised

Introduction to Anomaly Detection

Supervised anomaly detection:

- Training data: normal, abnormal
- Train a classifier

So reduced to existing problem of supervised classification.

Introduction to Anomaly Detection

Unsupervised anomaly detection:

- Mostly, this is clustering
- Increasingly, this is neural networks in advanced applications

Introduction to Anomaly Detection

Applications:

- Intrusion detection (physical or electronic)
- Fraud detection
- Health monitoring (people, animals, machines)

Introduction to Anomaly Detection

Techniques:

- Density: kNN, local outlier factor
- SVM
- Clustering: k-Means

kNN techniques and variations

- Voronoi diagrams
- aNN

k-Means

Local Outlier Factor

Local Outlier Factor (LOF)

- Measure average density using kNN
- Points with low local density are suspect outliers
- There is no good thresholding technique

Local Outlier Factor (LOF)

Let a be an object (point) in the set of samples.

Let $N_k(a)$ be the set of k nearest neighbours to a.

Define the k-distance from a:

$$d_k(a) = \max_{p \in N_k(A)} d(a, p)$$

Local Outlier Factor (LOF)

Define now the reachability distance:

$$r_k(a, b) = \max(d_k(a), d(a, b))$$

In otherwords, r_k is the distance between two points, but is no less than the k-distance.

So all the points in $N_k(a)$ are considered equally r_k distant from a.

Math note: r_k is not a true distance function.

Local Outlier Factor (LOF)

Define the *local reachability density* of object *a* by

$$\mathrm{lrd}(a) = \frac{1}{\left(\frac{\sum_{b \in N_k(a)} r_k(a,b)}{|N_k(A)|}\right)} = \left(\frac{|N_k(A)|}{\sum_{b \in N_k(a)} r_k(a,b)}\right)$$

This is the (inverse of the) average reachability distance of the k nearest neighbours.

Local Outlier Factor (LOF)

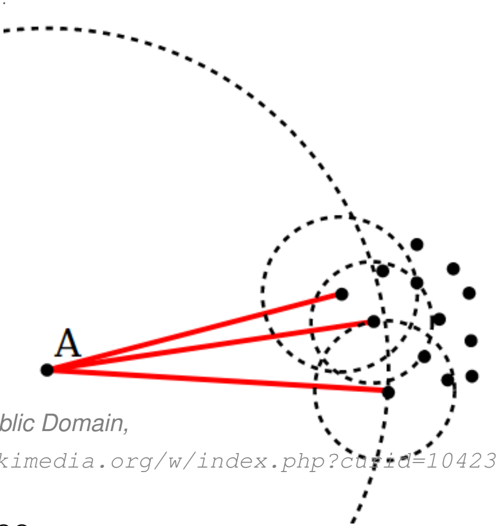
The

$$\mathrm{LOF}_k(a) = \left(\frac{\sum_{b \in N_k(a)} \frac{\mathrm{lrd}(b)}{\mathrm{lrd}(a)}}{|N_k(a)|} \right) = \left(\frac{\sum_{b \in N_k(a)} \mathrm{lrd}(b)}{|N_k(a)| \quad \mathrm{lrd}(a)} \right)$$

Interpretation:

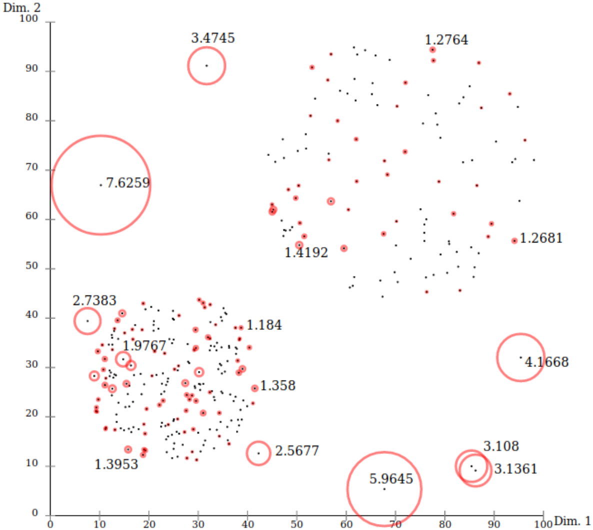
- 1 indicates a point is comparable to its neighbours
- < 1 indicates more densely packed than its neighbours
- > 1 indicates more sparsely packed than its neighbours

Local Outlier Factor (LOF)



By Chire - Own work, Public Domain,
`https://commons.wikimedia.org/w/index.php?curid=10423954`

Local Outlier Factor (LOF)

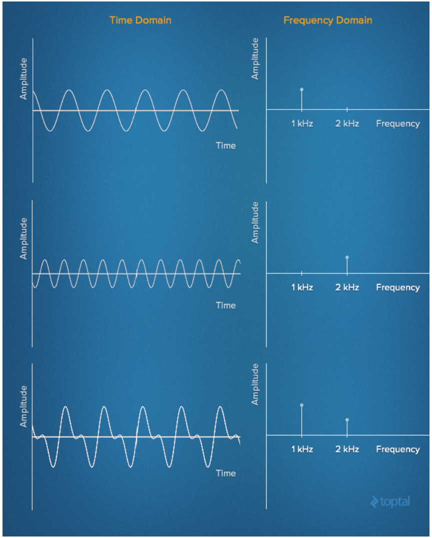


Local Outlier Factor (LOF)

Advantages: intuitive, often works well (e.g., intrusion detection)

Disadvantages: fails at higher dimension (curse of dimensionality), hard to interpret

Topic: Music



http://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

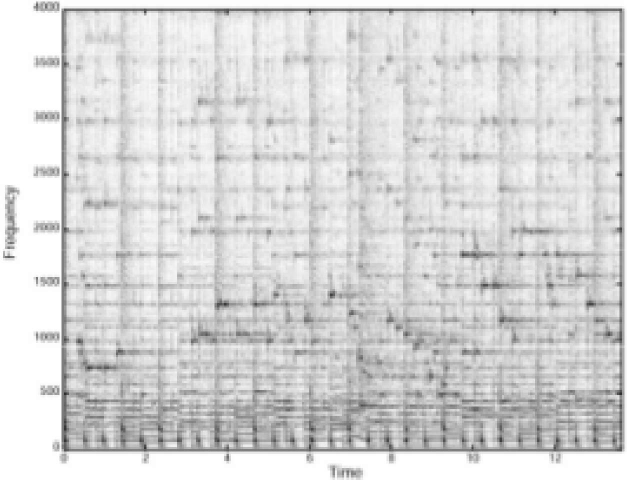


Fig. 1A - Spectrogram

https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

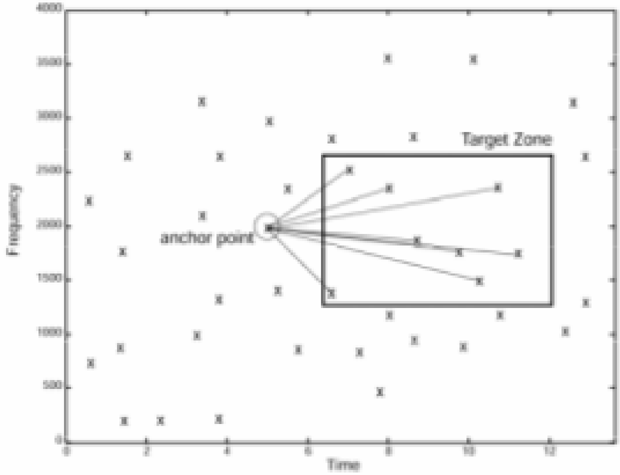


Fig. 1C - Combinatorial Hash Generation

https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

Topic: Time Series

This is hard, but it depends on your goals. And on context.

Introduction to time series

Definition (discrete time series):

$$\{s_t \mid t \in \mathbb{R}^+ \wedge s \in \mathbb{R}\}$$

(though s in any vector space is fine)

Examples domains:

- Weather
- Economics
- Industry (e.g., factories)
- Medicine
- Web
- Biological processes

Introduction to time series

Why?

- Predict
- Control
- Understand
- Describe

Introduction to time series

Some strategies:

- Differencing:
  $$y'_t = y_t - y_{t-1}$$
- Second-order differencing:
  $$y''_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2}$$

Introduction to time series

Some strategies:

- Clustering
- Hidden Markov Models (HMM)
- Recurrent neural networks (RNN)
- Autoregressive integrated moving average (ARIMA)
  - Generalisation of autoregressive moving average (ARMA) model
  - Regress on series' own lag

One model:

$$s_t = g(t) + \phi_t$$

where

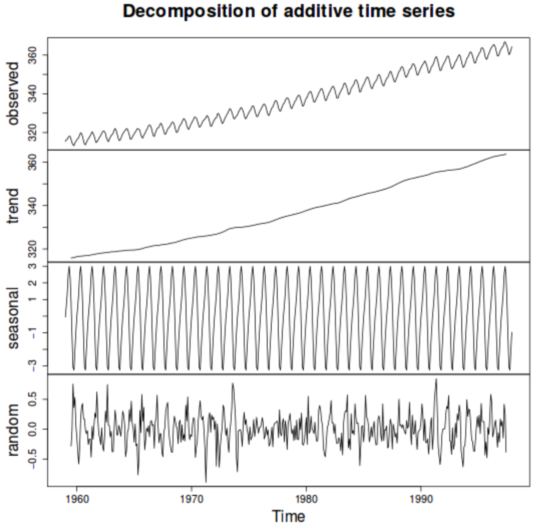
- $g(t)$ is deterministic: signal (or trend)
- ϕ_t is stochastic noise

Introduction to time series

Variation types:

- Trend (g)
- Seasonal effect (g)
- Irregular fluctuation (residuals: ϕ)

Introduction to time series



http://www.ulb.ac.be/di/map/gbonte/ftp/time_ser.pdf

Some easy things to try

- Introduce features to break out seasonality
- Introduce lags as features
- Some domain-specific transformation

Hidden Markov Models

“simplest dynamic Bayesian network”

Markov Chains

A **Discrete time Markov chain (DTMC)** is a random process that undergoes state transitions.

Markov Chains

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & xnn \end{bmatrix} \begin{bmatrix} v_1^{(i)} \\ v_2^{(i)} \\ \vdots \\ v_n^{(i)} \end{bmatrix} = \begin{bmatrix} v_1^{(i+1)} \\ v_2^{(i+1)} \\ \vdots \\ v_n^{(i+1)} \end{bmatrix}$$

Markov Chains

$$Xv_i = v_{i+1}$$

Markov Chains

Examples:

- Random walks
- Weather (first approximation in many places)
- Thermodynamics
- Queuing theory (so also telecommunications)
- Spam

Markov Chains

Properties:

- Stochastic process
- Memoryless (“Markov property”)

HMM’s

- State is not visible
- Output of state is visible

Examples: noisy sensor, medical diagnosis

HMM's

What we have:

- State space $S = \{s_1, \ldots, s_n\}$
- Observation space $O = \{o_1, \ldots, o_k\}$
- Transition matrix A of size $n \times n$
- Emission matrix B of size $n \times k$
- Initial state probabilities $\pi = \{\pi_1, \ldots, \pi_n\}$
- A sequence of observations $X = \{x_1, \ldots x_T\}$

Here

- $y_t = i \iff$ observation at time t is o_i
- $\Pr(x_1 = s_i) = \pi_i$

We want the sequence of states $X = \{x_1, \ldots, x_T\}$.

HMM’s

Some pointers to learn more about HMM:

- Forward-Backward Algorith
- Viterbi Decoding
- Baum-Welch Algorithm

Topic: Recommendation

Definition

Given data about a user, his environment, and some items of interest (*training data*), determine items to recommend.

Definition

Given data about a user, his environment, and some items of interest (*training data*), determine items to recommend.

We don't have to find the $\max k$.

It's enough to find k within some $\max n$.

Examples

- Amazon
- Google News (or Le Monde)
- Facebook
- Medical testing
- App Store / Google Play
- Youtube
- Advertising
- Netflix, last.fm, Spotify, Pandora, . . .
- Browser (URL recommendations)
- Search

Client Value Proposition

- Find opportunities
- Reduce choice
- Explore options
- Discover long tails
- Recreation

Provider Value Proposition

- Offer a unique or additional service (beyond competitors)
- Customer trust and loyalty
- Increase sales, CTR, conversions
- Better understand customers

Recommendation

Content-based filtering *(filtrage basée sur le contenu)*	More things similar to what I like
Collaborative filtering *(filtrage collaboratif)*	More of what other people who like what I like like
Knowledge-based filtering *(filtrage basée sur connaissance)*	More of what I need.

Content-based filtering

More things similar to what I like
Plus de ce qui ressemble à ce que j'aime

Advantages

- yes! No need for community
- yes! Possible to compare items

Disadvantages

- no Understand content
- yes Cold start problem
- no Serendipity

Collaborative filtering

More of what other people who like what I like like
Plus de ce que d'autres qui aiment ce que j'aime aiment

Advantages

yes! No need to understand content

yes! Serendipity

yes! Learn market

Disadvantages

no User feedback

yes Cold start problem (users)

yes Cold start problem (items)

Knowledge-based filtering

More of what I need
Plus de ce qu'il faut

Advantages

- yes! Deterministic
- yes! Certainty
- no! Cold start problem
- yes! Market knowledge

Disadvantages

- yes Studies to bootstrap
- yes Static model, doesn't learn from trends

Utility Matrix

- Users (utilisateurs)
- Items (objets)

Utility Matrix

- Users (utilisateurs)
- Items (objets)

The goal is to fill in the blanks.

	I_1	I_2	I_3	I_4	I_5
U_1	1				
U_2			1	1	1
U_3		1		1	1

Example: books sales at Amazon.

But thousands or millions of columns and rows.

Utility Matrix

- Users (utilisateurs)
- Items (objets)

The goal is to fill in the blanks.

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

Example: film advice at Netflix.

But thousands or millions of columns and rows.

Utility Matrix

How do we make the matrix?

- Ask users
- Observe users

That's usually expensive. . .

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Films :

Content: acters, directors, year (decade, etc.), length

Collaborative: seen, opinion (1–5), when seen relative to release

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Books:

Content : authers, genre, year (decade, etc.), number of pages, content (very difficult)

Collaborative: read, opinion (1–5), how read

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

News:

Content : source, section, TF-IDF word vectors

Collaborative:

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Images :

Content:

Collaborative:

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Also: user profile, user behavior

Mathematics

Vectors

Similarity

Similarity : Jaccard Index

or: *Indice de Jaccard, Jaccard similarity coefficient*

Similarity:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Similarity : Jaccard Index

Similarity:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Distance:

$$J_\delta(A, B) = 1 - J(A, B)$$

cosine similarity

or: *mesure cosinus, Similarité cosinus*

Similarity:

$$\cos\theta = \frac{A \cdot B}{\| A \| \| B \|}$$

cosine similarity

Similarity:

$$S_C(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

cosine similarity

Similarity:

$$S_C(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

Distance:

$$D_C(A, B) = 1 - S_C(A, B)$$

cosine similarity

Similarity:

$$S_C(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

Distance:

$$D_C(A, B) = 1 - S_C(A, B)$$

We only consider non-empty components in the vector.

Texts: TF-IDF

- Vectors of word frequencies
- Frequency $\not\Rightarrow$ significance

Texts: TF-IDF

- Vectors of word frequencies
- Frequency $\not\Longrightarrow$ significance
- Term Frequency - Inverse Document Frequency

Texts: TF-IDF

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \qquad IDF_I = \log_2\left(\frac{N}{n_i}\right)$$

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i$$

with :

$$f_{ij} = \text{frequency of word } i \text{ in document } j$$
$$N = \text{number of documents}$$
$$n_i = \text{number of documents in which we find word } i$$

Texts: TF-IDF

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \qquad IDF_I = \log_2 \left(\frac{N}{n_i} \right)$$

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i$$

with :

$$f_{ij} = \text{frequency of word } i \text{ in document } j$$
$$N = \text{number of documents}$$
$$n_i = \text{number of documents in which we find word } i$$

IDF is a mesure of how much information a word carries

TF-IDF tells us which words best characterise a document

Texts: TF-IDF

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \qquad IDF_I = \log_2\left(\frac{N}{n_i}\right)$$

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i$$

with :

$$\begin{aligned} f_{ij} &= \text{frequency of word } i \text{ in document } j \\ N &= \text{number of documents} \\ n_i &= \text{number of documents in which we find word } i \end{aligned}$$

IDF is a mesure of how much information a word carries

TF-IDF tells us which words best characterise a document
Variation: boolean, log, stop word filtering

Content-Based Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

More things similar to what I like

Plus de ce qui ressemble à ce que j'aime

Content-Based Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

More things similar to what I like

Plus de ce qui ressemble à ce que j'aime

Then, we can cluster (*regroupement*, *partitionnement de données*), etc.

Content-Based Filtering

Based on item profiles

- More stable (in principle)
- $O(n^2)$ (but often less, items often aren't categorised together)
- Can reduce to threshold
- Can pre-calculate, queries become faster

Collaborative Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3		$\boxed{4}$	$\boxed{2}$	
$\boxed{U_2}$			5	1	4
U_3		2		5	1

More of what other people who like what I like like

Plus de ce que d'autres qui aiment ce que j'aime aiment

Collaborative Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

User profile

Collaborative Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	
U_2			5	1	4
U_3		2		5	1

Item profile

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

But remember: 2 items being similar $\not\equiv$ 2 users similar.

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

But remember: 2 items being similar $\not\equiv$ 2 users similar.

Thought experiment: consider comparing people vs comparing objects.

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

To estimate $m_{u,i}$,

- Find k users like U_u
- Find k items like I_i

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

We have to compute the entire line (or the part which is likely to be important)

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

Once we've computed U_u, the other k users lets us take a shortcut.

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

For I_i, we have to compute most of the I_j before we can fill in a single line. But item-item filters are often more reliable.

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

In any case, we can mostly precompute in advance.

Utility Matrix

The matrix is sparse.

$\Longrightarrow$ clustering $\Longrightarrow$ reduced matrix

Utility Matrix

The matrix is sparse.

$\implies$ clustering $\implies$ reduced matrix

Estimate on the reduced matrix, then take items and users as representative for the cluster.

Amazon : Item-to-Item Collaborative Filtering

Observations :

Clustering is expensive, reduces quality

Amazon : Item-to-Item Collaborative Filtering

Observations :

Dimension reduction reduces quality

Amazon : Item-to-Item Collaborative Filtering

Observations :

Users interact with very few items

Amazon : Item-to-Item Collaborative Filtering

Observations :

Rapid response desirable

Amazon : Item-to-Item Collaborative Filtering

Scales independent of the number of users or of items

- Online
- Offline

G. Linden, B. Smith, J. York, *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, Internet Computing (7, 1), 22 Jan 2003.

Amazon : Item-to-Item Collaborative Filtering

Offline (Precomputation)

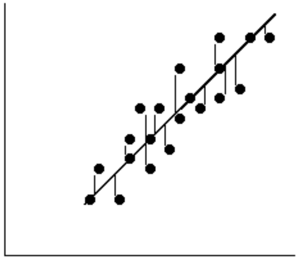
for *each item* I_1 *to sell* **do**
  for *each user C who has purchased* I_1 **do**
    for *each item* I_2 *bought by C* **do**
      (I_1, I_2)++
    end
  end
  for *each item* I_2 **do**
    $S_{I_1,I_2} \leftarrow S(I_1, I_2)$
  end
end

Slope One

Linear regression on user opinions (ratings)

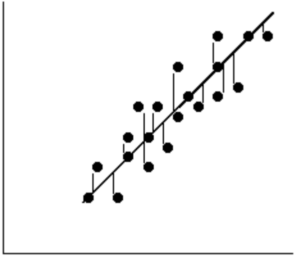
Daniel Lemire and Anna Maclachlan, *Slope One Predictors for Online Rating-Based Collaborative Filtering*,Proceedings of SIAM Data Mining (SDM) 2005.

Slope One : Regression



http://www.upa.pdx.edu/IOA/newsom/pa551/Image255.gif

Slope One : Regression



$$\min \sum (y_i - (ax_i + b))^2$$

http://www.upa.pdx.edu/IOA/newsom/pa551/Image255.gif

Slope One : algorithm

Offline :

for *chaque* I_i, I_j **do**

$\mathcal{U} \leftarrow \{$users who have expressed an opinion on $I_i, I_j\}$

$$\text{dev}_{i,j} \leftarrow \frac{1}{\parallel \mathcal{U} \parallel} \sum_{u \in \mathcal{U}} (r_u(i) - r_u(j))$$

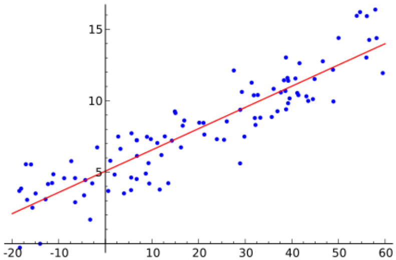
end

Online (for u) :

$\mathcal{V} \leftarrow \{j \mid u$ has expressed an opinion on $I_j\}$

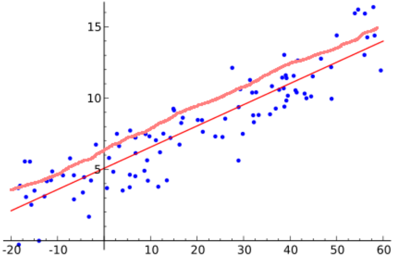
$$r_u(i) \leftarrow \frac{1}{\parallel \mathcal{V} \parallel} \sum_{u \in \mathcal{V}} (\text{dev}_{i,j} - r_u(j))$$

Slope One : Regression



"Linear regression" by Sewaqu - Own work. Licensed under Public domain via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Linear_regression.svg#mediaviewer/File:Linear_regression.svg`

Slope One : Regression



Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$M = U \Sigma V^*$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_1 & \cdots & a_m \end{pmatrix} \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = \text{scalar}$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} \begin{pmatrix} b_1 & \cdots & b_n \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix}$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} \end{pmatrix} \begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ b_{2,1} & \cdots & b_{2,n} \\ b_{3,1} & \cdots & b_{3,n} \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix}$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,k} \end{pmatrix} \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{k,1} & \cdots & c_{k,n} \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix}$$

Challenges

- How do we measure success?
- What are our features?

Clustering

- kNN
- Curse of Dimensionality
- Scalability

Clustering

- kNN *k*-Nearest Neighbor
- Curse of Dimensionality
- Scalability 10^7 clients, 10^6 objets

Topic: Natural Language Processing

Linear Programming

$$\text{Maximize} \quad c^T x$$
$$\text{subject to} \quad Ax \leqslant b$$

Warning: this is exclusively ANN (and probably LLM) now

Summarising Text

- Abstractive (used to be hard, until 2018 or so)
- Extractive (select sentences)

Summarising Text

Challenge problem (cf. greedy solutions):

The cat is in the kitchen.

The cat drinks the milk.

The cat drinks the milk in the kitchen.

Summarising Text

- Sentence selection
- Use n-grams
- Stemming
- Stop words
- Prune short sentences

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarising Text

Outline:

- ILP *(optimisation linéaire en nombres entiers)*
- Maximum coverage model

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarising Text

ILP in canonical form:

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \leqslant b \\ & x \geqslant 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

ILP in standard form:

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{subject to} \quad & Ax + s = b \\ & s \geqslant 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarising Text

ILP in standard form:

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{subject to} \quad & Ax + s = b \\ & s \geqslant 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

This is NP hard.

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarising Text

ILP in standard form:

$$
\begin{aligned}
\text{Maximize} \quad & c^T x \\
\text{subject to} \quad & Ax + s = b \\
& s \geqslant 0 \\
& x \in \mathbb{Z}^n
\end{aligned}
$$

Discussion: linear vs integer programming.

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarising Text

Let

$$\begin{aligned} c_i &: \text{ presence of concept } i \text{ in summary} \\ w_i &: \text{ weight associated with } c_i \\ l_i &: \text{ length of sentence } i \\ s_j &: \text{ presence of sentence } j \text{ in summary} \\ L &: \text{ summary length limit} \\ Occ_{ij} &: \text{ occurence of } c_i \text{ in } s_j \end{aligned}$$

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarisation

$$
\begin{aligned}
\text{Maximize} \quad & \sum_i w_i c_i \\
\text{subject to} \quad & \sum_j l_j s_j \leqslant L \\
& s_j Occ_{ij} \leqslant c_i, \qquad \forall i, j \\
& \sum_j s_j Occ_{ij} \geqslant c_i \qquad \forall i \\
& c_j \in \{0, 1\}, \qquad \forall i \\
& s_j \in \{0, 1\}, \qquad \forall j
\end{aligned}
$$

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Summarising Text

Notes:

- Selecting a sentence selects all concepts it contains
- Selecting a concept requires it be in at least one sentence
- $s_j Occ_{ij} \leqslant c_i, \ \forall i, j \ \Rightarrow$ no concept-less sentences

Dan Gillick, Benoit Favre, A Scalable Global Model for Summarization, 2009

Sentiment Analysis

Many variations:

- Entire documents using computational linguistics
- Manually crafted lexicons

Sentiment Analysis

Techniques

- Template instantiation (requires domain knowledge)
- Passage extraction

Sentiment Analysis

- Extract “opinion sentences” based on the presence of a predetermined list of product features and adjectives.
- Evaluate the sentences based on counts of positive vs negative polarity words (as determined by the Wordnet algorithm)

Hu and Lieu, Mining and Summarizing Customer Reviews, 2004

Sentiment Analysis

- Extract “opinion sentences” based on the presence of a predetermined list of product features and adjectives.
  - “The food is excellent.”
  - “The food is an excellent example of how not to cook.”
- Evaluate the sentences based on counts of positive vs negative polarity words (as determined by the Wordnet algorithm)

Hu and Lieu, Mining and Summarizing Customer Reviews, 2004

Sentiment Analysis

- Extract "opinion sentences" based on the presence of a predetermined list of product features and adjectives.
- Evaluate the sentences based on counts of positive vs negative polarity words (as determined by the Wordnet algorithm)

The good: fast, no training data, decent prediction.

The bad: fails on multiple word sense, non-adjectives; sensitive to context.

Hu and Lieu, Mining and Summarizing Customer Reviews, 2004

Sentiment Analysis

Words aren't enough.

- "unpredictable plot" vs "unpredictable performance"

This all changed in 2017, *Attention is All You Need*, Ashish Vaswani et al., Google

Turney, Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, 2002

Perceptron

Gradient Descent

Perceptron

- Supervised
- Binary linear classifier
- Online learning

Perceptron

Beginnings:

- One of first artificial neural networks (ANN's)
- Developed in 1957 by Frank Rosenblatt, Cornell University Aeronautical Laboratory
- First implemented in software (IBM 704)
- Intended to be a machine
- Designed for image recognition

Perceptron

Controversy:

- 1958, press conference, NYT
- Rosenblatt too optimistic
- 1969, Minsky and Papert

Perceptron

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

The algorithm terminates if and only if the data is linearly separable.

Perceptron

- Also called “single-layer perceptron”
- Not related to multi-layer perceptron
- Feedforward neural network

Perceptron

The training data is

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

Perceptron

The training data is

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

$x_{j,i}$ is the value of the *i*th feature of the *j*th training input vector

$x_{j,0} = 1$ (the bias is thus w_0 rather than b)

w_i is the weight on the *i*th feature

Perceptron

Start by setting the weight vector to zero (or to some small random noise).

For each input vector j in turn:

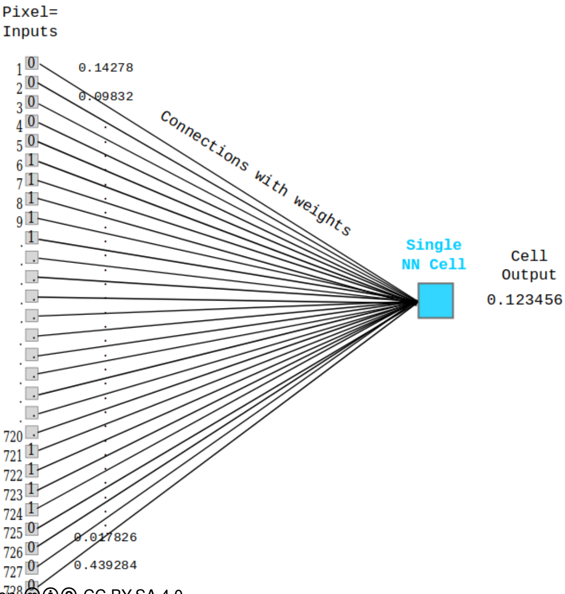
1. Compute $\hat{y} = f(w \cdot x)$
2. Update the weights: $w_i = w_i + (y_j - \hat{y}_j)x_{j,i}$

Multiclass Perceptron

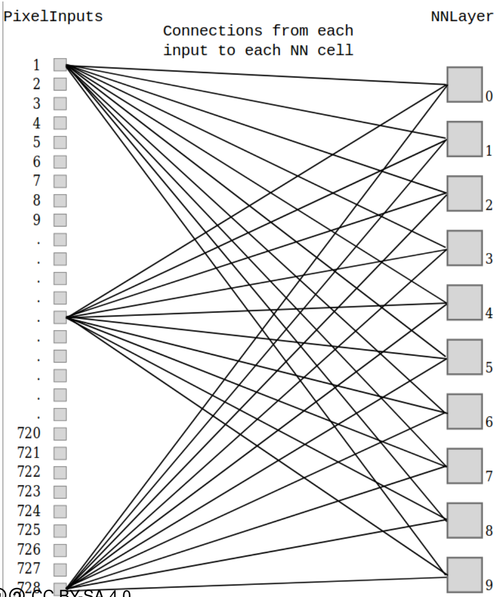
$$\hat{y} = \text{argmax}_y f(x, y) \cdot w$$

$$w = w + f(x, y) - f(x, \hat{y})$$

Multiclass Perceptron: Example



Multiclass Perceptron: Example



Perceptron History

- Perceptron is an example of SPR for image recognition
- Initially very promising
- IBM 704 (software implementation of algorithm)
- Mark 1 Perceptron at the Smithsonian Institution
- 400 photocells randomly connected to neurons.
- Weights encoded in potentiometers, updated during learning by electric motors

Frank Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386—408, 1958. doi:10.1037/h0042519

Perceptron History

- Minksy and Papert showed perceptrons are incapable of recognizing certain classes of images
- AI community mistakenly over-generalized to all NN's
- So NN research stagnated for some time
- Single layer perceptrons only recognize linearly separable input
- Hidden layers overcome this problem

M. L. Minsky and S. A. Papert, Perceptrons. Cambridge, MA: MIT Press. 1969.

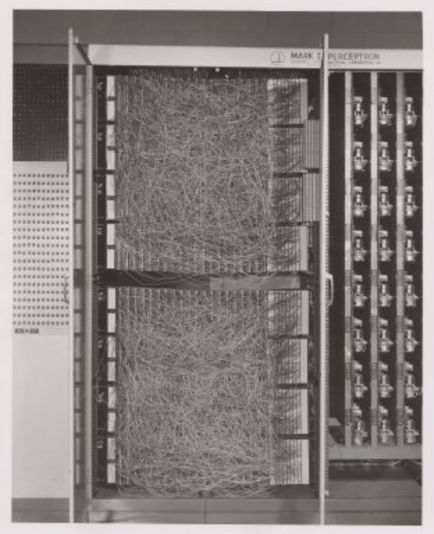
Perceptron History

- ANN's were slow.
- Vanishing gradient problem (Sepp Hochreiter)
- Support vector machines (SVN) were faster

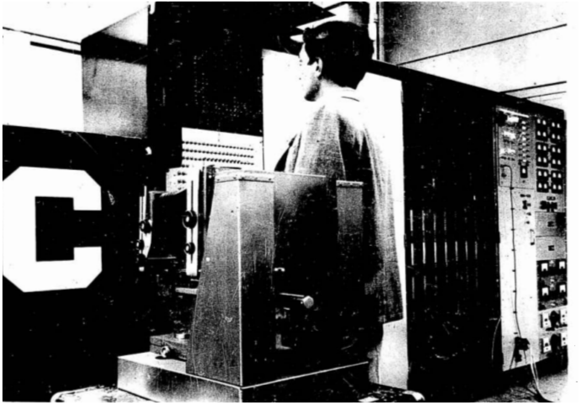
Perceptron History

- Inputs = 400 CdS photocells
- Weights = potentiometers
- Tuning = electric motors

Perceptron History



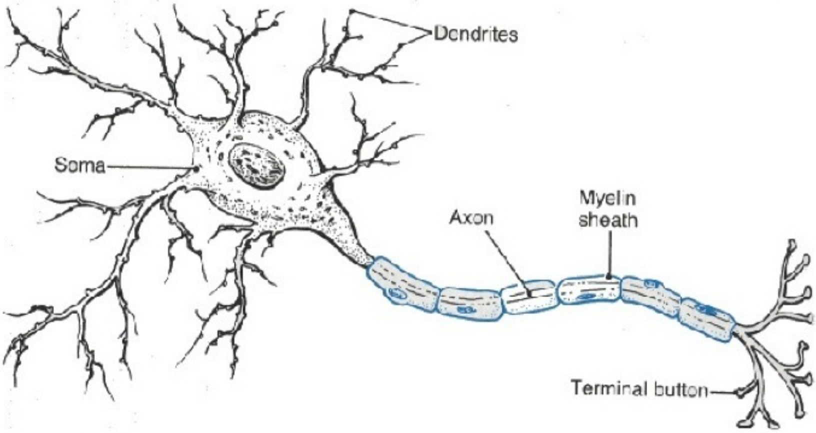
Perceptron History



Frank Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Report No. 1196-G-8, 15 March 1961, Cornell Aeronautical Laboratory

Neurons

Inspired by biology



. . . but only inspired

Linear neuron

$$y = b + \sum_i x_i w_i$$

Linear neuron

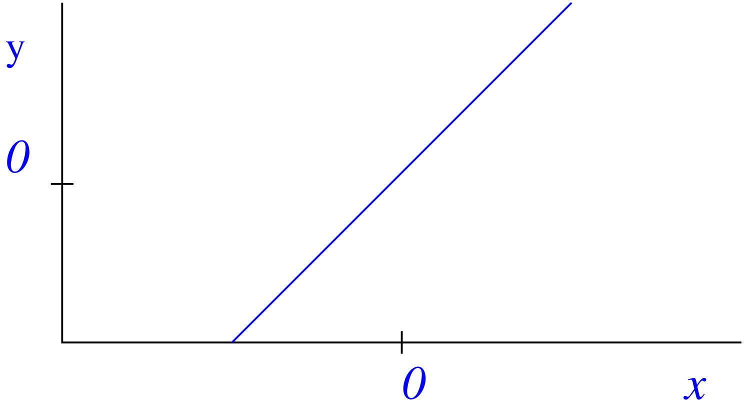
$$y = b + \sum_i x_i w_i$$

where

$$\begin{aligned} y &= \text{output} \\ b &= \text{bias} \\ x_i &= i^{\text{th}} \text{ input} \\ w_i &= \text{weight on } i^{\text{th}} \text{ input} \end{aligned}$$

Linear neuron

$$y = b + \sum_i x_i w_i$$



Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geqslant 0 \\ 0 \text{ otherwise} \end{cases}$$

Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geqslant 0 \\ 0 \text{ otherwise} \end{cases}$$

where

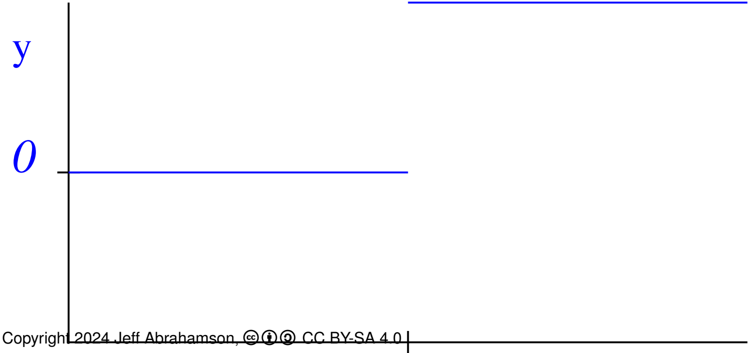
$$\begin{aligned} z &= \text{total input} \\ y &= \text{output} \\ x_i &= i^{\text{th}} \text{ input} \\ w_i &= \text{weight on } i^{\text{th}} \text{ input} \end{aligned}$$

W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115–133, 1943.

Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geqslant 0 \\ 0 \text{ otherwise} \end{cases}$$



Rectified linear neuron

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z \text{ if } z \geqslant 0 \\ 0 \text{ otherwise} \end{cases}$$

Rectified linear neuron

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z \text{ if } z \geqslant 0 \\ 0 \text{ otherwise} \end{cases}$$

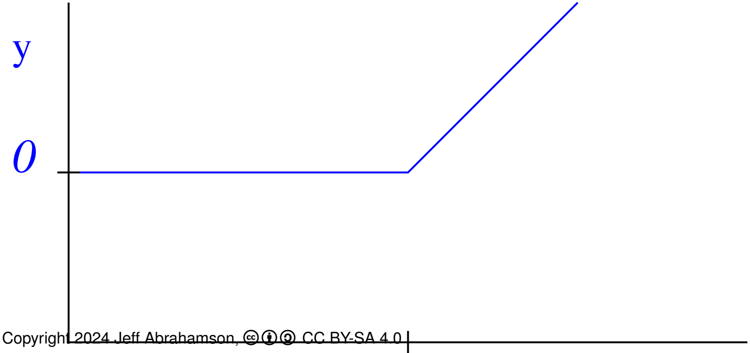
where

$$\begin{aligned} z &= \text{total input} \\ y &= \text{output} \\ b &= \text{bias} \\ x_i &= i^{\text{th}} \text{ input} \\ w_i &= \text{weight on } i^{\text{th}} \text{ input} \end{aligned}$$

Rectified linear neuron

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z \text{ if } z \geqslant 0 \\ 0 \text{ otherwise} \end{cases}$$



Sigmoid neuron

$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$

Sigmoid neuron

$$z = b + \sum_i x_i w_i$$

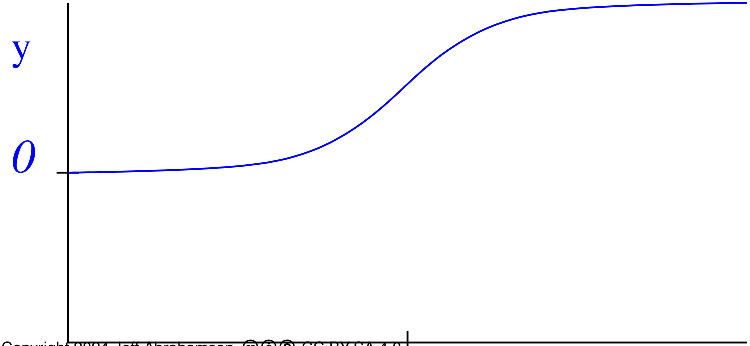
$$y = \frac{1}{1 + e^{-z}}$$

(It's differentiable!)

Sigmoid neuron

$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$



Stochastic binary neuron

$$z = b + \sum_i x_i w_i$$

$$p = \frac{1}{1 + e^{-z}}$$

$$y = \begin{cases} 1 \text{ with probability } p \\ 0 \text{ with probability } 1 - p \end{cases}$$

Stochastic binary neuron

$$z = b + \sum_i x_i w_i$$

$$p = \frac{1}{1 + e^{-z}}$$

$$y = \begin{cases} 1 \text{ with probability } p \\ 0 \text{ with probability } 1 - p \end{cases}$$

(a probability distribution)

Stochastic binary neuron

$$z = b + \sum_i x_i w_i$$

$$p = \frac{1}{1 + e^{-z}}$$

$$y = \begin{cases} 1 \text{ with probability } p \\ 0 \text{ with probability } 1 - p \end{cases}$$

Can also do something similar with rectified linear neurons, produce spikes with probability p with a Poisson distribution.

Neural Networks

Architecture

It's how we connect the dots (the states).

Architecture

Feedforward neural networks

- Flow is unidirectional
- No loops

Makes linear separators (perceptron).

Architecture

Idea: maybe add some layers in the middle

Architecture

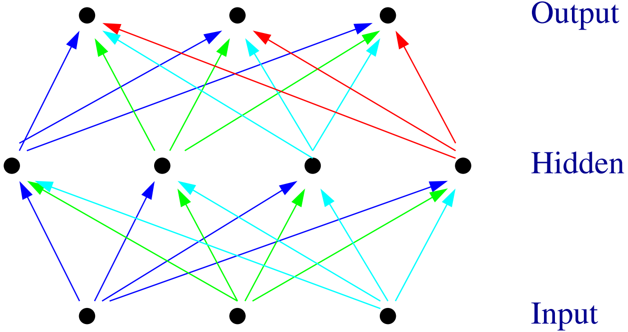
Idea: maybe add some layers in the middle

What would we put there?

Maybe choose not to care, call them “hidden layers”.

Layers

Neuron activity at each layer must be a non-linear function of previous layer



If more than two hidden layers, then we call it deep

Recurrent neural networks (RNN)

- Cycles
- Memory
- Oscilalations
- Harder to train

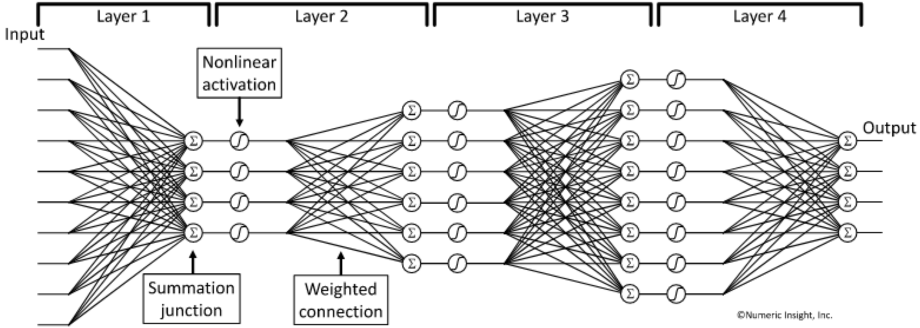
Animated gif time

So how would we train such a thing?

So how would we train such a thing?

It turns out that the perceptron algorithm is unstable and prone to many problems on deep or non-linear networks.

Answer: Backpropagation



$$(10 \times 4) + (4 \times 6) + (6 \times 8) + (8 \times 4) = 144 \text{ connections}$$

Shashi Sathyanarayana, A Gentle Introduction to Backpropagation, 22 July 2014.

Question: How do we find the weights on those 144 connections?

Question: How do we find the weights on those 144 connections?

Need a way of refining an initial (random) guess.

Question: How do we find the weights on those 144 connections?

Need a way of refining an initial (random) guess.

Feedforward is not stable.

Question: How do we find the weights on those 144 connections?

So work backwards.

Backprogation

- Modify weights at output layer by an amount proportional to the partial-derivative of the error with respect to that weight.
- Then do next layer.
- Continue through all layers, recomputing partial derivatives at each step.

Backprogation

- Modify weights at output layer by an amount proportional to the partial-derivative of the error with respect to that weight.
- Then do next layer.
- Continue through all layers, recomputing partial derivatives at each step.

Repeat.

This was hard to learn to do right.

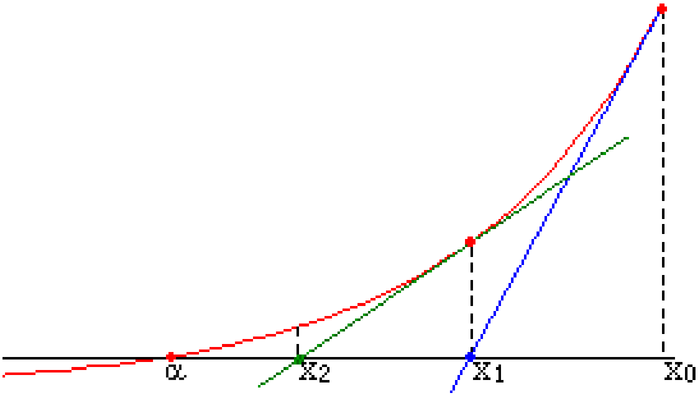
Backpropagation

Backpropagation

Assign small random weights
while *not converged* **do**
    Feed forward to find values at each node
    Backpropagate to correct weights:
    Algorithm `algo()`
        Compute partial derivatives
        Use partial derivatives to compute gradient
        Use gradient to update weights (classic gradient descent)
end

Backpropagation

Newton's Method



Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{f(x_n)}{\frac{\mathrm{d}f}{\mathrm{d}x}(x_n)}$$

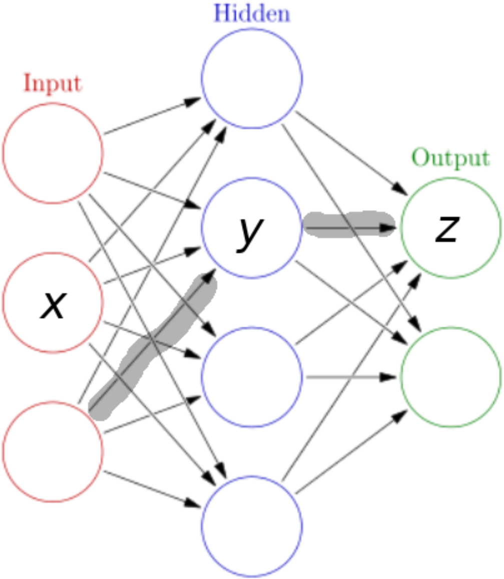
Backpropagation

Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{f(x_n)}{\frac{\mathrm{d}f}{\mathrm{d}x}\left(x_n\right)}$$

or

$$y = \frac{\mathrm{d}f}{\mathrm{d}x}\left(x_n\right)(x - x_n) + f(x_n)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

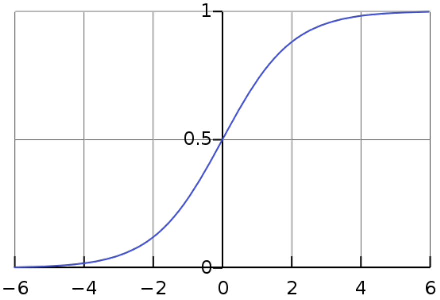
Perceptron

Vocabulary

Activation function

How the neuron decides when to fire.

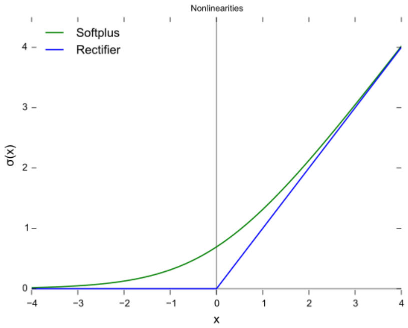
Vocabulary



Sigmoid

By Qef (talk) - Created from scratch with gnuplot, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=4310325

Vocabulary

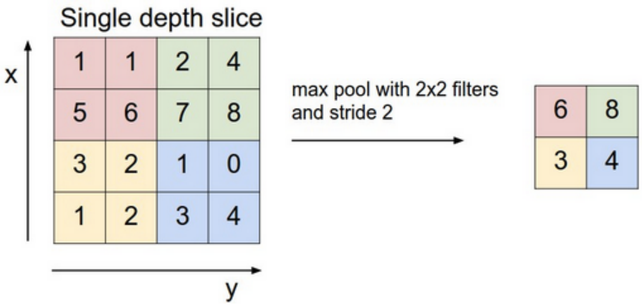


ReLU and Softmax

Rectified Linear Unit

Wikimedia Commons

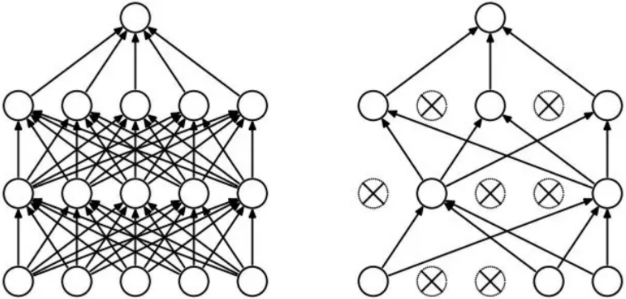
Vocabulary



Pooling

max pooling
downsampling

Vocabulary



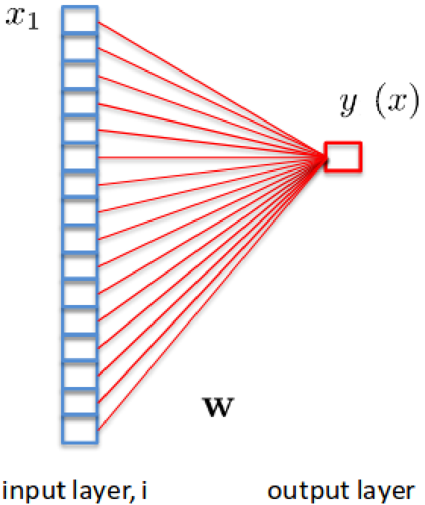
Dropout

Multilayer Perceptron (MLP)

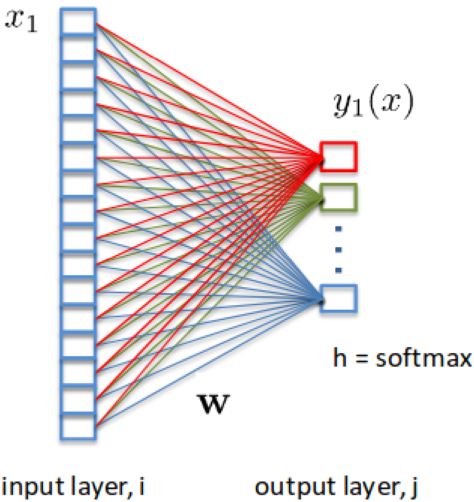
MLP

1. Multiple layers
2. First layer is linear
3. Later layers use non-linear activation functions (typically sigmoid)
4. Feedforward
5. Can find non-linear separators

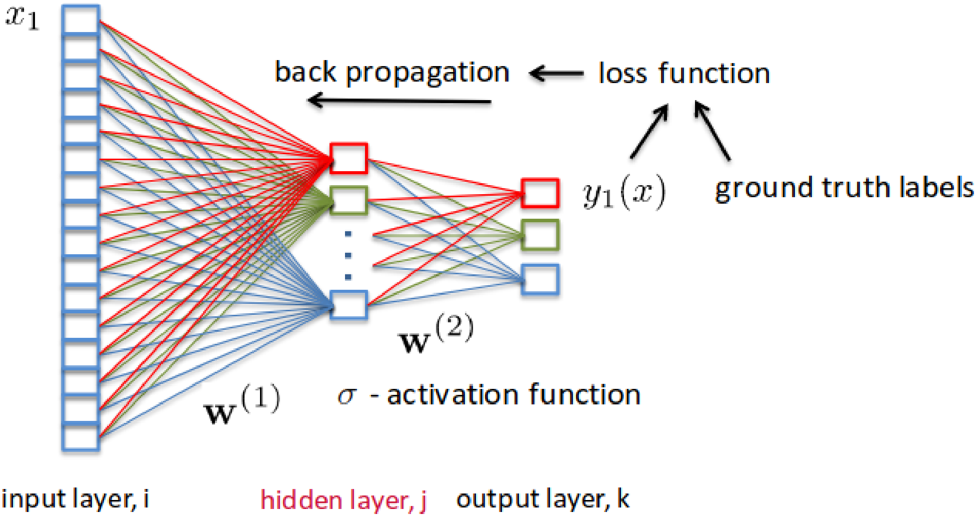
MLP



MLP



MLP


x_1
back propagation
loss function
$y_1(x)$
ground truth labels
$\mathbf{w}^{(2)}$
$\mathbf{w}^{(1)}$
σ - activation function
input layer, i
hidden layer, j
output layer, k

Example: MNIST

Example: simple classification tasks

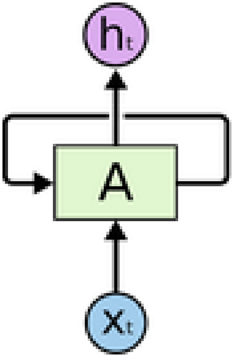
Example: time series

Recurrent Neural Networks

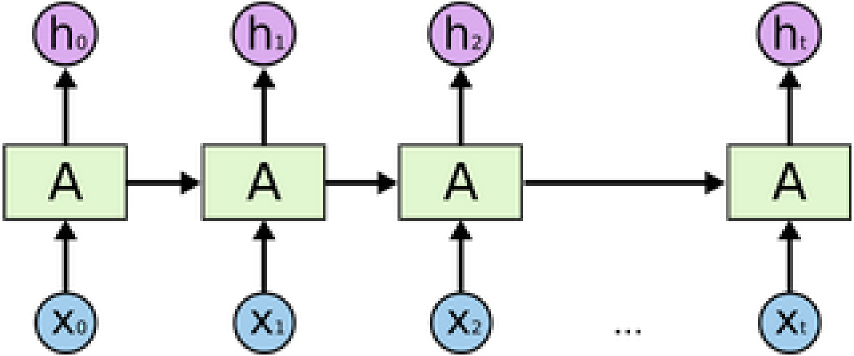
loops

loops

so also memory



https://colah.github.io/posts/2015-08-Understanding-LSTMs/



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Convolutional Neural Networks

ConvNets

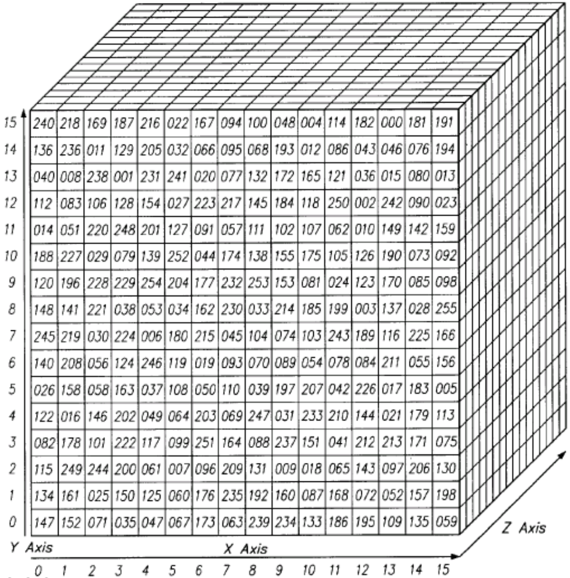
Tensors

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Tensors

$$\begin{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} & \begin{pmatrix} 3 \\ 4 \end{pmatrix} & \begin{pmatrix} 5 \\ 6 \end{pmatrix} \\ \begin{pmatrix} 7 \\ 8 \end{pmatrix} & \begin{pmatrix} 9 \\ 10 \end{pmatrix} & \begin{pmatrix} 11 \\ 12 \end{pmatrix} \end{pmatrix}$$

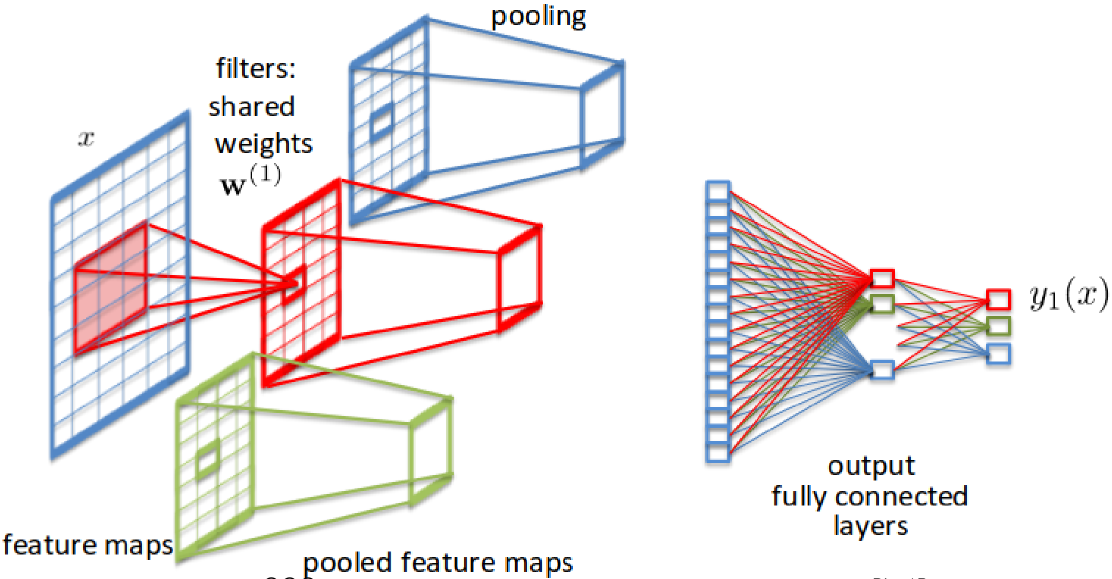
Tensors


15 240 218 169 187 216 022 167 094 100 048 004 114 182 000 181 191
14 136 236 011 129 205 032 066 095 068 193 012 086 043 046 076 194
13 040 008 238 001 231 241 020 077 132 172 165 121 036 015 080 013
12 112 083 106 128 154 027 223 217 145 184 118 250 002 242 090 023
11 014 051 220 248 201 127 091 057 111 102 107 062 010 149 142 159
10 188 227 029 079 139 252 044 174 138 155 175 105 126 190 073 092
9 120 196 228 229 254 204 177 232 253 153 081 024 123 170 085 098
8 148 141 221 038 053 034 162 230 033 214 185 199 003 137 028 255
7 245 219 030 224 006 180 215 045 104 074 103 243 189 116 225 166
6 140 208 056 124 246 119 019 093 070 089 054 078 084 211 055 156
5 026 158 058 163 037 108 050 110 039 197 207 042 226 017 183 005
4 122 016 146 202 049 064 203 069 247 031 233 210 144 021 179 113
3 082 178 101 222 117 099 251 164 088 237 151 041 212 213 171 075
2 115 249 244 200 061 007 096 209 131 009 018 065 143 097 206 130
1 134 161 025 150 125 060 176 235 192 160 087 168 072 052 157 198
0 147 152 071 035 047 067 173 063 239 234 133 186 195 109 135 059
Y Axis
X Axis
Z Axis
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

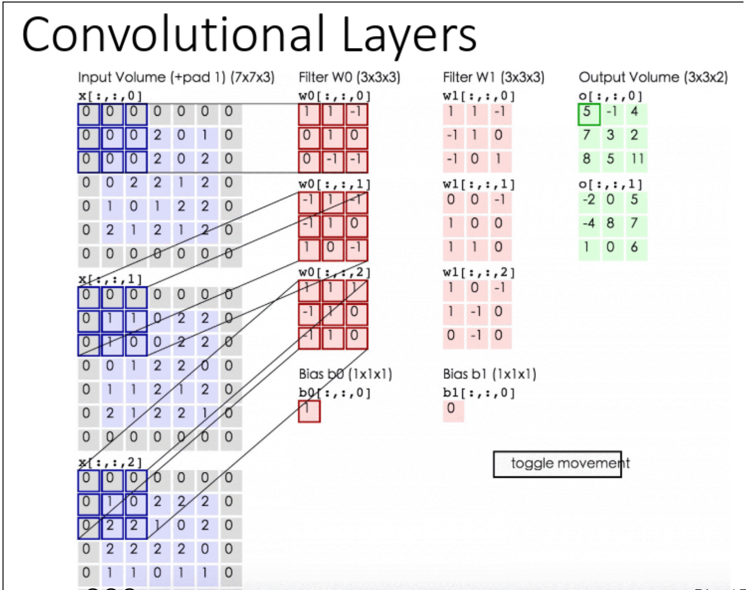
Convolutional Neural Networks

The neurons are convolutions

ConvNets

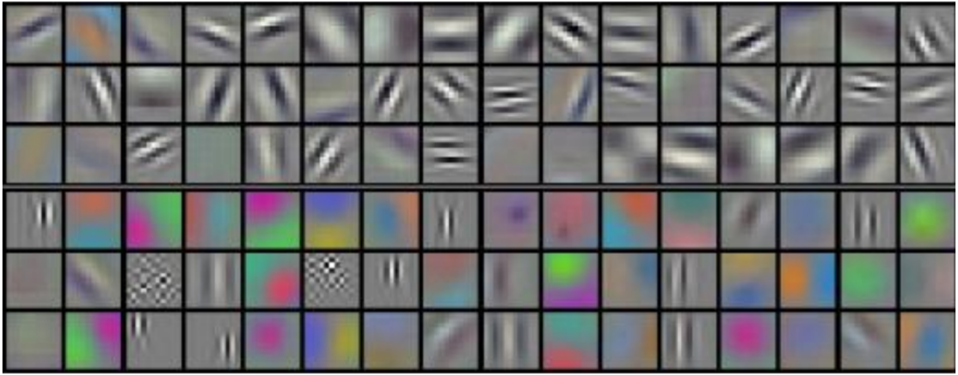

pooling
filters:
shared
weights
$\mathbf{w}^{(1)}$
x
$y_1(x)$
output
fully connected
layers
feature maps
pooled feature maps

ConvNets



Diva / Beapp

ConvNets



Example: images

Transfer Learning

- Pre-trained models
- Only retrain the classifier at the end

Example: PlacesVGG (2015)

PlacesVGG (Places2 2015)

`https://static.turi.com/models/.../places_vgg_16-1.0.tar.gz`

This model is trained with VGG-16 architechture, on the Places2 dataset. The Places2 dataset contains 8 million images of 400 different scene categories. More details about the dataset can be found at http://places2.csail.mit.edu/

https://turi.com/products/create/docs/graphlab.mxnet.pretrained_image_model.html

Enough for Today