COPENHAGEN BUSINESS ACADEMY





Multivariate and logistic regression

Jens Egholm Pedersen
<jeep@cphbusiness.dk>

Recap

- Populations
  - Samples
  - How do we measure the difference?
- Distributions
  - Probability that the sample is distributed like the population
- Regression functions
  - Describes how the mean of the response variable changes
- Linear regression
  - Predict values based on the formula: $y = ax + b$
- Testing/training data

Goal of this block

- Have a basic understanding and knowledge of various terms, models and tests in statistics.
- Compute basic statistics on data using the Python's scientific stack and the Sklearn library.
- Develop an informed guess of when to choose a certain model to answer a concrete type of question and apply technology appropriately.

See also: BI plan

Goal for today

- Hand-in debriefing
- Dimensionality
- Multivariate linear regression
- Cross-validation
- Logistic regression
- Polynomial regression

See also: BI plan

Types of machine learning

- Is it trained by a human
  - Supervised / unsupervised
- Can they learn on the fly?
  - Online learning / offline (batch) learning
- Do they include new data?
  - Instance-based / model-based
- Today: supervised, offline model-learning

See also: Géron: Hands-on machine learning (book)

Linear regression

- Training a model
  - y = ax + b
  - For a given x, what is y?
  - How long does it **on average** take to get 1000 points?
- What kind of machine learning?
  - Supervised, offline, model-based

See also: BI plan

Linear regression

- What if there are multiple factors?
- How can we expand our model?
  - What is “one factor” in the model?
- $y = ax_1 + bx_2 + c$

See also: BI plan

Multivariate regression

- Multivariate regression
  - No longer simply one input value
- $y = ax_1 + bx_2 + c$

- What about our error models?
  - MAE?
  - MSE?
  - Pearson’s r?

See also: Python for multivariate analysis

Multivariate regression

- Multivariate regression
  - No longer simply one input value

- Linear regression in sklearn
  - model = LinearRegression()
  - model.fit(.. ? ..)
  - model.predict (.. ? ..)

Hands-on: Wine

- How do you get good wine?

- `https://github.com/datsoftlyngby/soft2017fall-business-intelligence-teaching-material`

Array shapes

- What do you call an array with one dimension?
- What do you call an array with two dimensions?
- What do you call an array with three or more dimensions?

```
numpy.shape

np.random.sample(10)

np.random.sample(10).shape // (10, )

np.random.sample((10, 10)).shape // (10, 10)
```

Array zipping

- numpy.stack
  - Join a sequence of arrays along a new axis.

- Parameters:
  - arrays : sequence of array_like
  - axis : int, optional
    - The axis in the result array along which the input arrays are stacked.

- In a linear regression, how many input dimensions do we have?
  - How can we 'stack' two 1-d arrays into one 2-d array?

- `numpy.stack((x, y), axis = 1)`

Multivariate regression

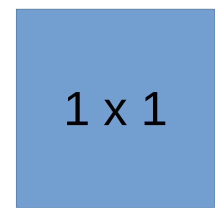
- Multivariate regression
  - Multiple input variables
- What other problems could this help you solve?
- How do you verify that your model improves your ability to predict?

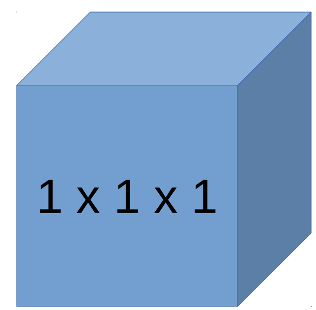
The curse of dimensionality

- You have a unit square

- Average distance between two random points

- In 2-d: .52
- In 3-d: .66
- In 1’000’000-d: 408.25





The curse of dimensionality

- Multivariate regression
  - Input values in multiple dimensions
  - I.e. We have to work with higher dimensionality

- My advice: don’t try to visualise it
  - I know it’s intuitive, but it won’t help you

- Either:
  - Think about causality: how do they “bind”?
  - Reduce the dimensionality (sometimes necessary)

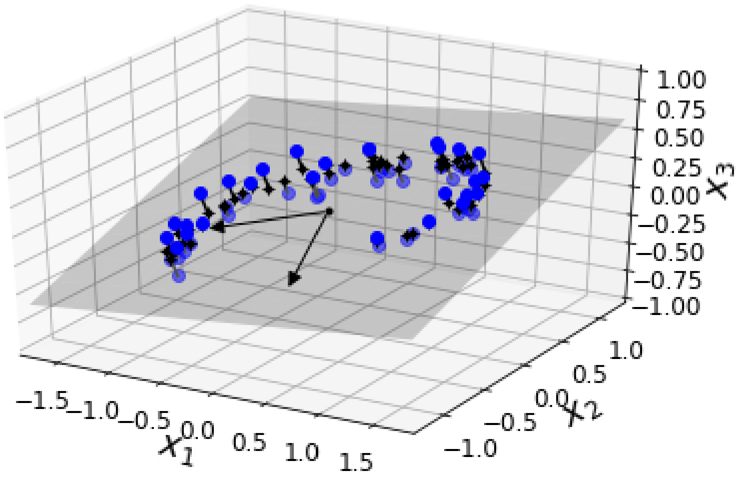
Dimensionality reduction

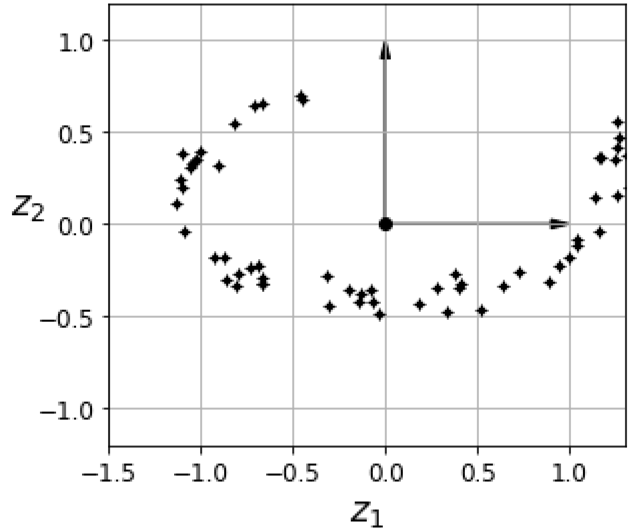
- Reducing dimensionality == compression
  - You *will* loose data

- Most common approaches:
  - Manifold (we won’t touch this)
  - Principal component analysis

PCA

- How can you simplify this?

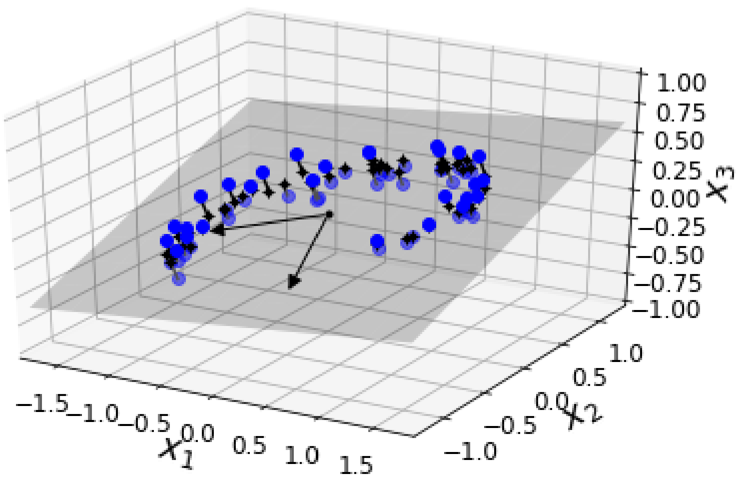


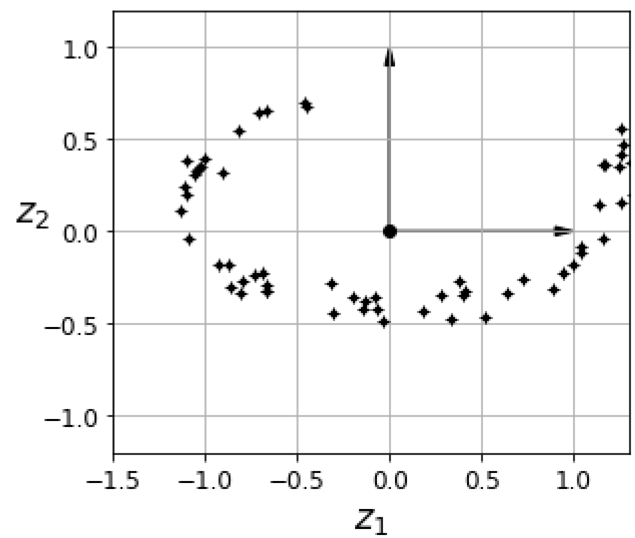


See also: Géron: Dimensionality reduction examples

PCA

- PCA is
  1) Finding a hyperplane
  2) Project data onto that hyperplane

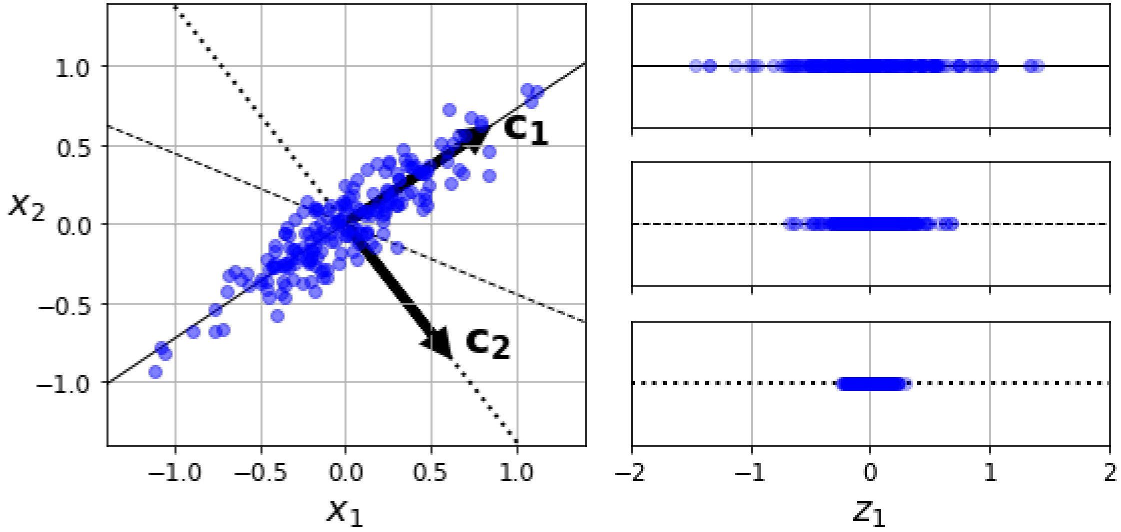




See also: Géron: Dimensionality reduction examples

PCA

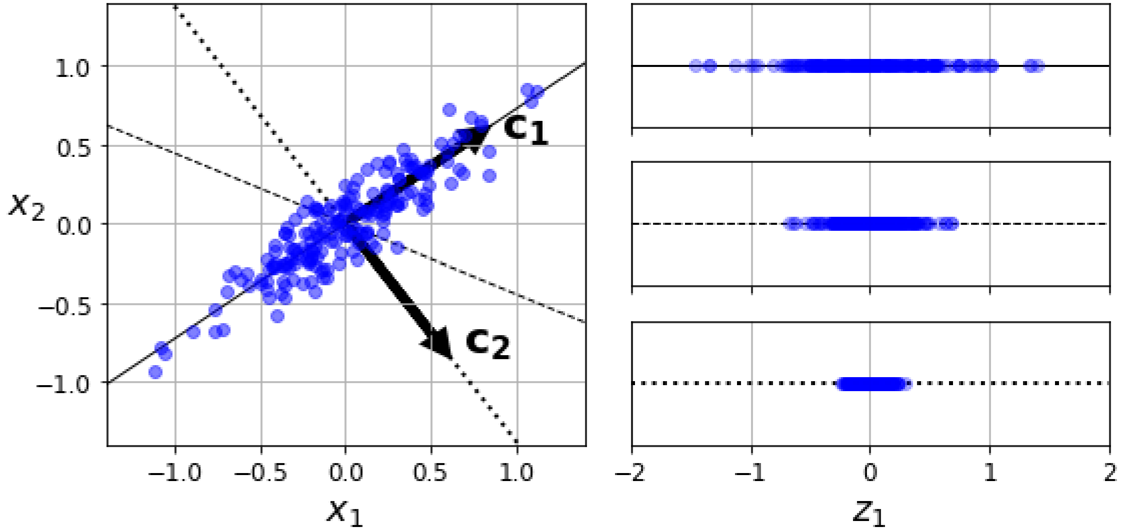
- PCA is
    1) Finding a hyperplane
    2) Project data onto that hyperplane



See also: Géron: Dimensionality reduction examples

PCA

- PCA is
  1) Finding a hyperplane
  2) Project data onto that hyperplane



See also: Géron: Dimensionality reduction examples

PCA in sklearn

- PCA is
  1) Finding a hyperplane

     model = PCA()

     model.fit(X)

  2) Project data onto that hyperplane

     model.transform(X)

- Not so fast…. What could be a problem here?
  - PCA finds the *most significant* dimensions (principal components)
  - What if one dimension is far more "impactful" than another?

Data standardisation

- PCA assumes that all the data has the same variance
- Problem: data rarely has the same variance
- Solution: scale it to the same mean and std.dev
- `sklearn.preprocessing.scale()`

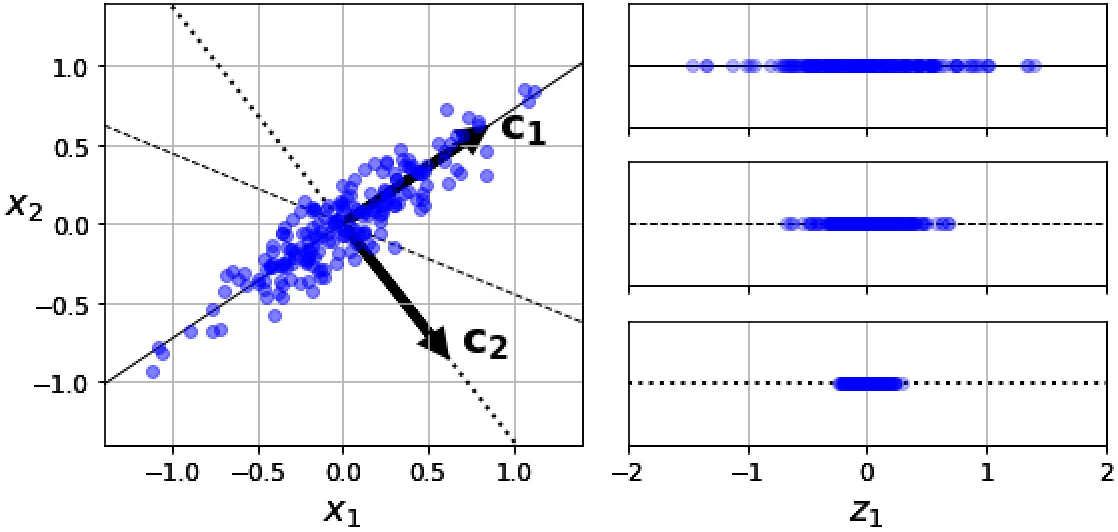
Hands-on: Dimensionality

- How do you handle high dimensionality?

- `https://github.com/datsoftlyngby/soft2017fall-business-intelligence-teaching-material`

PCA explained

- PCA is
  1) Finding a hyperplane
  2) Project data onto that hyperplane



See also: Géron: Dimensionality reduction examples

PCA explained

- PCA is
  1) Finding a hyperplane
  2) Project data onto that hyperplane

- PCA gives you a matrix (W) that can project data

$$x_{projected} = x \cdot w_d$$

See also: Géron: Dimensionality reduction examples

PCA explained

- PCA is
  1) Finding a hyperplane
  2) Project data onto that hyperplane

- PCA gives you a matrix (W) that can project data

- `sklearn.components_`
- `sklearn.explained_variance_ratio_`

See also: Géron: Dimensionality reduction examples

Under- and overfitting

- Polynomial example
- Why is it bad?
  - How can you measure that?
  - How can you avoid that?
- One solution: “hide” data from the model

See also: Sklearn cross-validation

Cross-validation

- Training data versus testing data
  - Normally 80/20 split
- Model training → model prediction
- Why not construct many models and switch between training/testing data?

See also: Sklearn cross-validation

Cross-validation

1) Split data into *n* “folds”
2) Choose one fold for testing and the rest for training
3) Repeat *n* times

- Called: k-fold cross validation

- Solves overfitting, because data is “hidden”

See also: Sklearn cross-validation

K-fold cross-validation

- K-fold cross-valiation
  - Normally 10 folds

- Still need one thing: How does the models compare?

from sklearn.model_selection import cross_val_score

See also: Example: Cross validation pipeline

So far: numerical predictions

- For wine quality of x, what is y?
- For income rate of x, what is y?

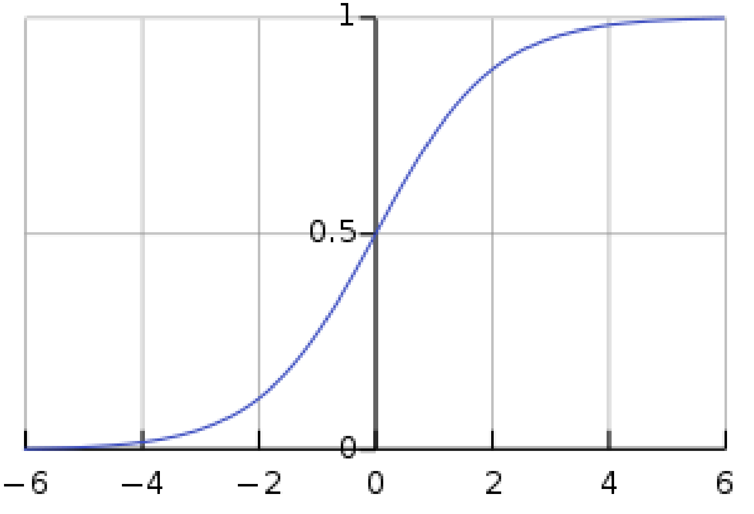
- What if we don't want to predict a number?
  - What is the gender (male/female/other)
  - Medical: what causes death?

See also: Example: Cross validation pipeline

Logistic regression

- Linear regression model:
  - y = ax + b
- Logistic regression model:

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

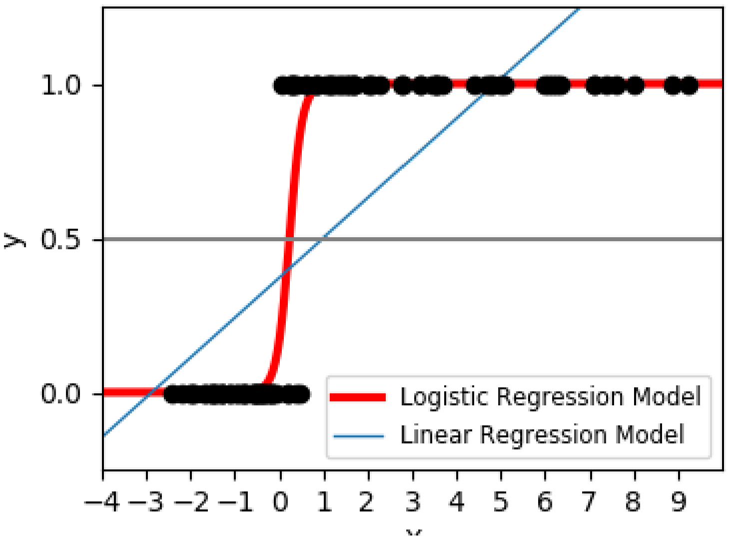


See also: Example: Cross validation pipeline

Logistic regression

- Linear regression model:
  - y = ax + b
- Logistic regression model:

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



See also: Example: logistic regression

Logistic regression errors

- We are no longer measuring distance to “actual”
  - Instead: right or wrongly classified?
- Do we have the same error metrics as before?
  - Nope
- For now use accuracy
  - `sklearn.metrics.accuracy_score`
- We’ll talk more about that next week

Logistic regression in sklearn

- Same as with linear regression
  - Supervised, offline, model learning

1) Train the model
2) Predict

- Only difference: categorical y value

```
from sklearn.linear_model import LogisticRegression
```

Next hand-in: Assignment 6

- Deadline: **20th of November 23:59:59**

- Multivariate linear regression:
  - Same hackernews data set
  - Modified to contain number of posts submitted

- Diagnosis of breast cancer
  - Logistic regression (benign/malign)

Next hand-in: Assignment 6

- Deadline: **20th of November 23:59:59**
- The hand-in (on Moodle) should be a link to a GitHub release containing a single file with the code and written text for the assignment parts

- This can either be a .ipynb, .py, .pdf or .md file
- The file must be clearly identifiable. Please name it accordingly. (for instance report.pdf)