Autoencoders and GANs (Ch.17)

- Autoencoders: the key idea

- Variants:
  - AE and PCA
  - Denoising
  - Sparse
  - Variational (VAE)
  - Convolutional, recurrent, …

Study (and play) with the tutorial:

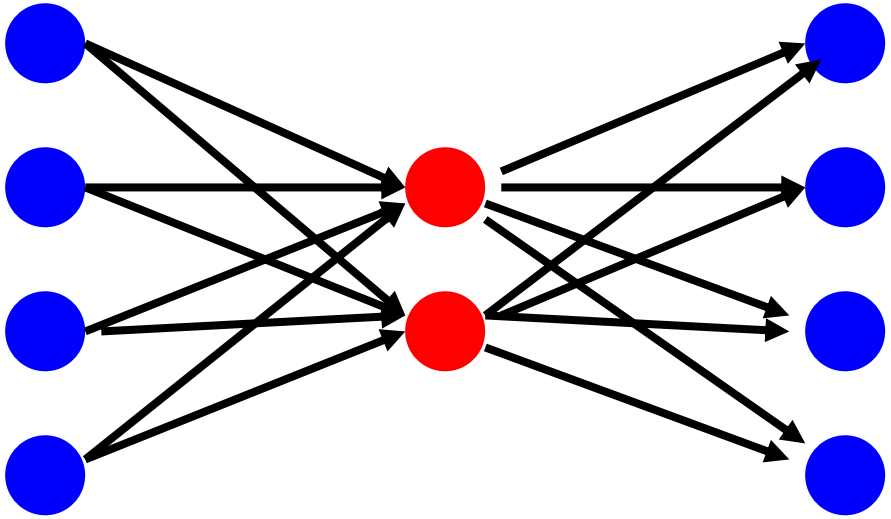
https://blog.keras.io/building-autoencoders-in-keras.html

Generative Adversarial Networks:

http://slazebni.cs.illinois.edu/spring17/lec11_gan.pdf

The simplest autoencoder

- Consider a 4:2:4 network that computes identity function



- Inputs: (0 0 0 1), (0 0 1 0), (0 1 0 0), (1 0 0 0)
- Outputs: (0 0 0 1), (0 0 1 0), (0 1 0 0), (1 0 0 0)

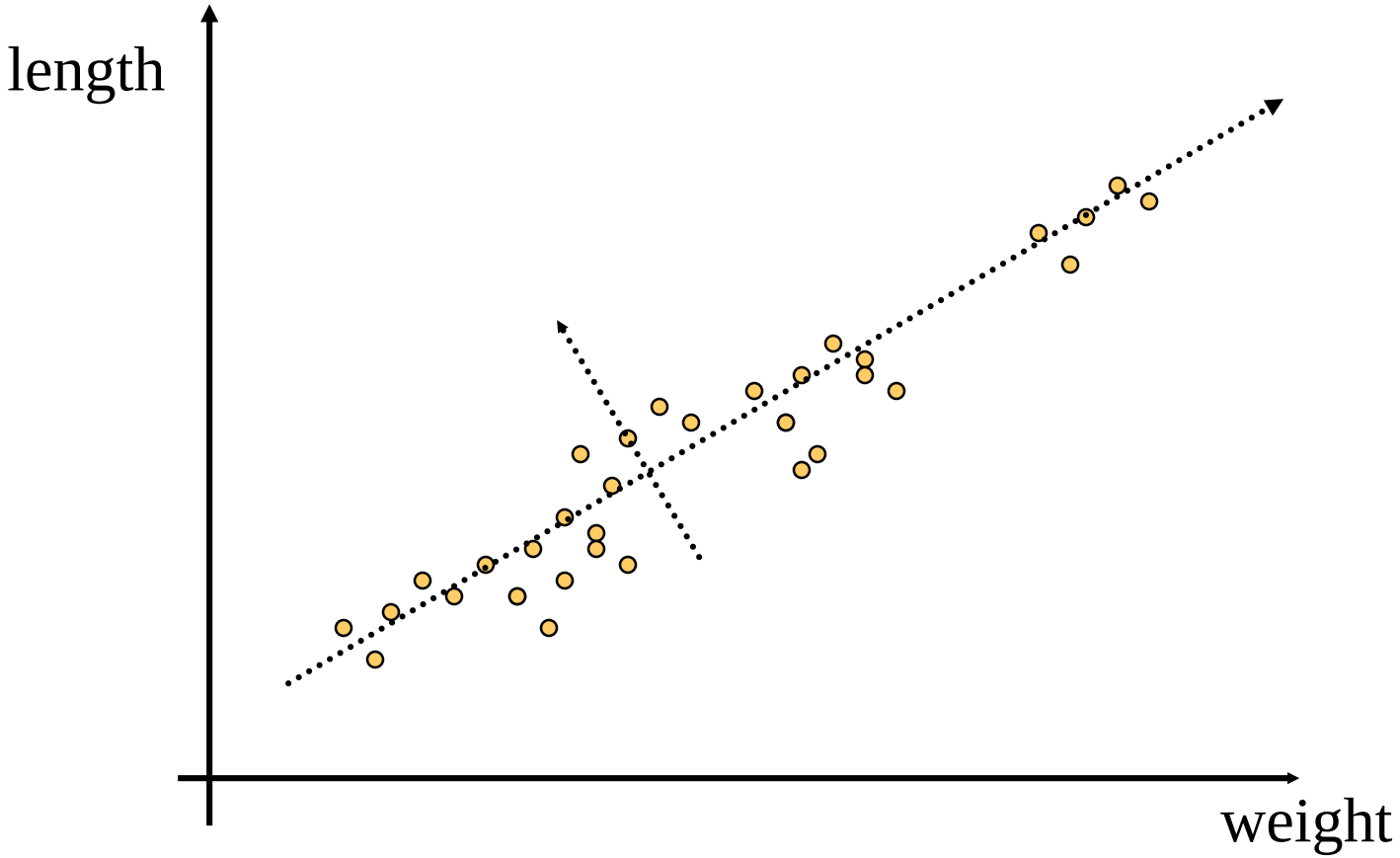
Autoencoders

- The network "learns" the concept of binary representation of integers!
- General case: 2^n:n:2^n networks
- Somehow the network is able to "extract"
  the most concise representation of the input!
- The same trick works for "real numbers": PCA networks
- The same for “bigger networks” with “coder” and “decoder” parts being a convolutional or recurrent networks!
- It can be used for data compression, dimensionality reduction, visualization, anomaly detection, data generator(!)

PCA and Autoencoders

- m inputs, n (n<<m) hidden nodes, m outputs
- Activation function: f(x)=x (identity function)
- the network reduces the dimensionality of data (from m to n) while preserving as much information as possible (MSE minimized)
- Equivalent to Principal Component Analysis
- What happens when activations are non-linear? More hidden layers?

Principal Components Analysis



Principal Components

Given: n variables $x_1, x_2, ..., x_n$

PCs are linear combinations of $x_1, ..., x_n$ such that:

1) they are orthogonal to each other

2) they maximize the “variance of projections”

3) the first PC explains most of the variance, second PC less, ...

In practice the first few PCs (2-3) explain most of the variance

Remark:
PCs (may) have nothing to do with classification !!!

Finding Principal Components

Given: n variables $x_1, x_2, ..., x_n$

1) calculate the matrix of covariances (or correlations)

$$C = (cov(x_i, x_j)) \ , \ i, j = 1, ..., n$$

2) Calculate the eigen structure of the covariance matrix
[V, D]=eig(C) , such that D is a diagonal matrix of eigen values and V is a matrix of eigen vectors.

3) Choose k eigen vectors with the biggest eigen values to form k principal components:

x -> x*[v_1, v_2, ..., v_k]' (n dimensions -> k dimensions)

Example application of PCA's: flower quality estimation

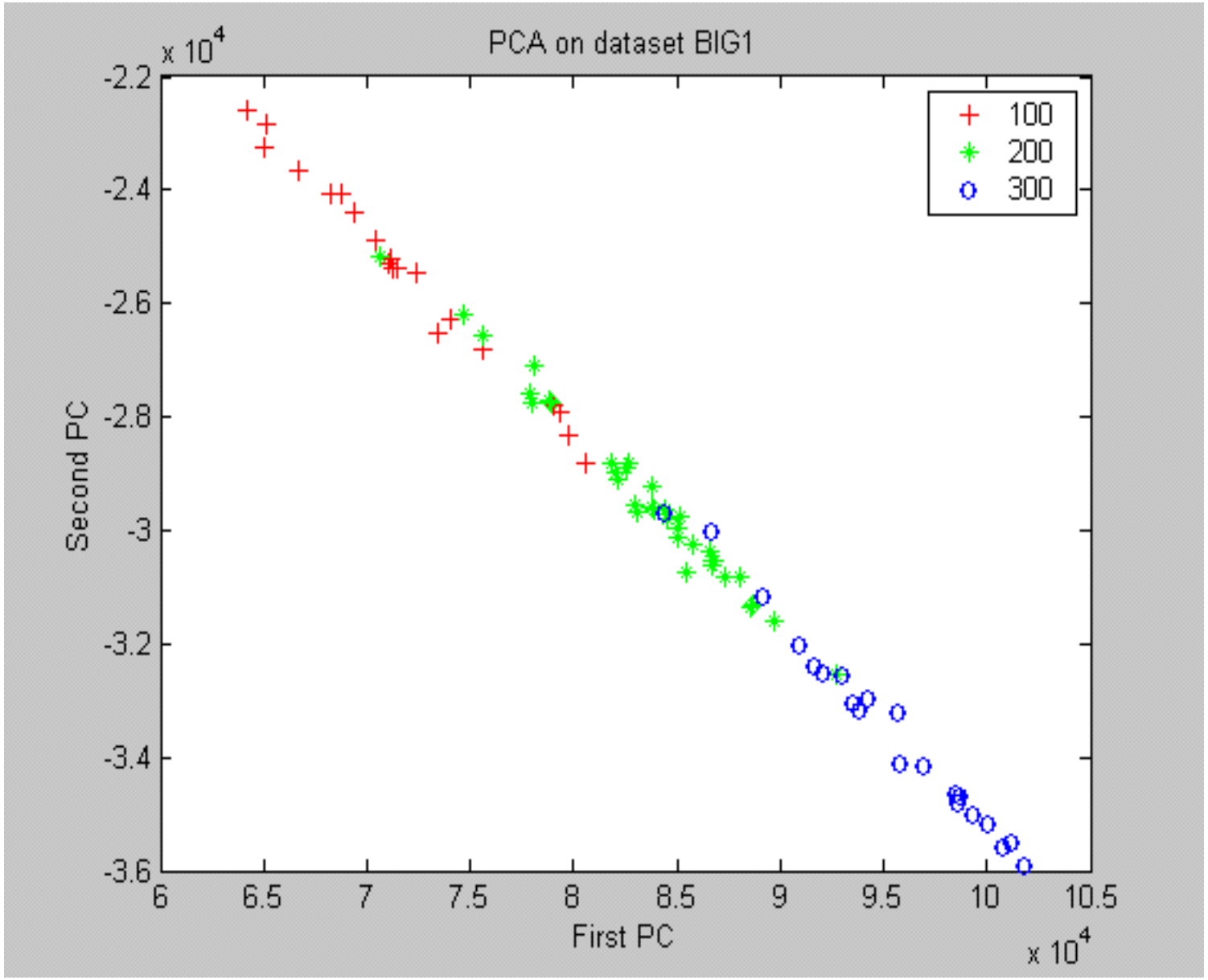
- a digital camera takes several photos of a flower

- some features are extracted from images (sizes, proportions, etc): numbers

- some flowers are labeled as "nice" others as "ugly"

Problem:

develop a system that would automatically estimate flower quality from 30 features


PCA on dataset BIG1
x 10^4
-2.2
-2.4
-2.6
-2.8
-3
-3.2
-3.4
-3.6
Second PC
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
First PC
x 10^4
100
200
300

Denoising AutoEncoders

- Key idea:
- Given a set of “clean” images add noise to them and train an autoencoder on:
  - Input: a noisy image
  - Output: a clear image
- In this way the network “learns” what is essential in an image and what is not!

Sparse AutoEncoders

The key idea:

- The bottleneck layer computes some “essential features” that are needed for input reconstruction.

- Usually, only a few features are really important; the rest represent noise.

- => impose some constraints on the bottleneck layer: e.g., “on average, activations are close to 0”.

- => Kullback-Leibler divergence: measures the distance between two probability distributions; used as a part of the loss function.

Variational AutoEncoders

The key idea:

- The bottleneck layer represents a *latent space*:
  vectors of features that are needed for reconstruction

- "to reconstruct an output we can slightly vary the values
  of the these essential features"

- Assume that the latent features are "normally distributed"

=> Extend the bottleneck layer to include "means" and "std's".

Mastering Autoencoders

Study the tutorial:

blog.keras.io/building-autoencoders-in-keras.html

- - get the code working (TF1.0 -> TF2.x)
- - run various experiments, playing with different parameters
- - play with alternative data sets (eg. FashionMNIST)

References on GANs

- Short explanation:

  https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

- A longer one:

  http://slazebni.cs.illinois.edu/spring17/lec11_gan.pdf

- Several lectures on GANs:

  https://sites.google.com/view/cvpr2018tutorialongans/

- https://thispersondoesnotexist.com/