

# CS 360: Machine Learning

Prof. Sara Mathieson

Fall 2020



**HVERFORD**  
COLLEGE

# Admin

- Office hours **today 4:30-6pm**
- **Study guide** posted for the exam
  - Exam posted mid next week, due Dec 18
- **Lab Thursday**: discuss projects with each group
  - Lab attendance is optional if you're taking the exam
- **Project**
  - Email proposal by Thursday Dec 3
  - Presentation: last day of classes or Dec 18 (morning)
  - Git repo should be finalized by Dec 18 at noon

# CS360 Final Project

Proposal Email: Thursday, December 3 at 11:59pm

Short Presentation: Friday, December 11 (in class) ←

Code + Lab Notebook: Friday, December 18 at noon

There will be an alternative presentation time on Dec 18 (morning)

## Outline for a typical project:

- Find a dataset (see project writeup)
- Run an algorithm we've discussed on the dataset
- Try to do a comparison
  - run the algorithm in multiple ways
  - different data pre-processing
  - try a different algorithm
- Record and interpret the results

# Project Deliverables

- Main deliverable: presentation

- Group of 1: 5 min
- Group of 2: 9 min
- Group of 3: 12 min

- On git:
  - Lab Notebook
  - Project Code

# Outline for December 1

- Training a NN: backpropagation
- Begin unsupervised learning
- K-means clustering
- Next time
  - Gaussian Mixture Models (GMM)
  - Principal Component Analysis (PCA)

# Outline for December 1

- Training a NN: backpropagation
- Begin unsupervised learning
- K-means clustering
- Next time
  - Gaussian Mixture Models (GMM)
  - Principal Component Analysis (PCA)

# Backpropagation

- *High-level goal:* we want to know how the output depends on the input
- *Issue:* network is very complicated and overall gradient may be difficult to compute
- *Idea:* use the chain rule to compute local gradients throughout the network
- *Takeaway:* nodes can know about their value and local gradient without knowing about the network they are imbedded in

# $J(\vec{w})$ Backpropagation

loss/cost

weights  
 $\vec{w}$

Idea: want to know how output depends on input

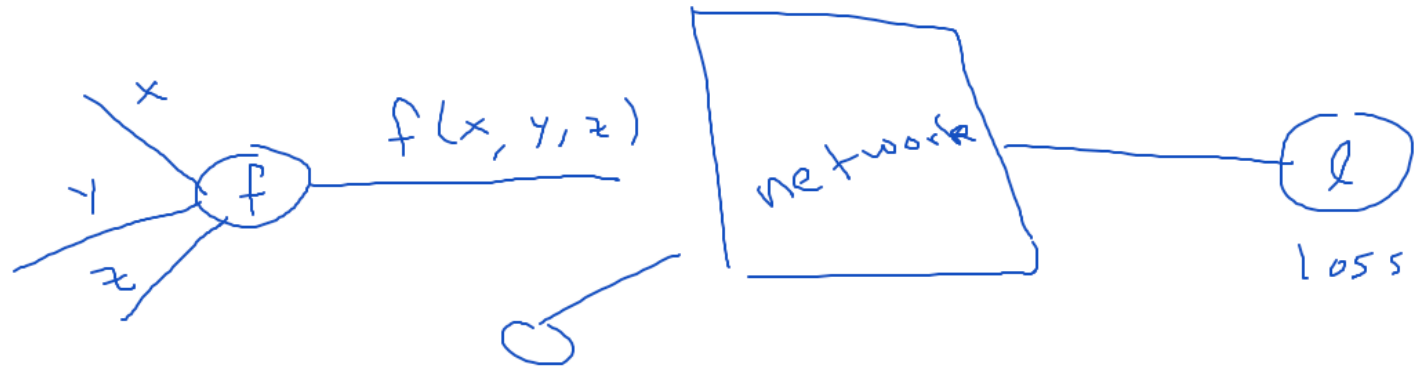
example

$$f(x, y, z) = (x + y)z = \underbrace{g}_{\text{weights}} \cdot z$$

$$\frac{\partial f}{\partial x} = \underbrace{\frac{\partial f}{\partial g}}_z \cdot \underbrace{\frac{\partial g}{\partial x}}_1 = z$$

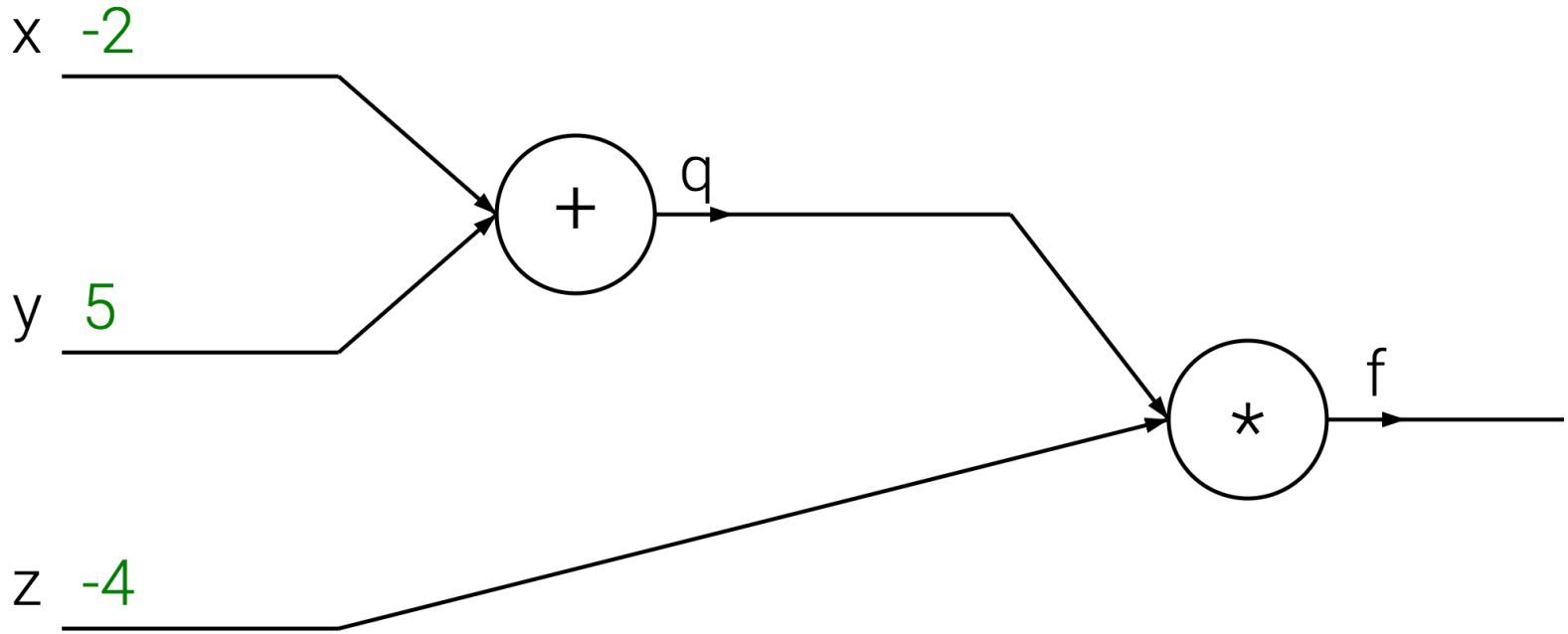
SGD

$$\begin{aligned} x &\leftarrow x - \alpha \frac{\partial f}{\partial x} \\ y &\leftarrow y - \alpha \frac{\partial f}{\partial y} \\ z &\leftarrow z - \alpha \frac{\partial f}{\partial z} \end{aligned}$$





# Backpropagation: Example 1



Example from:

<http://cs231n.github.io/optimization-2/>

# Backpropagation: Example 1

compute function value

$$f(x, y, z) = (x+y)z$$

$$(-2+5)(-4)$$

- ① forward pass
- ② backprop the derivative

initial weights

$$\alpha = 0.1$$

$$x = -2$$

$$x \leftarrow -1.6$$

$$y = 5$$

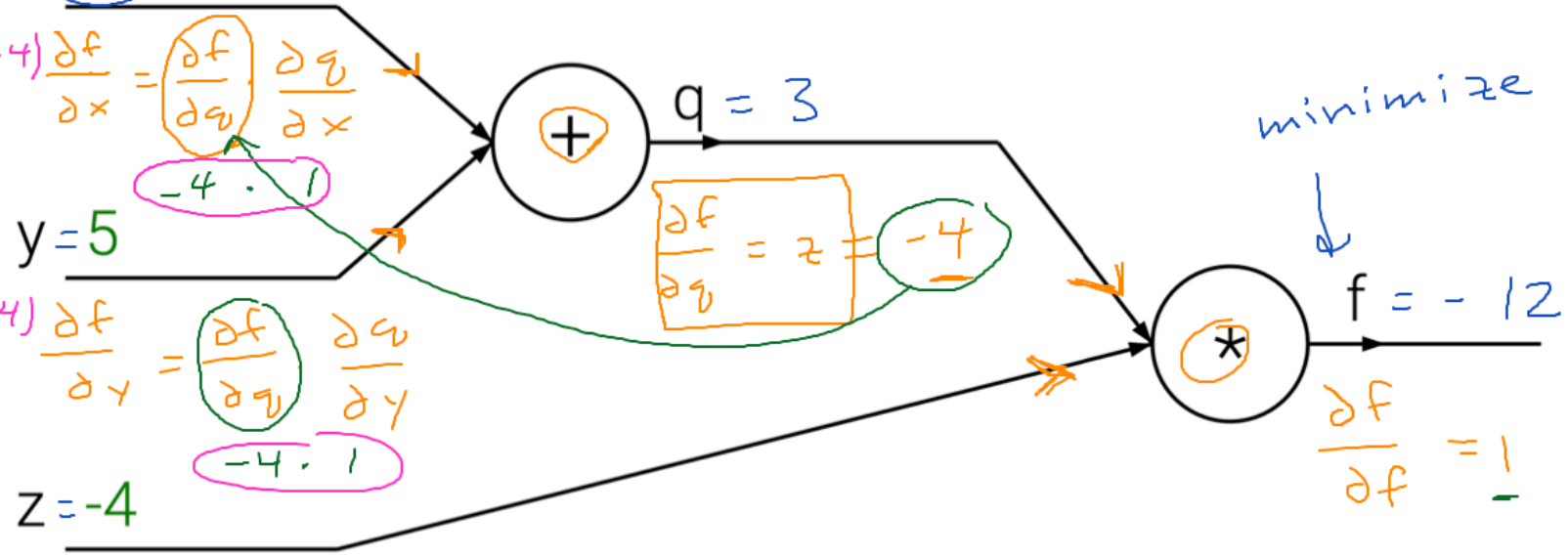
$$y \leftarrow 5.4$$

$$z = -4$$

$$z \leftarrow -4.3$$

$$z \leftarrow -4.3$$

$$f(-1.6, 5.4, -4.3) = -16.34$$



$$\frac{\partial f}{\partial z} = q = 3$$

lower!

$$f = q \cdot z$$

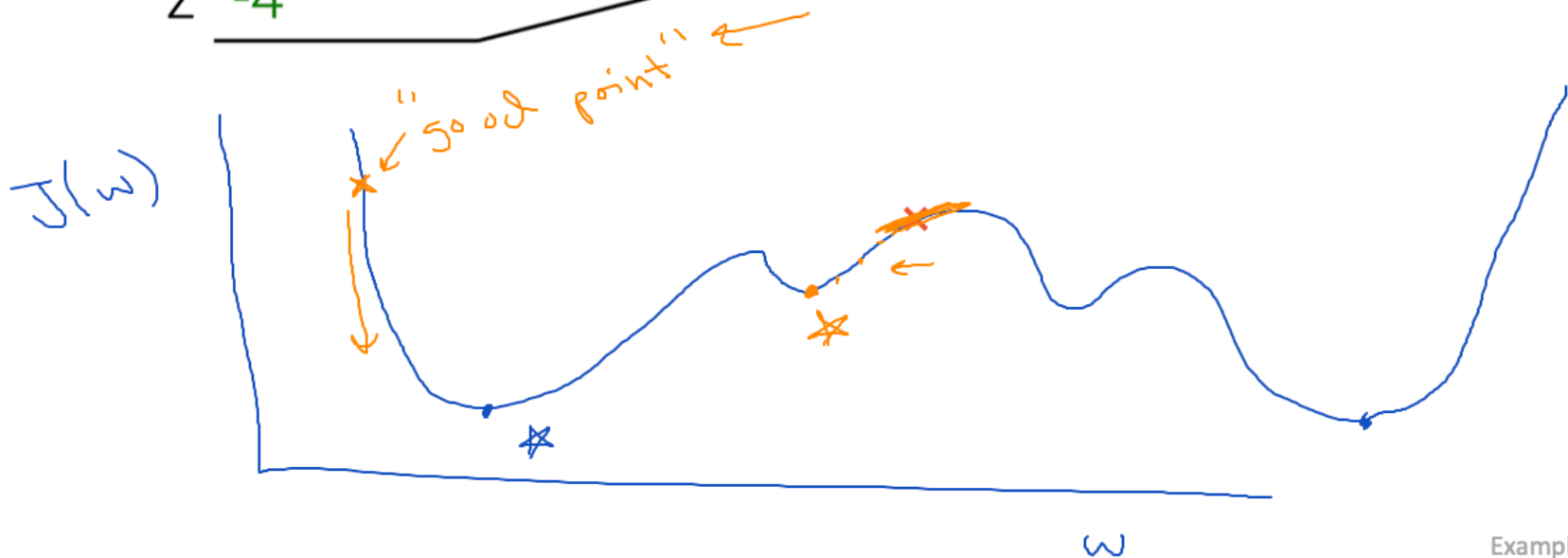
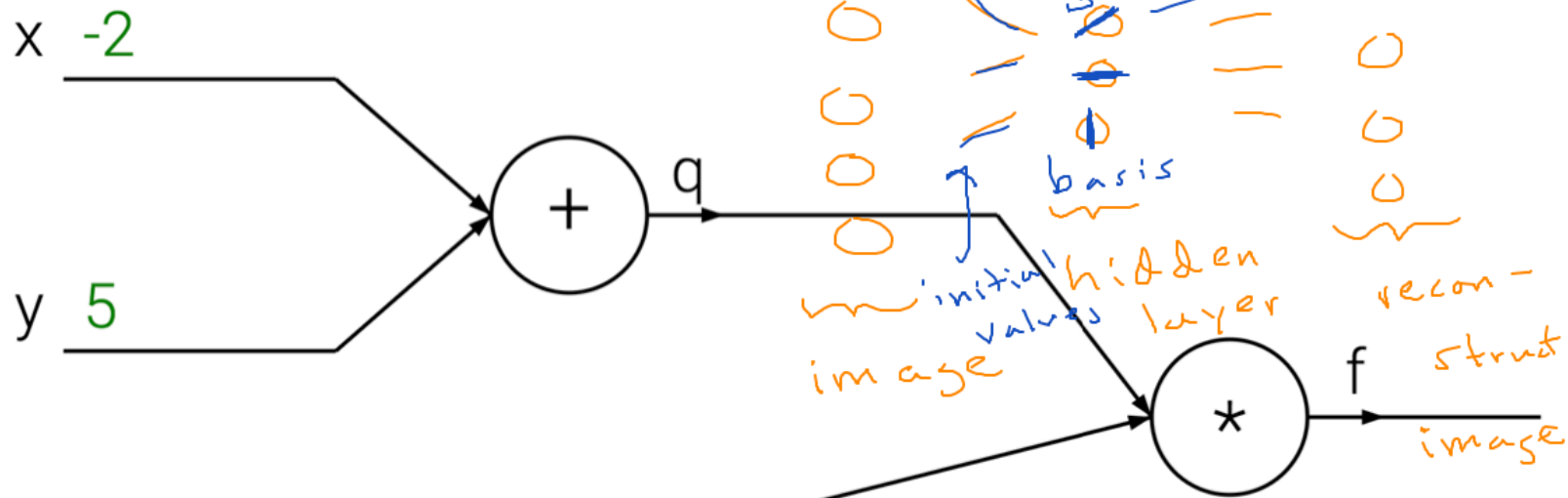
$$\frac{\partial f}{\partial q} = z$$

$$\frac{\partial f}{\partial z} = q$$

Example from:

<http://cs231n.github.io/optimization-2/>

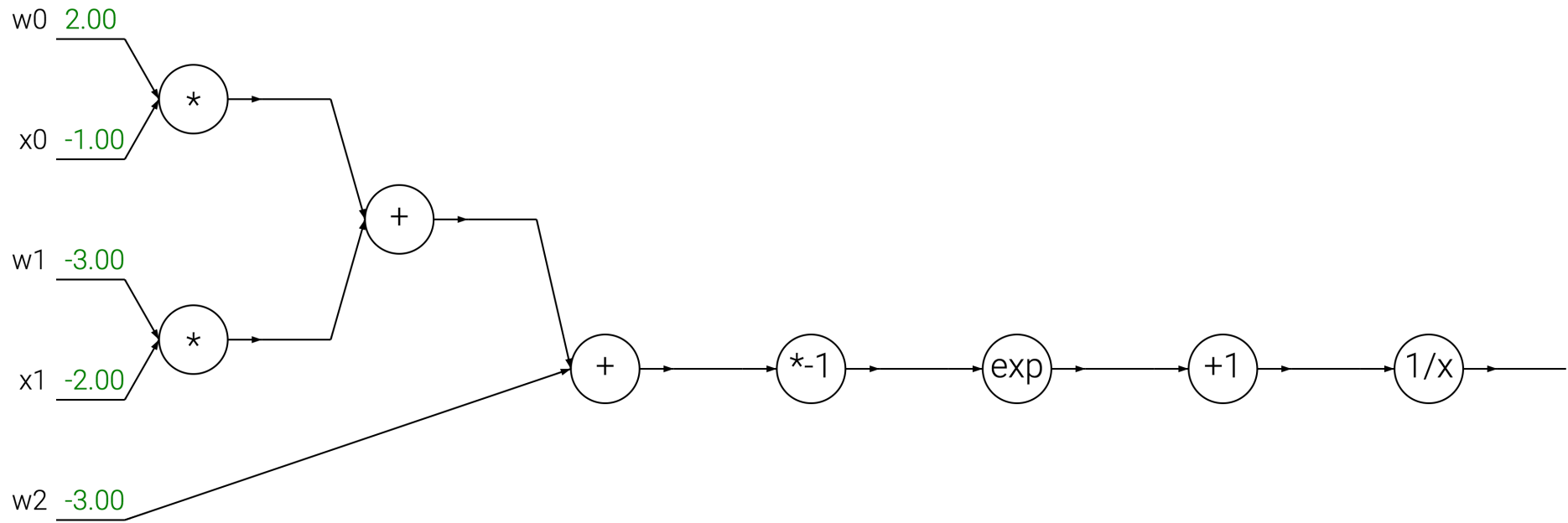
# Backpropagation: Example 1



Example from:

<http://cs231n.github.io/optimization-2/>

# Backpropagation: Example 2



Example from:

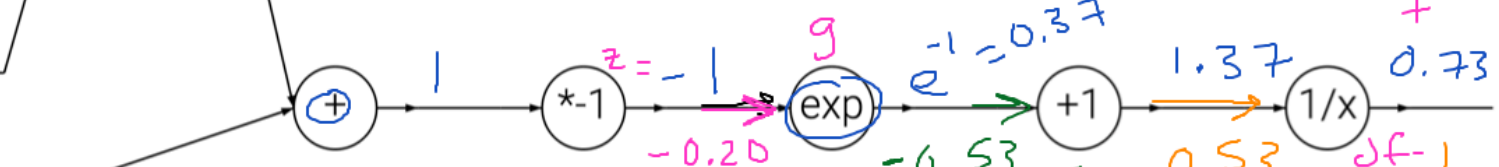
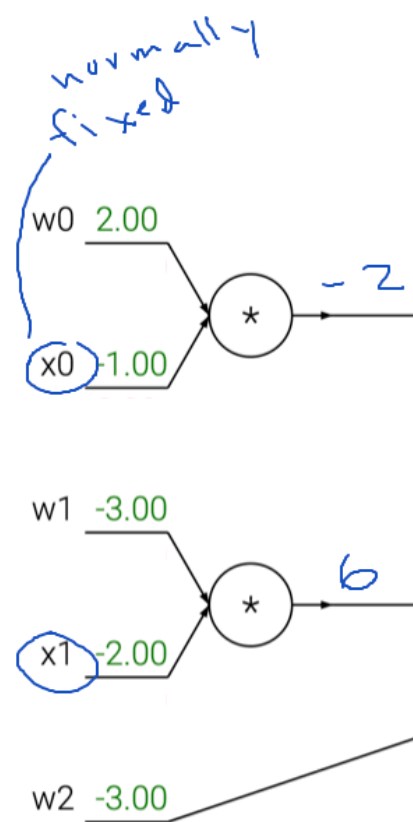
<http://cs231n.github.io/optimization-2/>

# Backpropagation: Example 2

logistic

$$f(w_0, w_1, w_2, x_0, x_1)$$

$$= \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(z) = f(g(z))$$

$$g(x) = e^x$$

$$g'(x) = e^x$$

$$g'(-1) = e^{-1} = 0.37$$

$$g(x) = x + 1$$

$$g'(x) = 1$$

$$g'(0.37) = 1$$

$$g(x) = \frac{1}{x}$$

$$g'(x) = -\frac{1}{x^2}$$

$$g'(1.37) = -\frac{1}{(1.37)^2} = -0.53$$

input to node g backprop so far

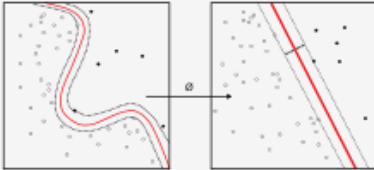
$$\frac{df}{dz} = \frac{dg}{dz} \cdot \frac{df}{dg} = (0.37) \cdot (-0.53)$$

# Outline for December 1

- Training a NN: backpropagation
- **Begin unsupervised learning**
- K-means clustering
- Next time
  - Gaussian Mixture Models (GMM)
  - Principal Component Analysis (PCA)

**Supervised Learning:**  
makes use of examples where we know the underlying “truth” (label/output)

**Machine learning and data mining**



**Problems** [show]

**Supervised learning** [hide]  
(classification · regression)

Decision trees · Ensembles (Bagging, Boosting, Random forest) · *k*-NN · Linear regression · Naive Bayes · Neural networks · Logistic regression · Perceptron · Relevance vector machine (RVM) · Support vector machine (SVM)

**Clustering** [hide]

BIRCH · Hierarchical · *k*-means · Expectation-maximization (EM) · DBSCAN · OPTICS · Mean-shift

**Dimensionality reduction** [hide]

Factor analysis · CCA · ICA · LDA · NMF · PCA · t-SNE

**Structured prediction** [hide]

Graphical models (Bayes net, CRF, HMM)

**Anomaly detection** [hide]

*k*-NN · Local outlier factor

**Neural nets** [hide]


Autoencoder · Deep learning · Multilayer perceptron · RNN · Restricted Boltzmann machine · SOM · Convolutional neural network

**Reinforcement Learning** [hide]

Q-Learning · SARSA · Temporal Difference (TD)

**Theory** [show]

**Machine learning venues** [show]

 **Machine learning portal**

V · T · E

**Unsupervised Learning:**  
Learn underlying structure or features without labeled training data

# Unsupervised learning: 3 main areas

- 1) Clustering: group data points into clusters based on features only
- 2) Dimensionality reduction: remove feature correlation, compress data, visualize data
- 3) Structured prediction: model latent variables (example: Hidden Markov Models)



# Unsupervised learning examples from biology: clustering

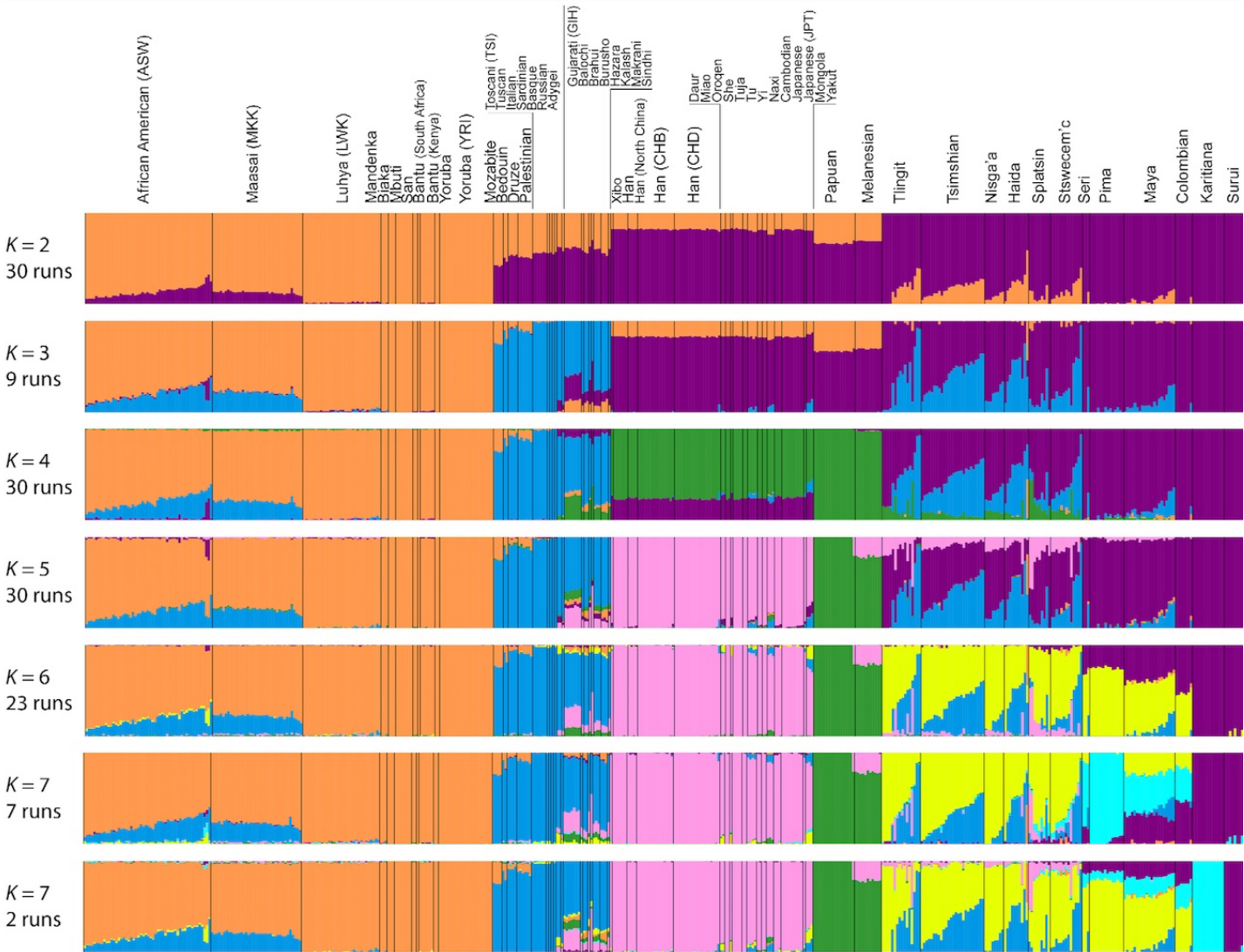
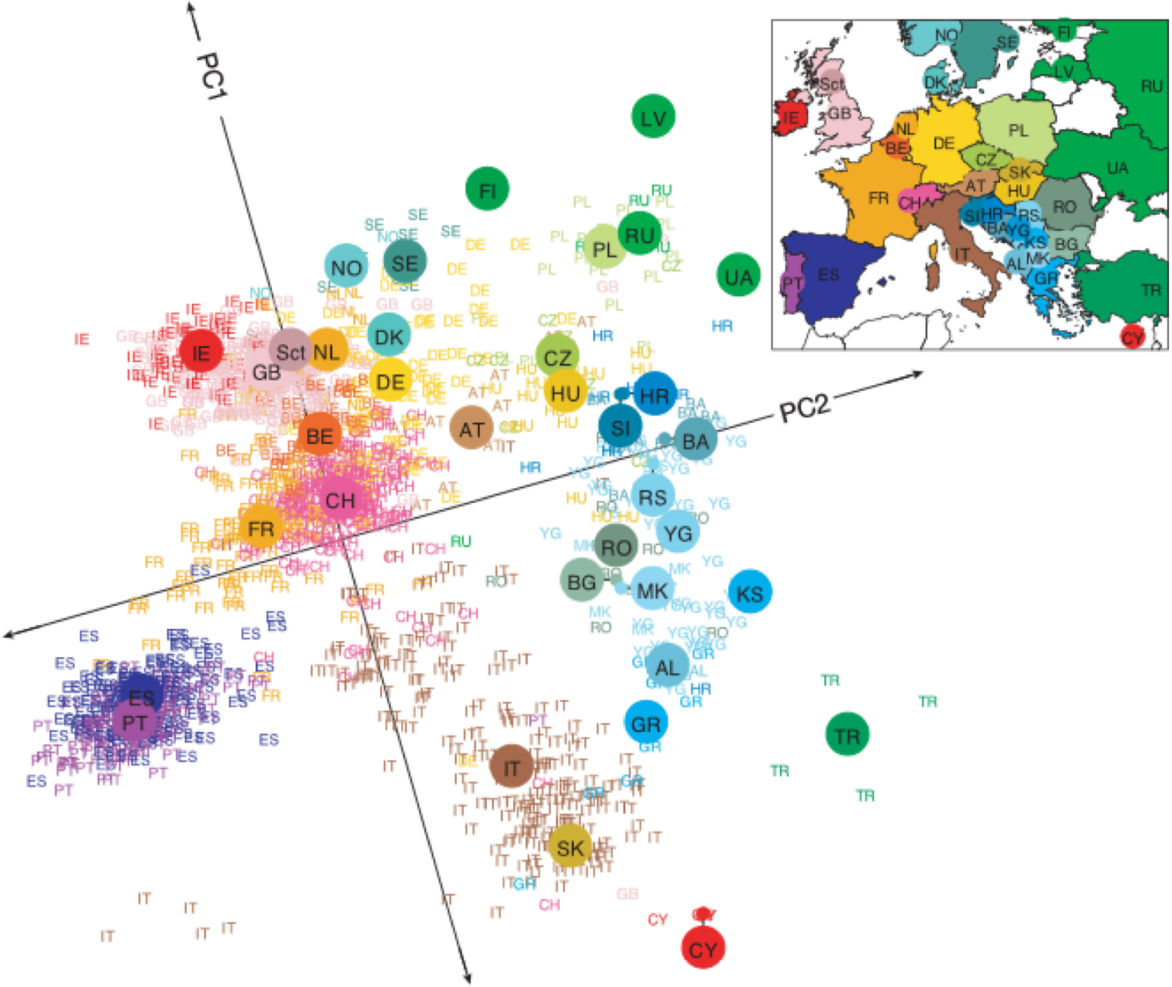
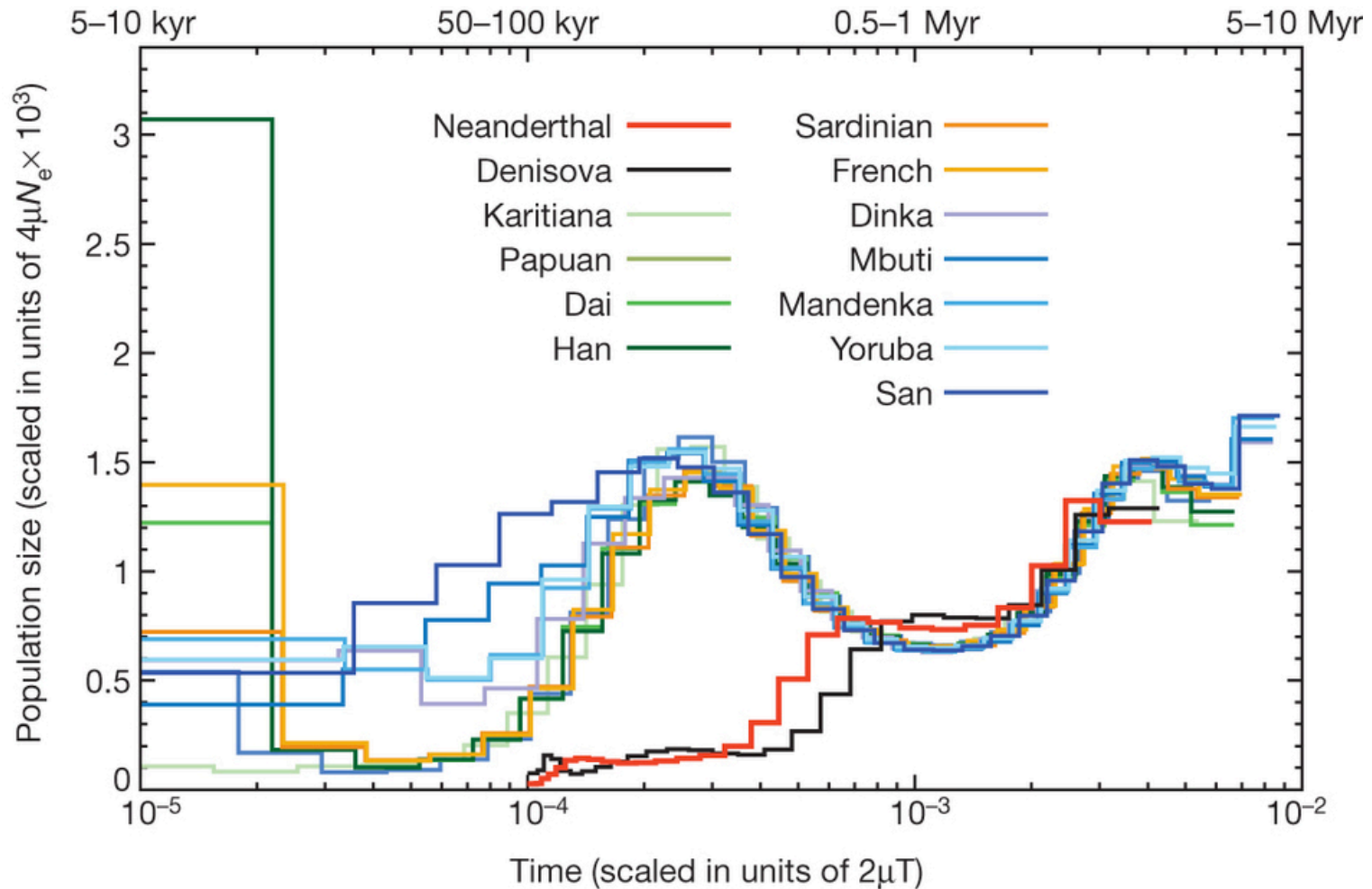


Figure: German Dziebel

# Unsupervised learning examples from biology: structured prediction



# Unsupervised learning examples from biology: structured prediction

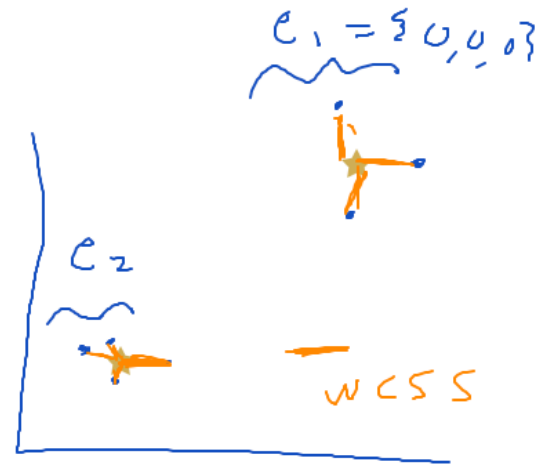


# Outline for December 1

- Training a NN: backpropagation
- Begin unsupervised learning
- **K-means clustering**
- Next time
  - Gaussian Mixture Models (GMM)
  - Principal Component Analysis (PCA)

# Clustering Goals

- Learn about underlying structure in the data
- Cluster new data (testing)
- Goal: minimize “within cluster sum of squares” (WCSS)



find  $\{c_1, c_2, \dots, c_k\} = c$  s.t.

WCSS  $\rightarrow$  (cost)

$$J(c) = \sum_{k=1}^K \sum_{i \in c_k} \| \vec{x}_i - \vec{\mu}_k \|^2$$

all clusters

train data

cluster mean

truly minimizing WCSS

$\Rightarrow$  NP-hard

# K-means algorithm

$n, p$

- 0) Initialization: choose means (centers) of K clusters

$$\vec{\mu}_1^{(1)}, \vec{\mu}_2^{(1)}, \dots, \vec{\mu}_K^{(1)}$$

← typically from training data

- 1) **E-step (assignment)**: for each training example, find closest mean

- 2) **M-step (update)**: compute new means