Lightning Guide to Machine Learning and Deep Learning Neural Networks

(Learning about machine learning using your neural networks to discover artificial neural networks that might be useful for automatically helping others do their jobs with their own neural networks in analyzing still other people’s neural networks…and other stuff.)

Supervised vs. Unsupervised (and Semi-Supervised) Learning

- Supervised: Data classification based on examples with class labels. Use the data and class assignment to infer a mapping function for new data
- Unsupervised: Automatically learning relationships within unlabeled data
- Semi-Supervised: Maybe guess some labels for unlabeled data and/or use those to label or classify other data.

Supervised Learning

- Data classification based on examples with class labels

Weight (kg)	Tail Length (cm)	Color	Cat Class
3.6	32	ORANGE	DOMESTIC
180	168	STRIPED	TIGER

- Use the data and class assignment to infer a mapping function for new data
- Algorithms: Aritificial Neural Networks (ANNs), Support Vector Machines (SVMs), logistic regression (LR), Naïve Bayes Classifier, K Nearest Neighbor (KNN), Decision Trees

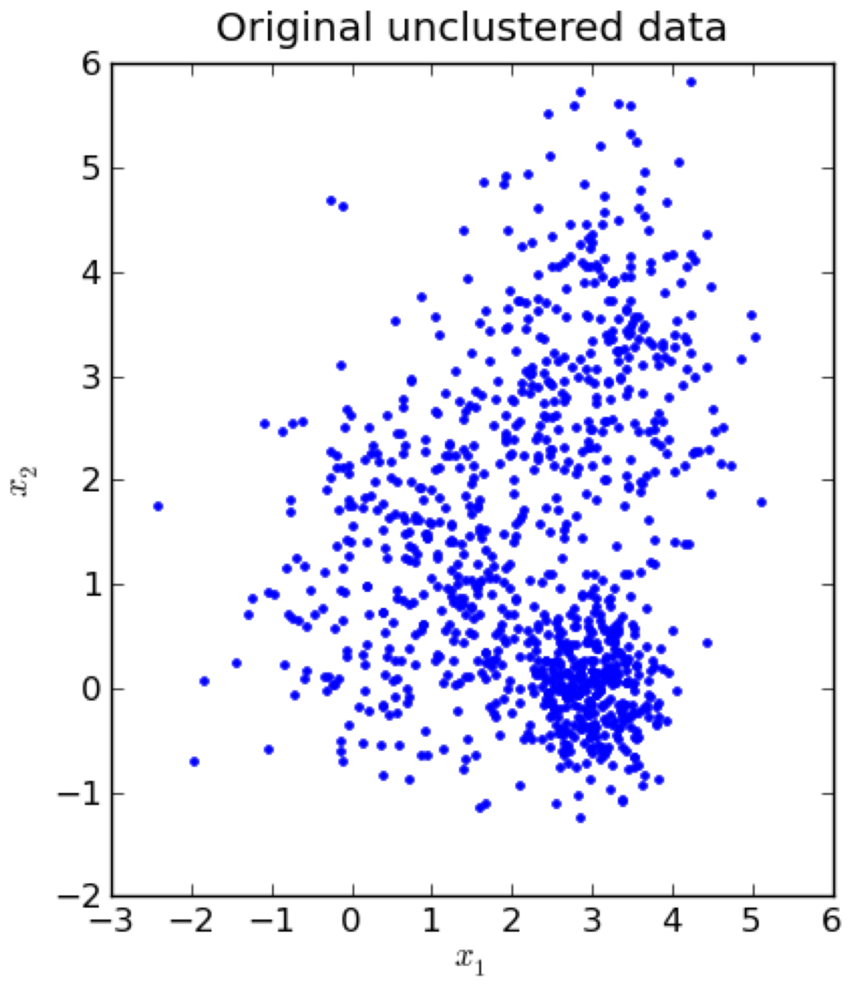
Unsupervised Learning

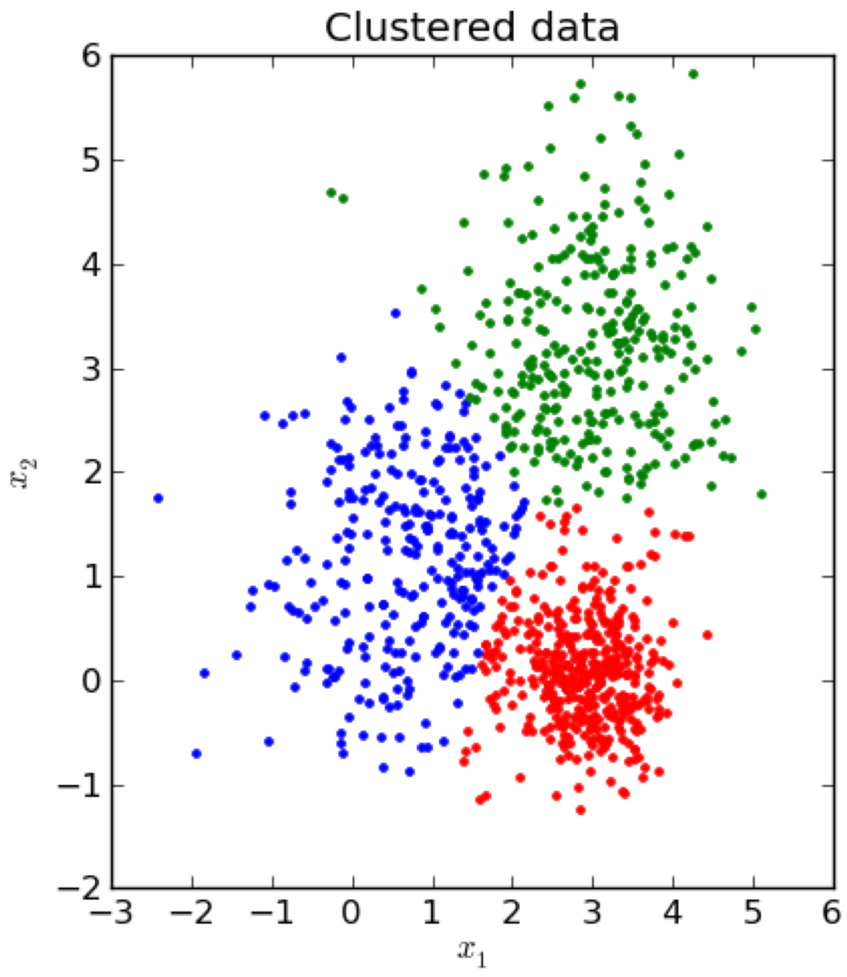
- Data organization without labels

Weight (kg)	Tail Length (cm)	Color
3.6	32	ORANGE
180	168	STRIPED

- Organize the data based on patterns within it.
- Algorithms: Clustering (k-means, agglomerative, top-down, etc.), Self-Organizing Maps (SOMS)

Example Clustering

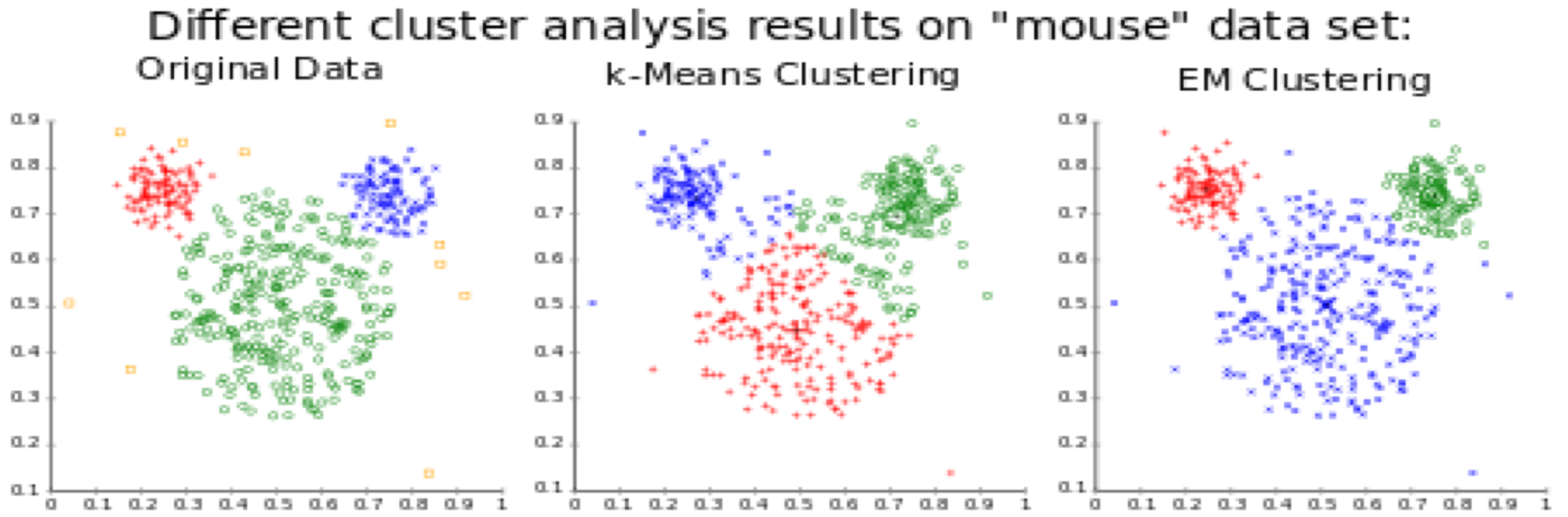




Example Clustering

- For each data point, find the nearest data point using some distance metric (like ”straight line” or Euclidean distance)
- Add those two data points to Cluster A
- Take the point midway between the data points in A and now add the nearest data point to A
- Repeat until some criteria is reached for A (like the only remaining points are too far away from the midpoint of A)
- Start again on the remaining points and put them in B.
- Etc.

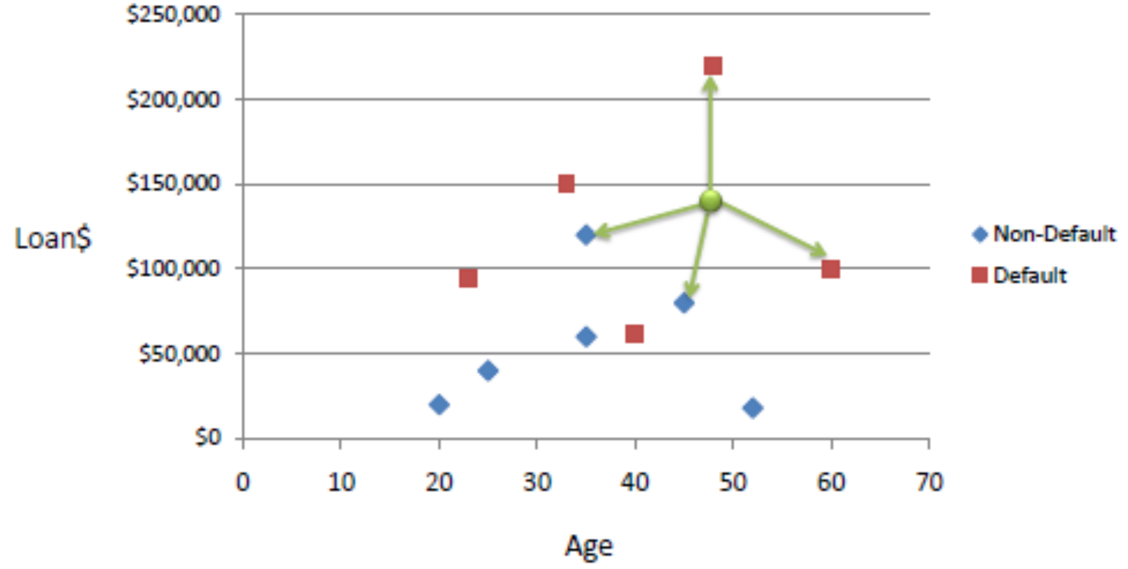
More on Clustering



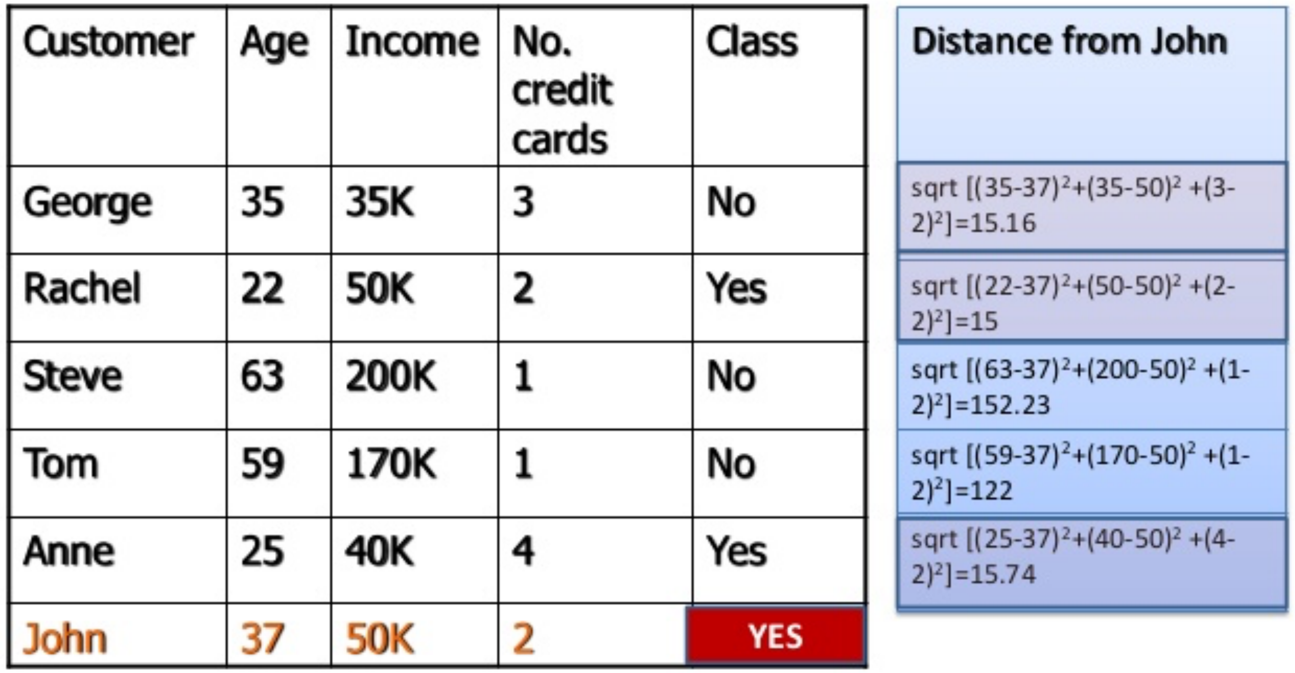
- Different “metrics” result in different results
- Assumes some kind of “separability in the data”
- Can’t label the data very easily (the ”cluster labeling problem”)

Supervised Learning

- Example: K Nearest Neighbor algorithm



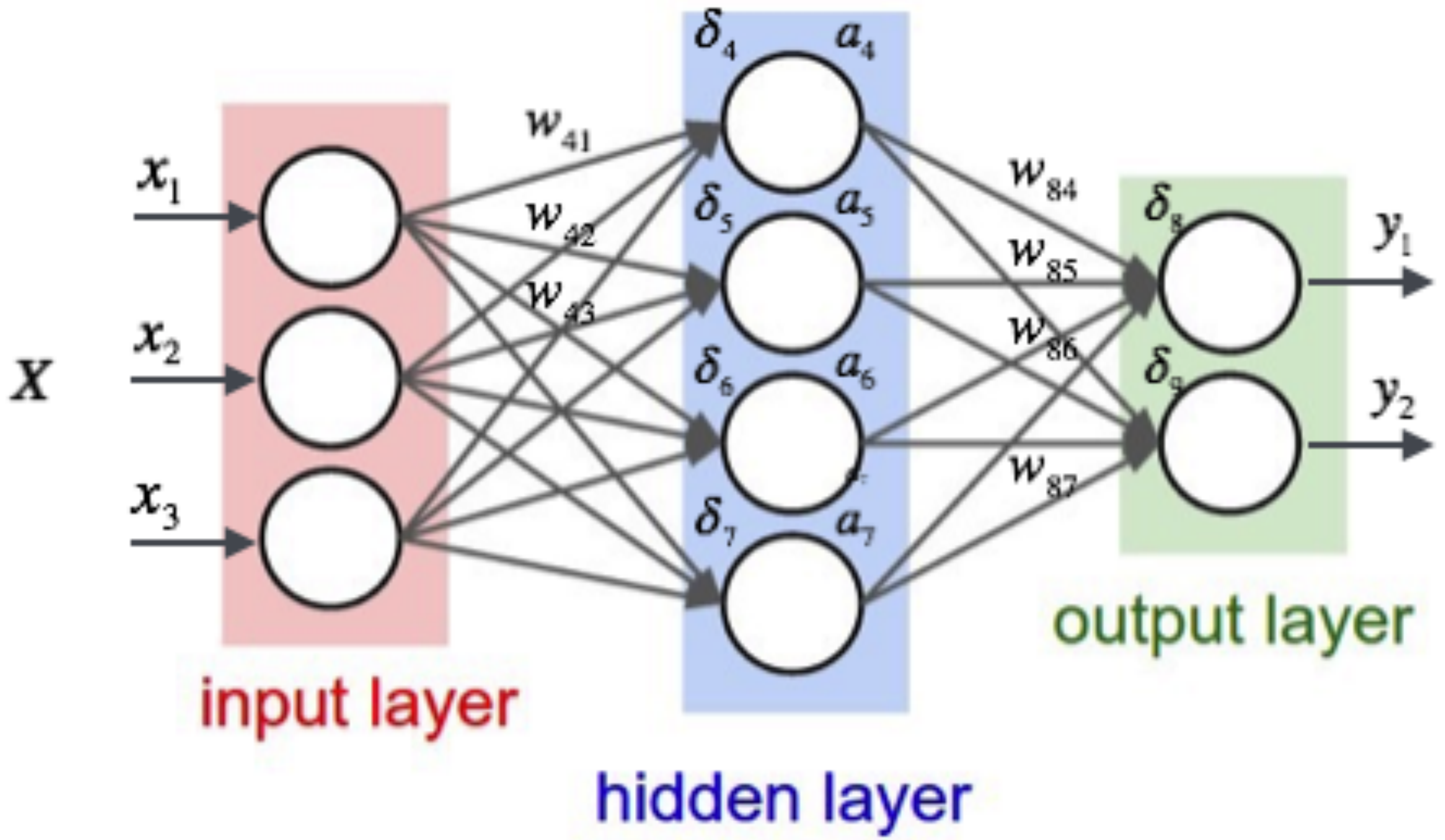
3-KNN: Example(1)



Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2 +(3-2)^2]$=15.16
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2 +(2-2)^2]$=15
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2 +(1-2)^2]$=152.23
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2 +(1-2)^2]$=122
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2 +(4-2)^2]$=15.74
John	37	50K	2	YES	

Supervised Learning: Artificial Neural Networks (ANNs)

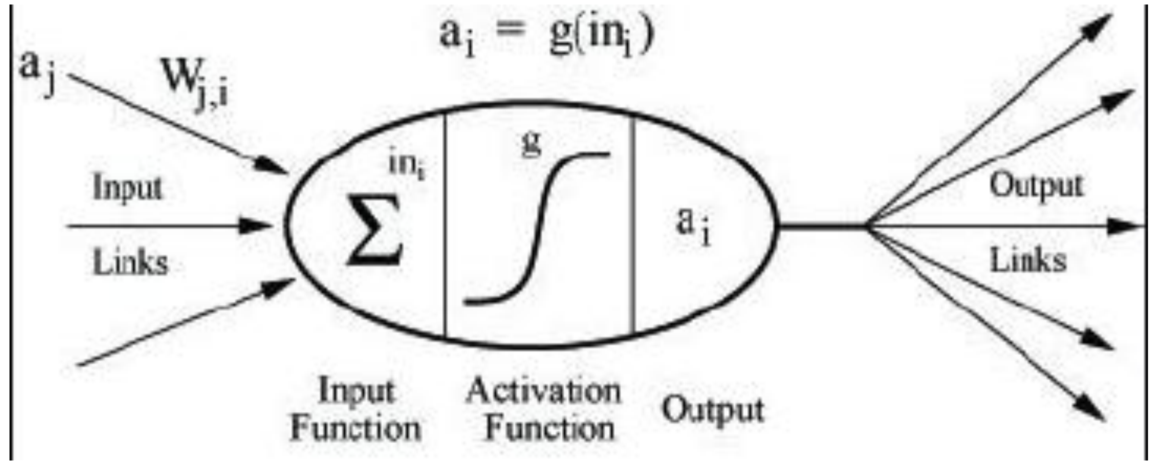
- Loosely based on early models of neurons
- The axon is connected to the dendrite
- When neuron fires, the dendrites transmit a weighted signal to the next neuron



X
x_1
x_2
x_3
input layer
w_{41}
w_{42}
w_{43}
δ_4 a_4
δ_5 a_5
δ_6 a_6
δ_7 a_7
hidden layer
w_{84}
w_{85}
w_{86}
w_{87}
δ_8
δ_9
output layer
y_1
y_2
Y

Training vs. Classification in ANNs

- Training phase:
  - Initialize network weights to random real values [0,1]
  - Present data vectors to input layer (suitably encode data values)
  - Multiply each value by the weights and apply some kind of non-linear (or semi-linear) firing threshold (activation function) for hidden layer nodes

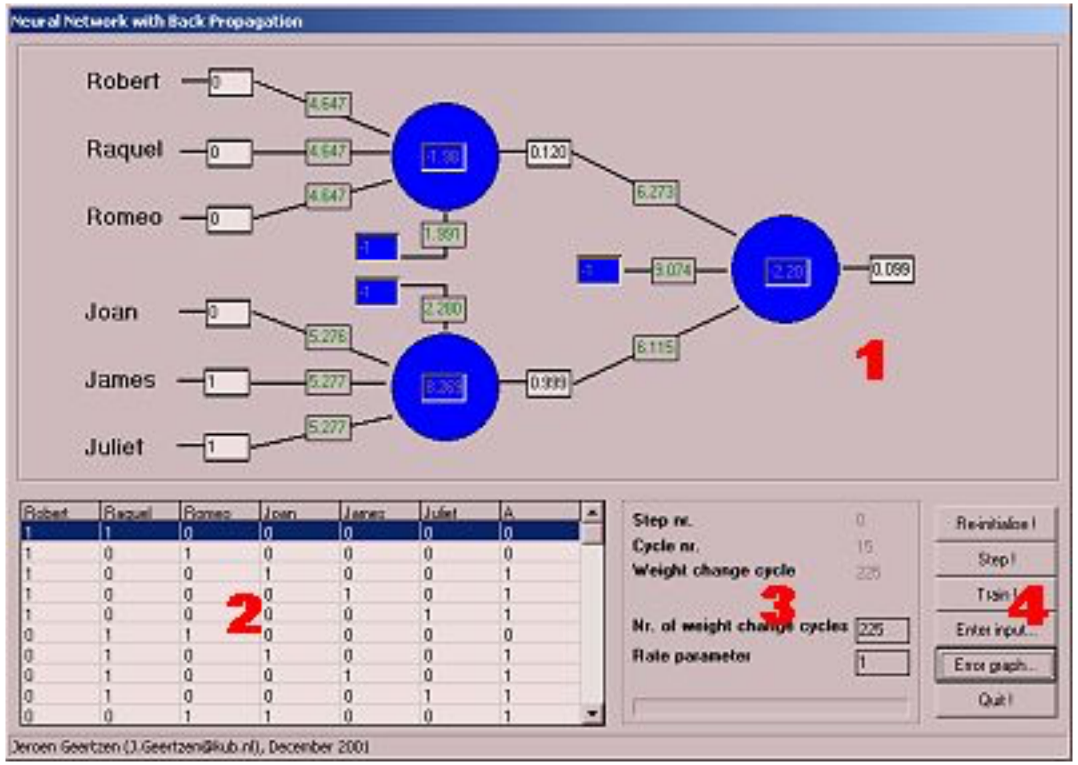


Training vs. Classification

- Carry activation through to output layer
- Calculate the error at the output layer (the difference between the output layer activation value that should be for the input, and the actual output)
- Propagate the error back by adjusting the weight values a little bit
- Do it again through multiple random training presentations through multiple training “epochs” until the error becomes very low
- This is “backpropogation” training
- Then, you can classify new data by presenting the new data to the network and reading the output layer classes

Backpropagation Example

- https://nlp-ml.io/jg/software/ann/

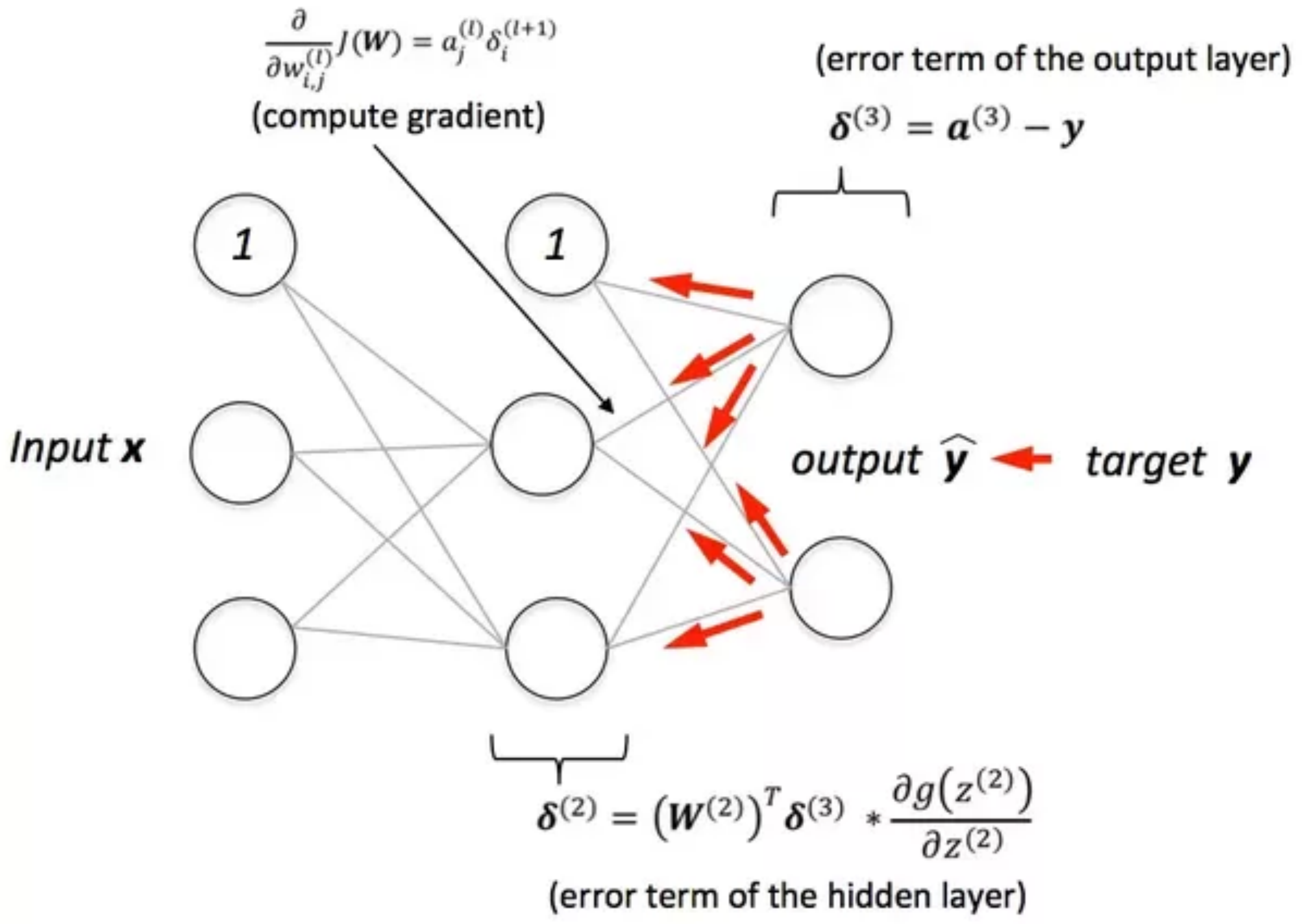


Robert	Raquel	Romeo	Joan	James	Juliet	A
1	1	0	0	0	0	0
1	0	1	0	0	0	0
1	0	0	1	0	0	1
1	0	0	0	1	0	1
1	0	0	0	0	1	1
0	1	1	0	0	0	0
0	1	0	1	0	0	1
0	1	0	0	1	0	1
0	1	0	0	0	1	1
0	0	1	1	0	0	1

Step nr. 0
Cycle nr. 15
Weight change cycle 225
Nr. of weight change cycles 225
Rate parameter 1

Jeroen Geertzen (J.Geertzen@kub.nl), December 2001

More on Backprop



$\frac{\partial}{\partial w_{i,j}^{(l)}} J(\boldsymbol{W}) = a_j^{(l)} \delta_i^{(l+1)}$
(compute gradient)
(error term of the output layer)
$\boldsymbol{\delta}^{(3)} = \boldsymbol{a}^{(3)} - \boldsymbol{y}$
1
1
Input $\boldsymbol{x}$
output $\hat{\boldsymbol{y}}$
target $\boldsymbol{y}$
$\boldsymbol{\delta}^{(2)} = \left(\boldsymbol{W}^{(2)}\right)^T \boldsymbol{\delta}^{(3)} * \frac{\partial g\left(z^{(2)}\right)}{\partial z^{(2)}}$
(error term of the hidden layer)

Problems with ANNs

- Like all machine learning, they train on whatever distinguishes the data. They are too literal. Classic tank training problem (probably apocryphal).
- Fully connected ANNs are too big: 1024x728 pixel HD frame = 745472 pixels. One input node per pixel and as many hidden layer nodes equals 745472^2 weights that need calculation (forward pass) and updating (backpropagation), just on first layer! Just too much.
- This is the "curse of dimensionality" and it applies to images or sentences (too many possible words in natural language).
- Regular ANNs don't handle sequential data very well (video scenes, sentences, stock market price fluctuations, etc.) and "recurrent" neural networks that connect output back to input are too hard to train.

Deep Learning Neural Networks

- Bigger, better, more interesting
- Run them faster on GPUs or custom chips
- Specific architectural adjustments for handling images, sequential data, and external memory stores
  - Convolutional Neural Networks (CNN) for images
  - Treating network like an autoencoder first for images
  - Long Short-Term Memory (LSTM) for sequential data
  - Restricted Boltzman Machines (RBMs) as a variant on backpropagation
  - External memory models for things like Wikipedia-size ontologies

Convolutional Neural Networks

- For image analysis and scene classification
- Perform different 2D discrete convolutions on images
- Pool together the results
- Then feed them into a backprop neural network as the input and train, train, train

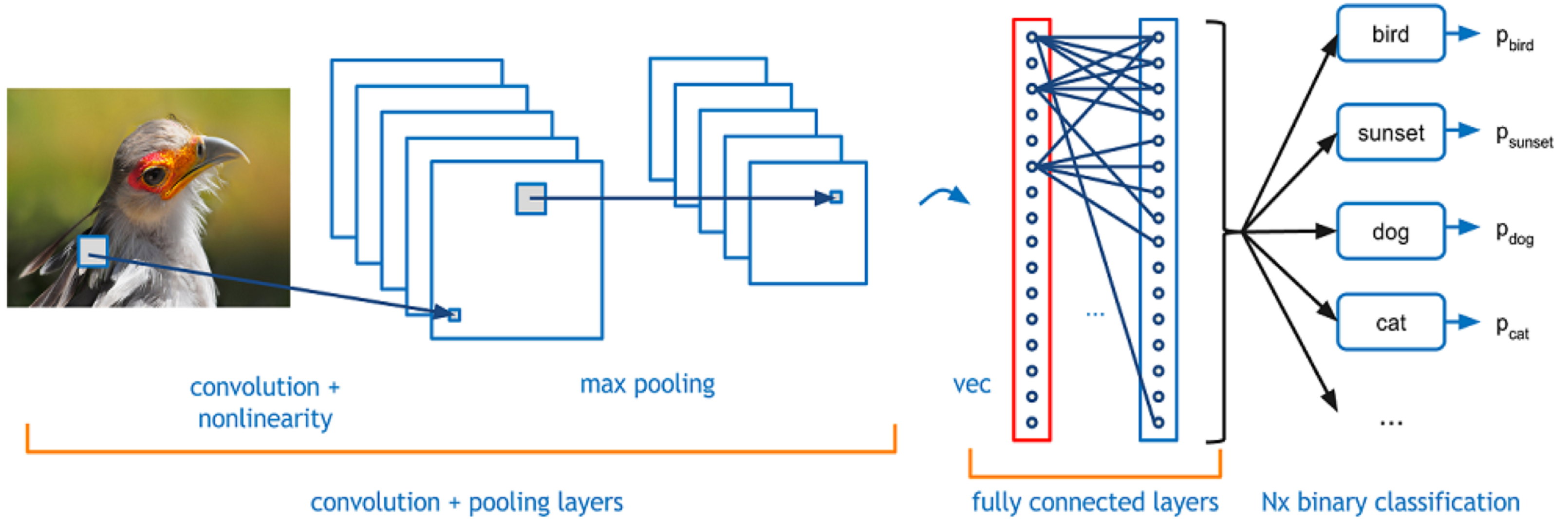
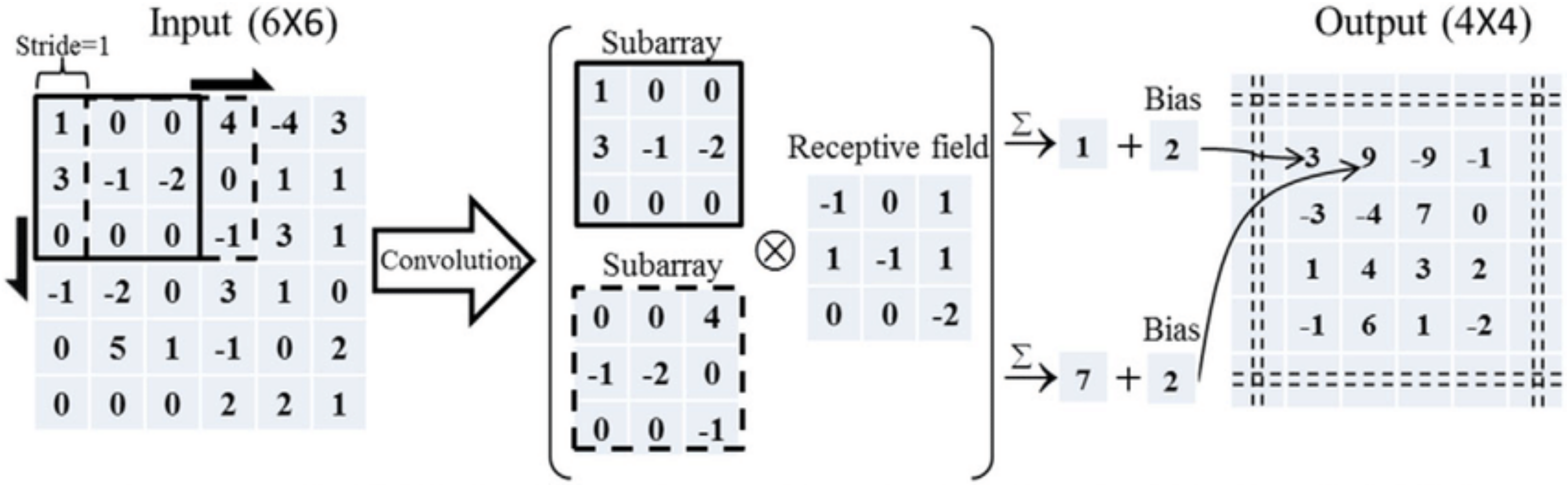
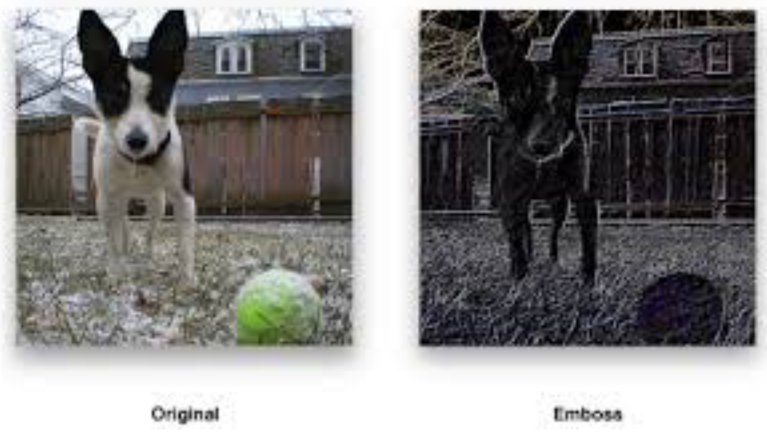


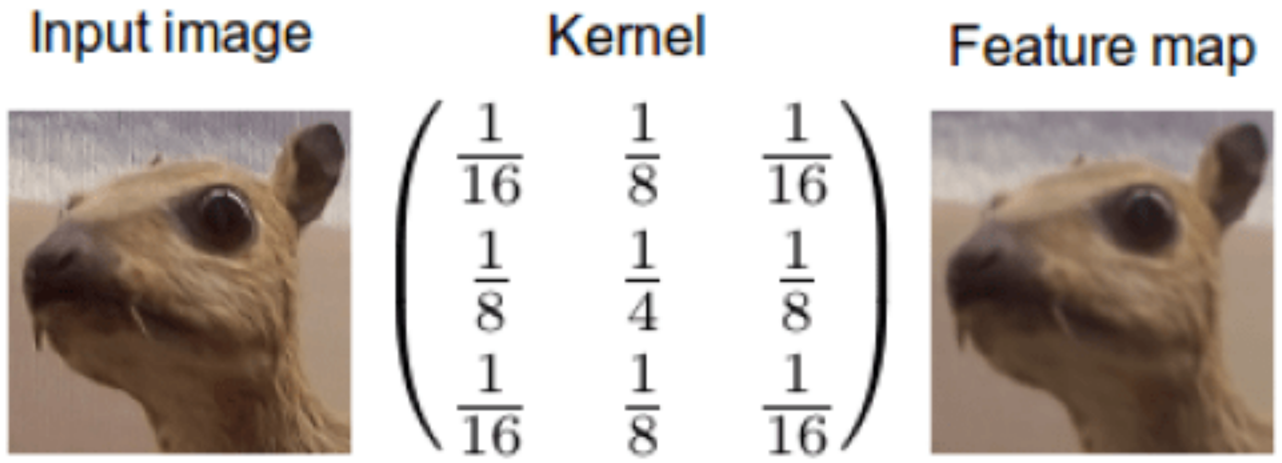
Image Convolution



Output size = (I – R) / S + 1, I = Input size, R = Receptive field size, S = Stride size; (6-3)/1+1=4

Different Kernels Do Different Things





$$\begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$$

Some Design and Experimental Terminology

- Training/Test data splits: often we split data into ten parts (randomly), train on 9 parts and evaluate on the last part which is called the “held out data.”
- If the hidden layer is too large, a single node can become assigned to each output class resulting in ”overtraining” meaning that the network is not properly generalizing. By using a smaller hidden layer we “bottleneck” the system, forcing generalization.

Next Lecture

- TensorFlow practicalities
  - Download
  - Install
  - Run
  - Understanding the code base
  - APIs and language bindings
- Programming GPUs
  - Memory models
  - A bit about CUDA