Deep Learning

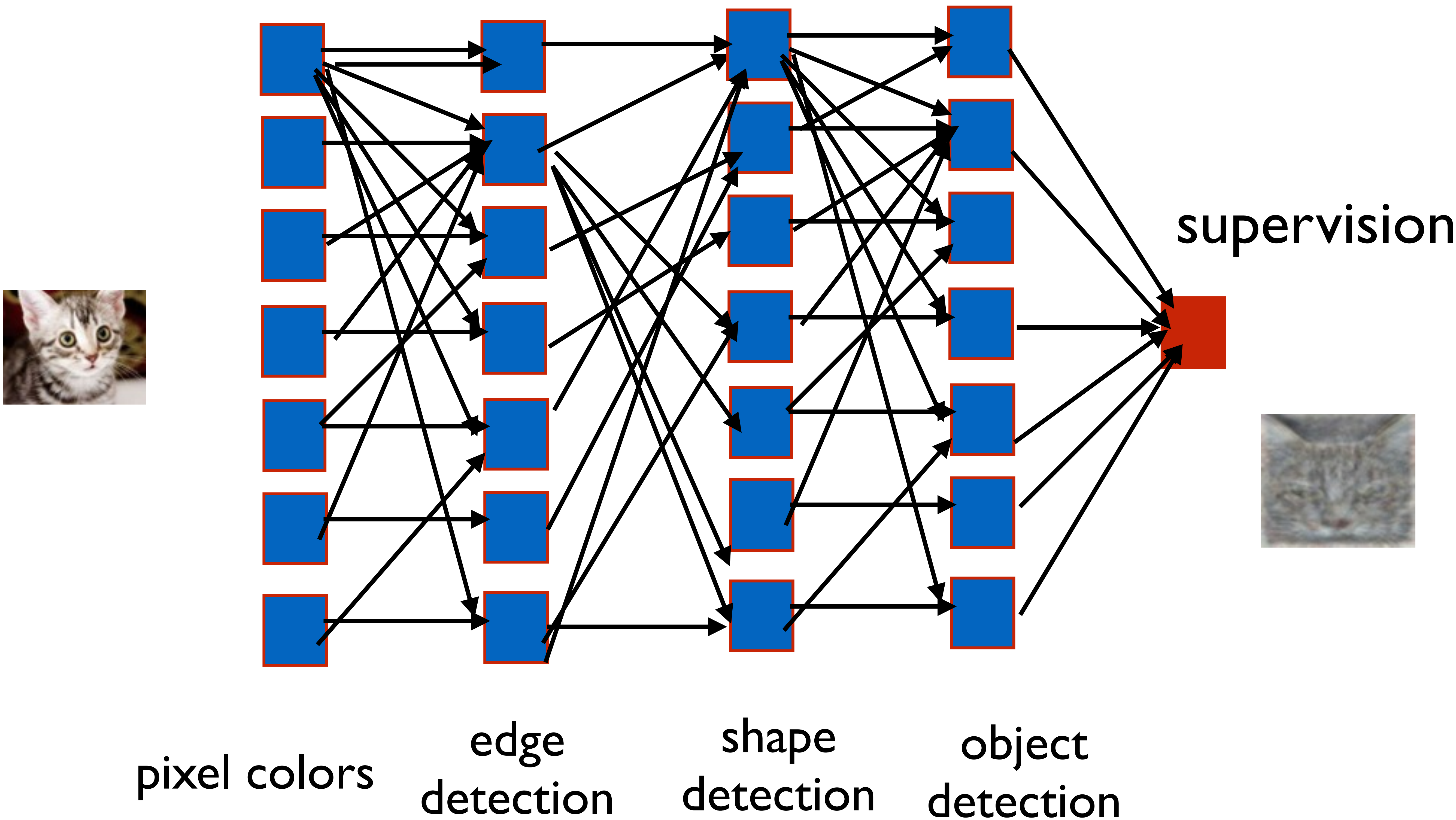
COMP 527
Danushka Bollegala

UNIVERSITY OF LIVERPOOL

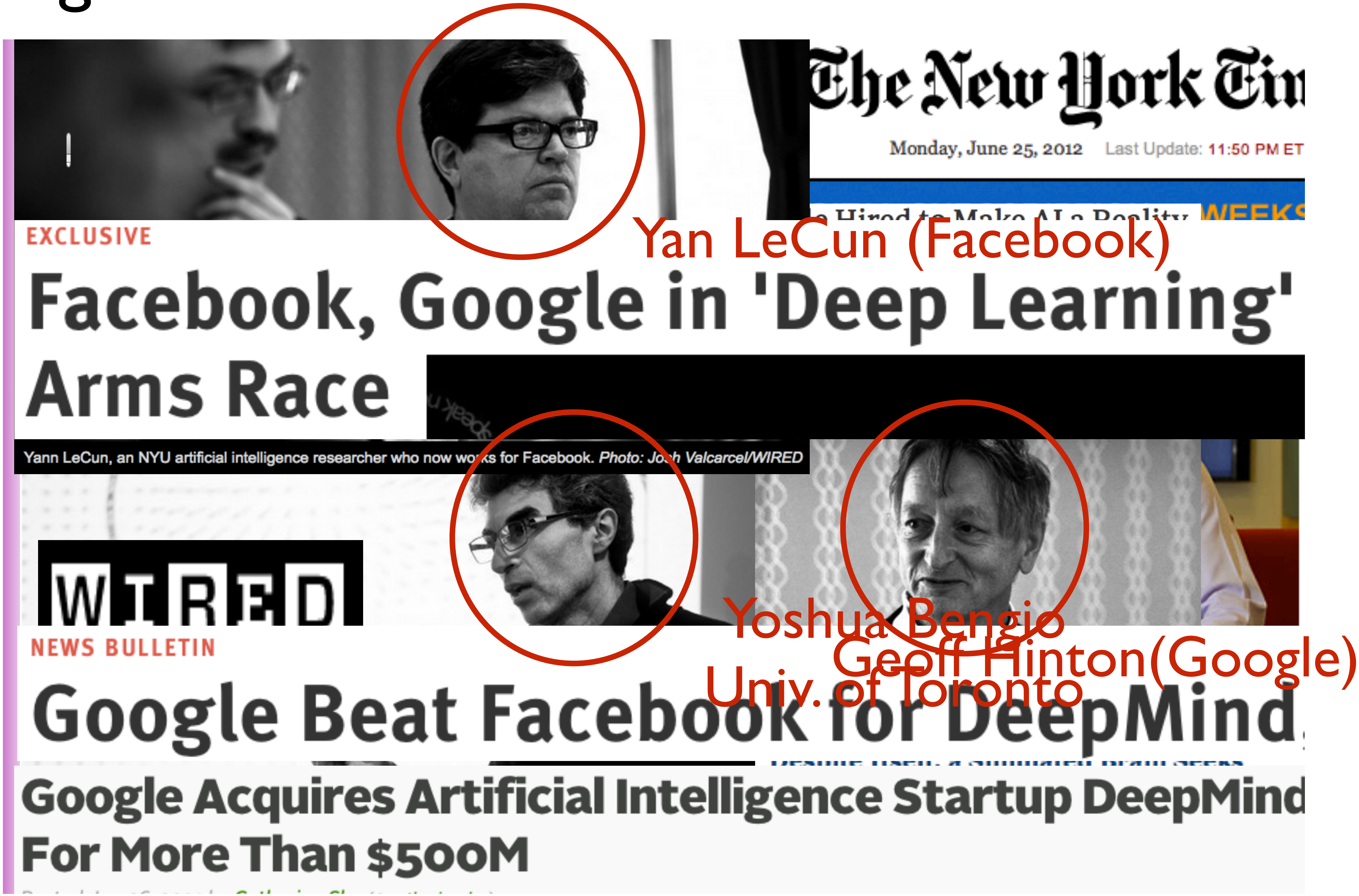
Deep Learning

- So far, in all our machine learning, we designed features by ourselves. But can we do this automatically?
  - feature learning from data
- How can we combine different types of features and decide the useful combinations as part of the learning process?
- One solution
  - kernels
    - Considers only fixed, limited, and specific combinations. (eg. polynomial kernel considers only pairwise combinations)
- Another solution
  - Multi-layer perceptrons
    - Overfitting, difficulty to train, time consuming
- Can we train deep models (with many hidden layers) efficiently and without over-fitting?
  - This is the central problem considered in *Deep learning*

Deep Learning at a glance



Big news



slide credit: Bengio KDD'14

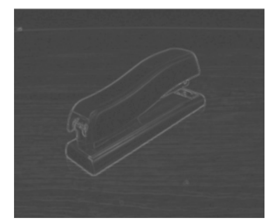
Applications of DL

- Image Recognition
  - ILVSCR 14, 15: outperformed human level
- NLP
  - Machine translation, text similarity, sentiment analysis
- Voice
  - Voice recognition
- Robotics
  - DeepMind, Computer Games, Reinforcement Learning

Object Recognition

input → features → classifier

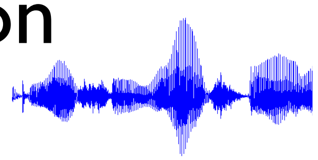
image

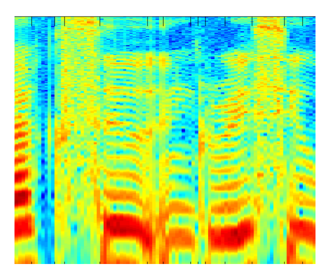


features

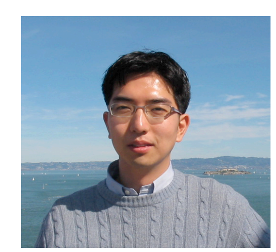
recognition

speaker identification





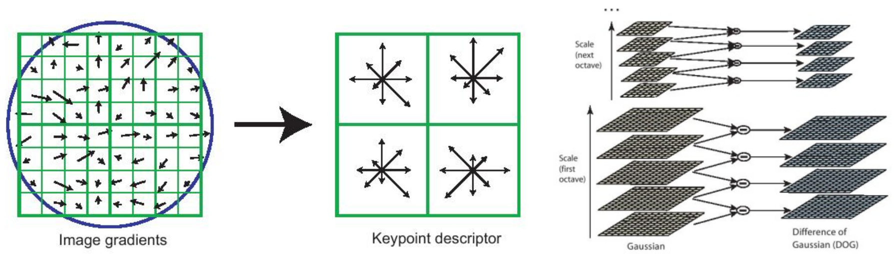
features



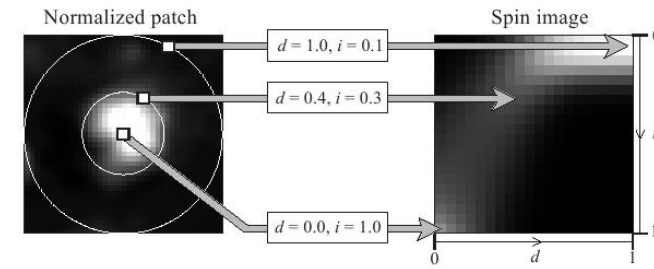
detection

slide credit: Honglak Lee

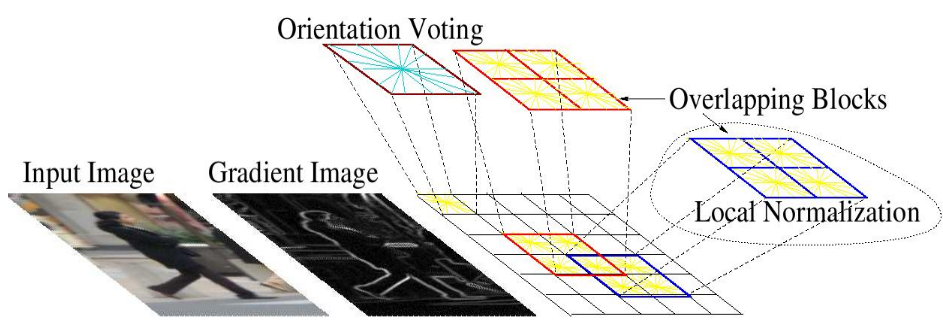
Image Features



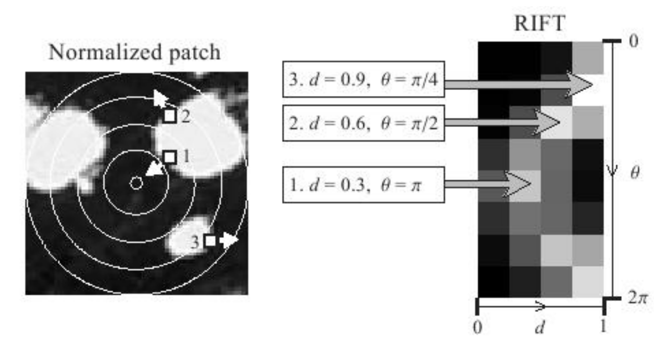
SIFT



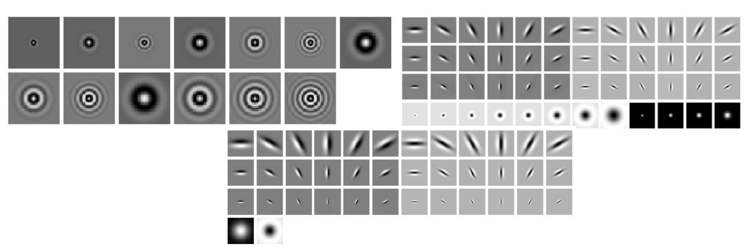
Spin image



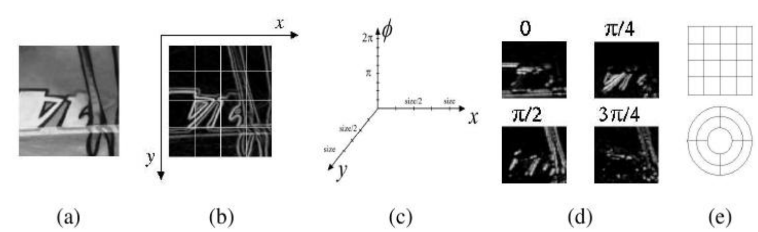
HoG



RIFT



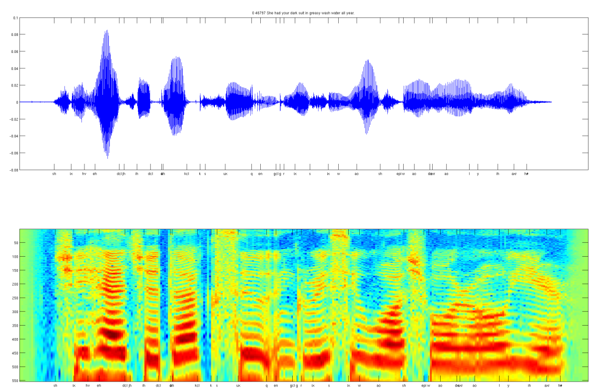
Textons



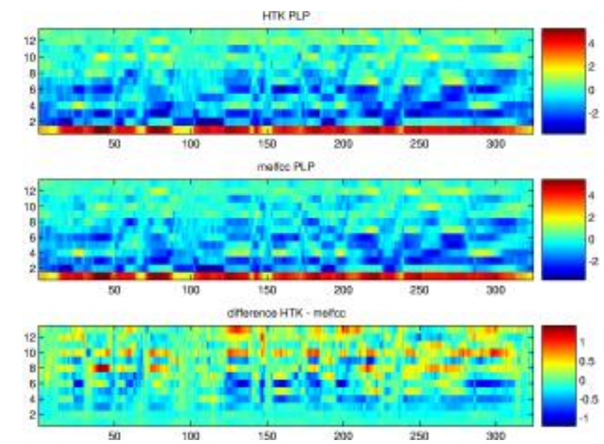
GLOH

Slide Credit: Honglak Lee

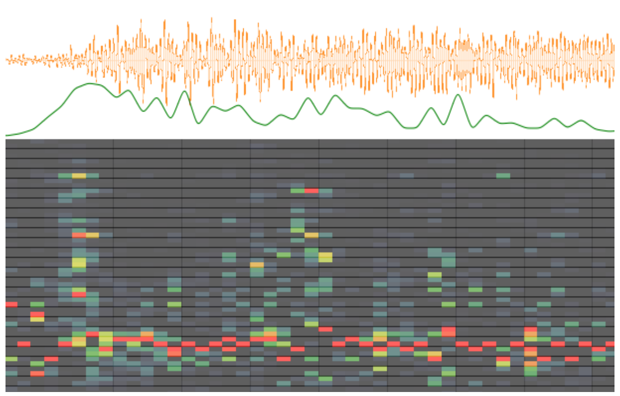
Voice Features



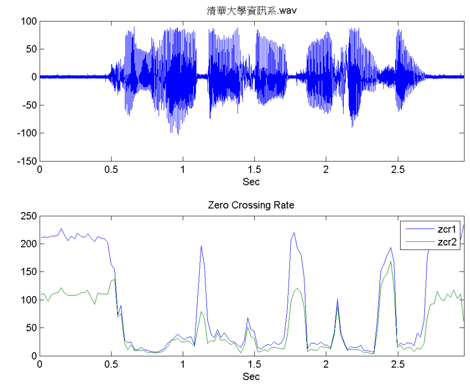
Spectrogram



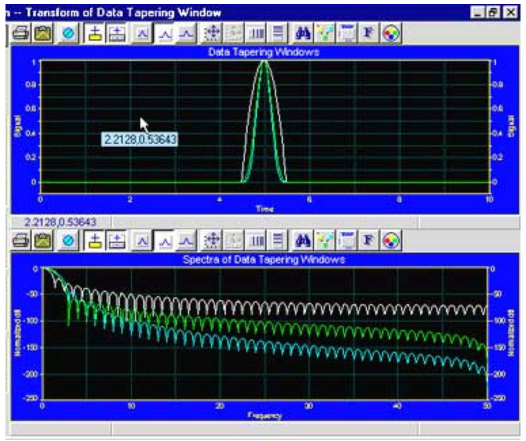
MFCC



Flux

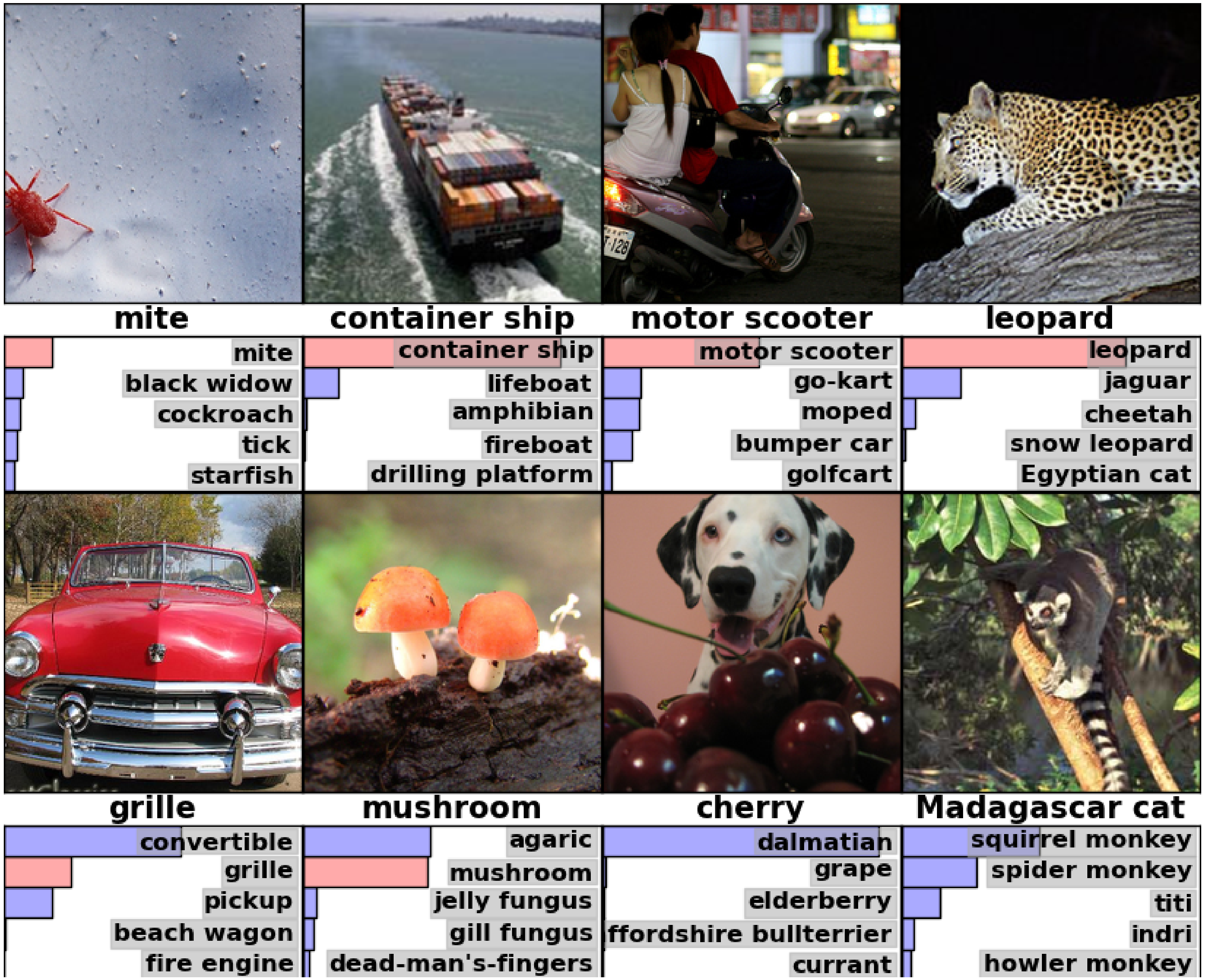


ZCR



Rolloff

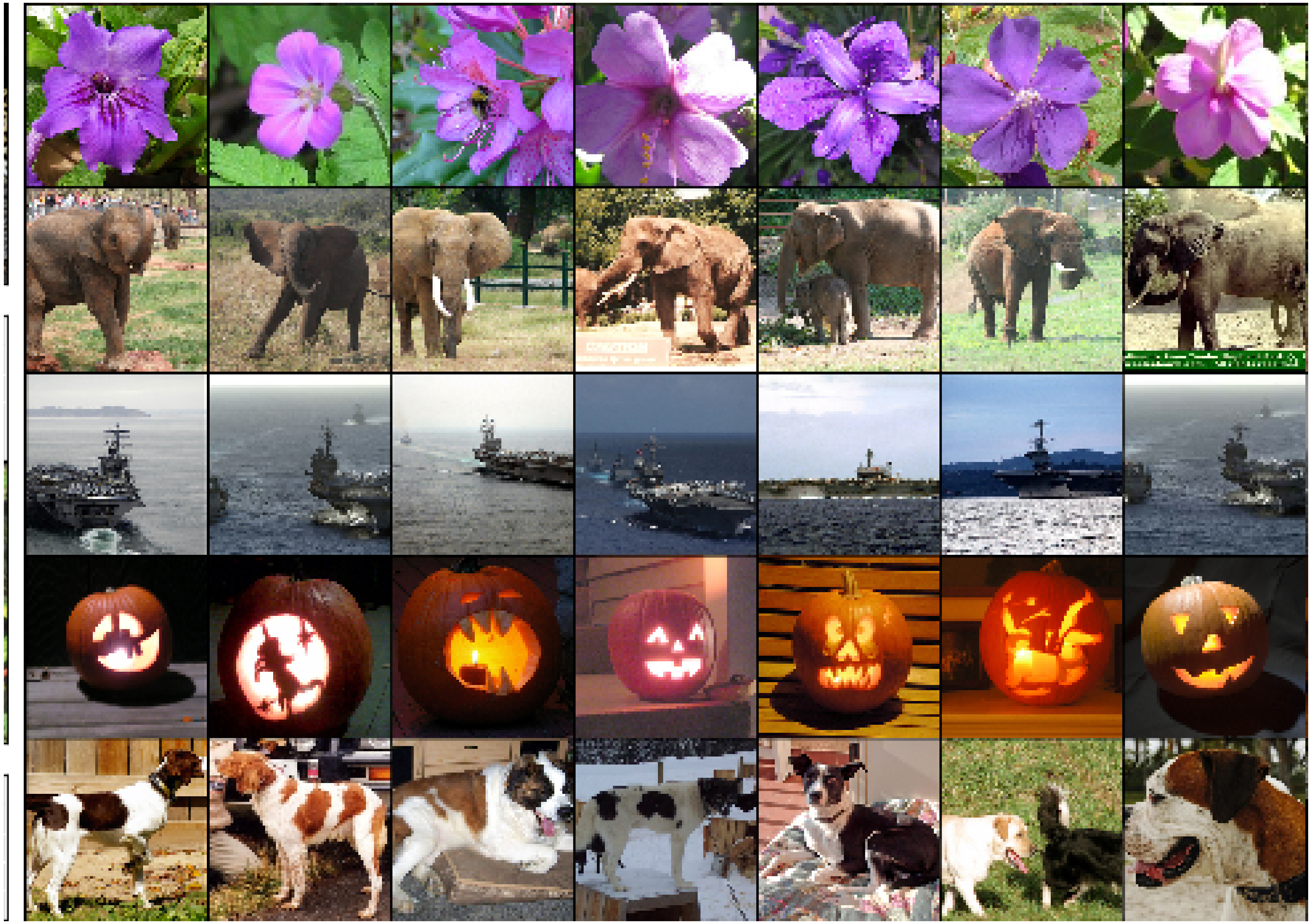
Image labeling



Krizhevsky+ NIPS'12

- demo：http://deeplearning.cs.toronto.edu/

Similar image retrieval



Krizhevsky+ NIPS'12

20 layer NN by Google!

Detecting roads from satellite images

Word Analogy Detection

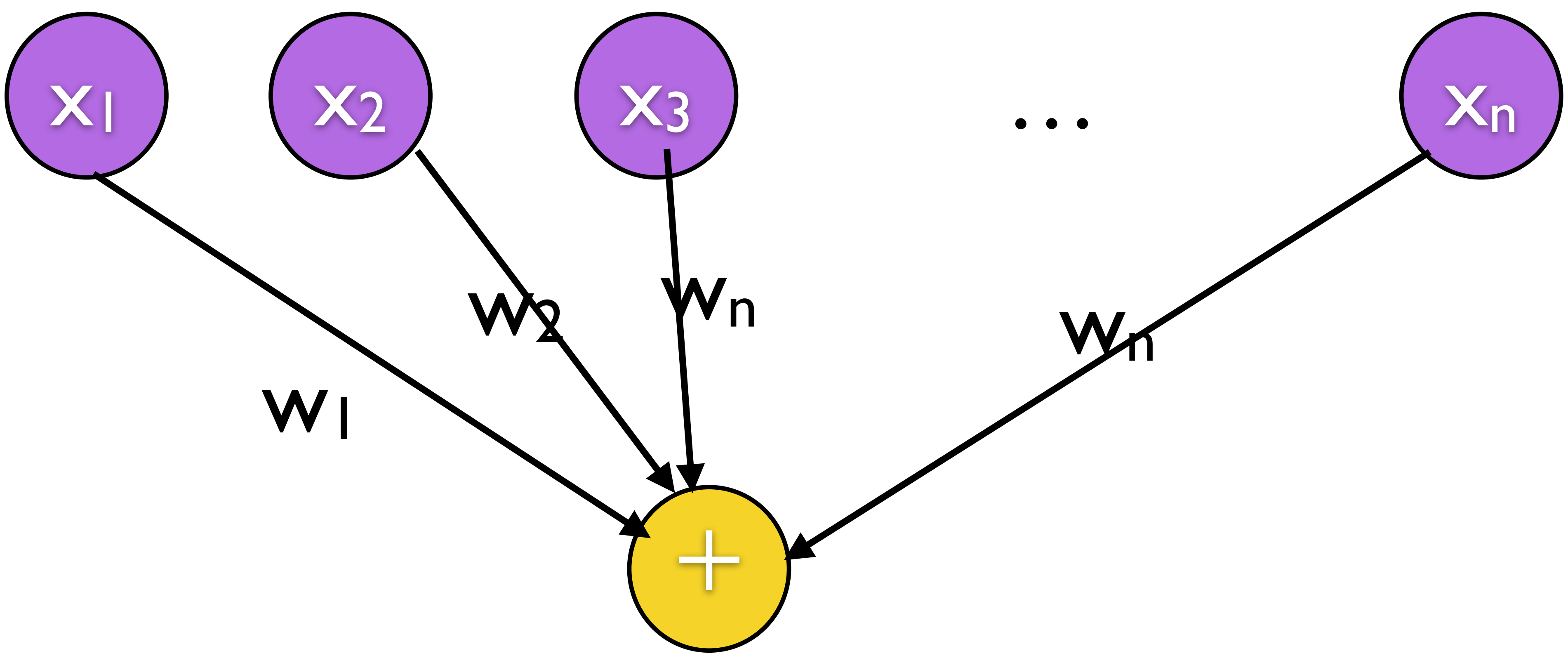
- How to learn representations for words?
- *you shall know a word by the company it keeps* — Firth
- Distributional Hypothesis
  - We can predict the meaning of a word by looking into its local context
- Can we learn vector representations for words such that we can accurately predict its neigbours in a sentence?
  - word2vec (skip-gram model) Mikolov+13
  - GloVe (Global Vector Prediction) Pennington+14
- v(king) - v(man) + v(woman) = v(queen)

Brief History

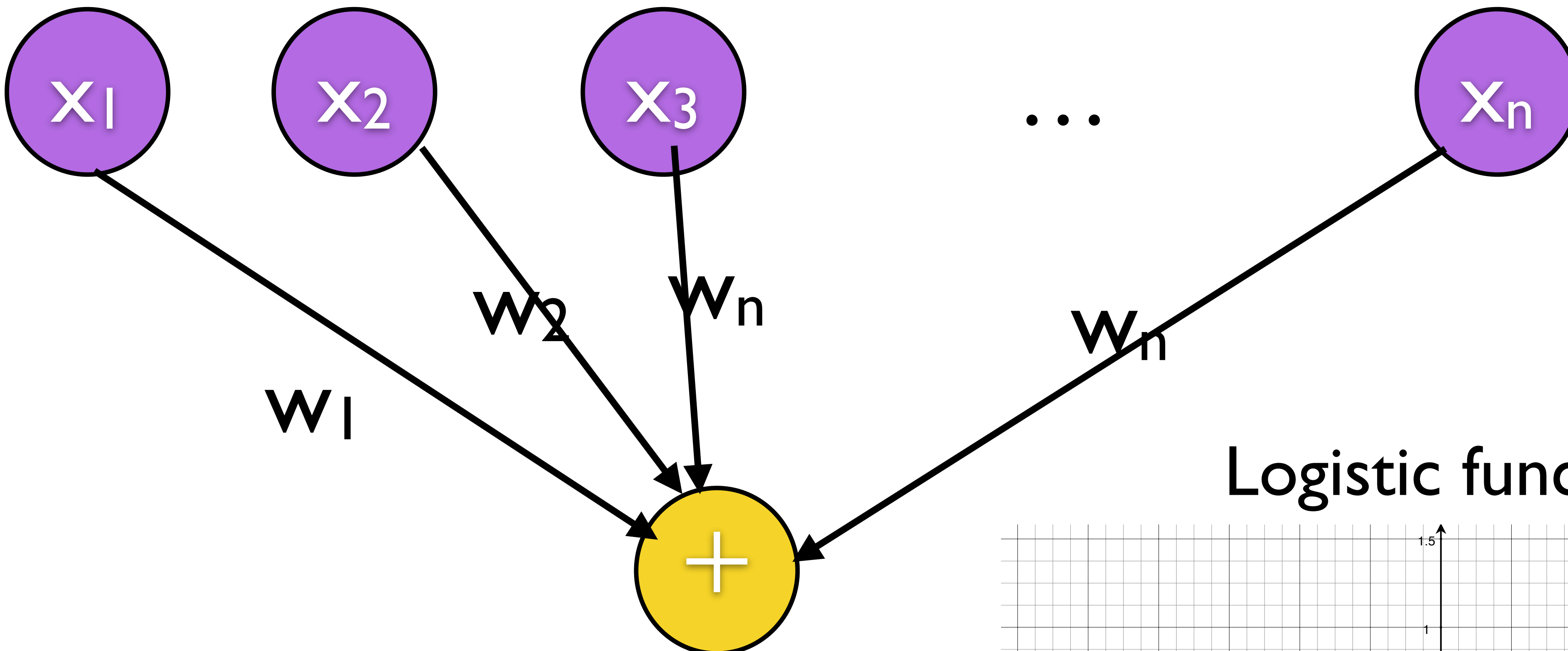
- Neural networks were around even back in 1950s
- It was shown that you cannot learn non-linearly separable data using single layer neural networks (ca. Perceptron)
  - Marvin Minsky [1960]
- First NN winter

Perceptron (revision)

- Perceptron is a single layer neural network

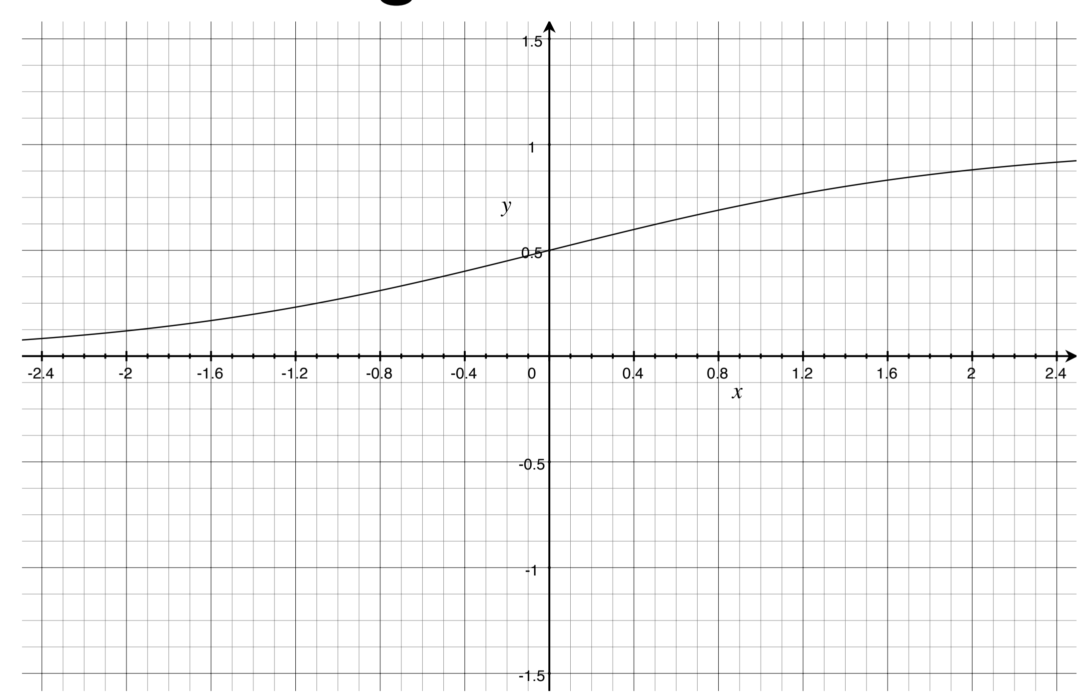


Perceptron (revision)

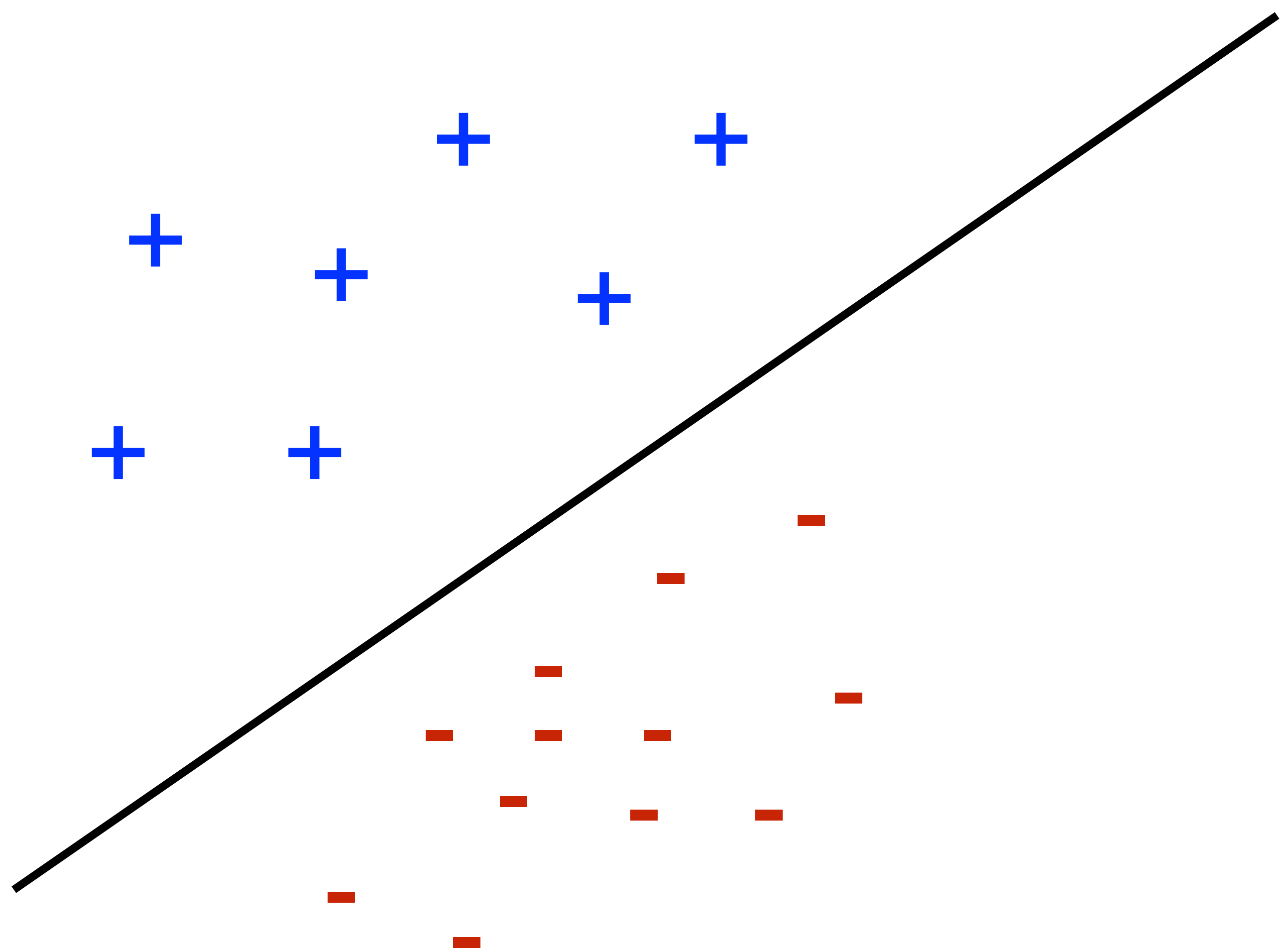



```
s = x1w1 + x2w2 + … + xnwn
if s > 0:
    return 1
else:
    return 0
```

Logistic function



linear separability

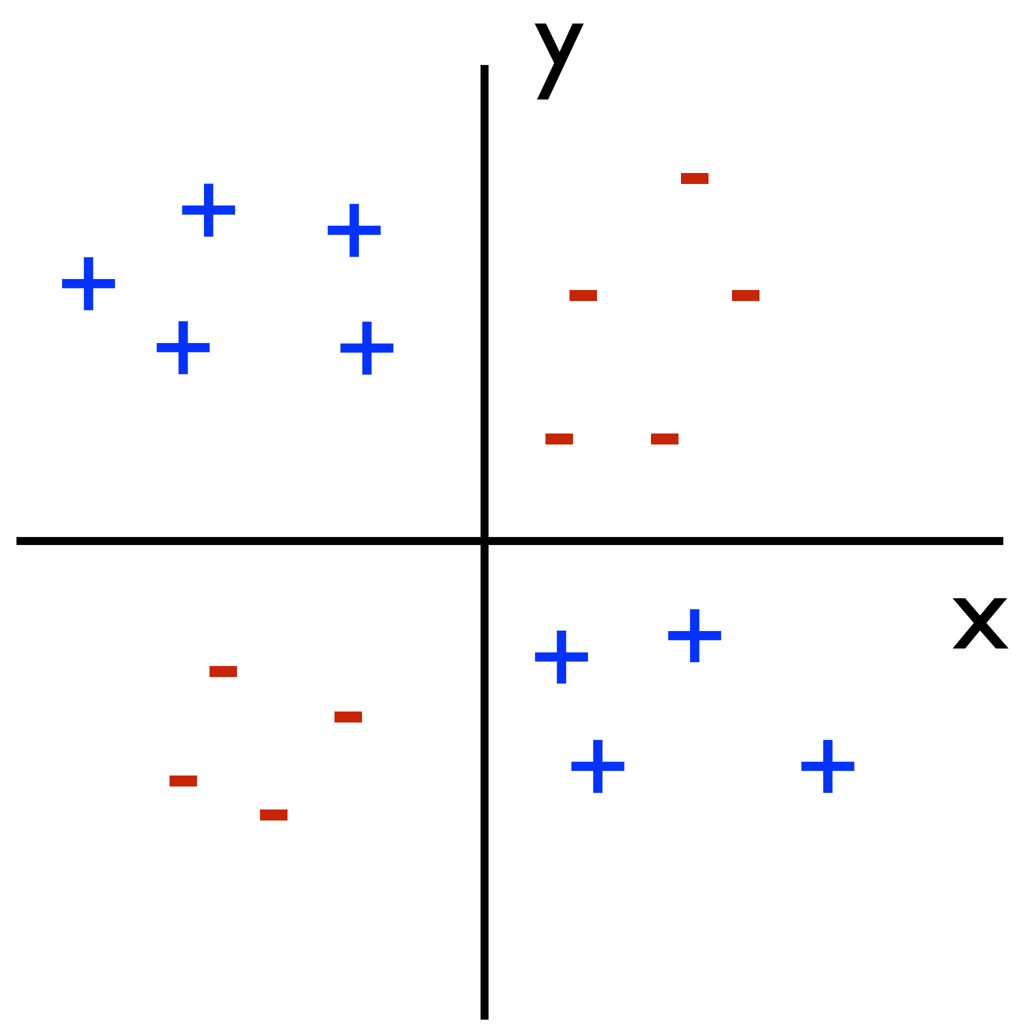


In 2D space linear separability means you can separate the two classes by a straight line. $ax + by + c = 0$

Non-linear separable case

XOR (exclusive OR)

	x=0	x=1
y=0	0	1
y=1	1	0



Multi-layer Neural Networks

- XOR is a very common logical operation
  - Perceptron being unable to handle this common case was seen as a show stopper for neural networks
- We could get over this issue by using multiple hidden layers.
  - But there was no algorithm to learn the weights
  - Until, error backpropagation (Rummelhart+86) was proposed
- However, deep neural networks are likely to overfit and training them was time consuming (no GPUs back then!)
  - Second NN winter.

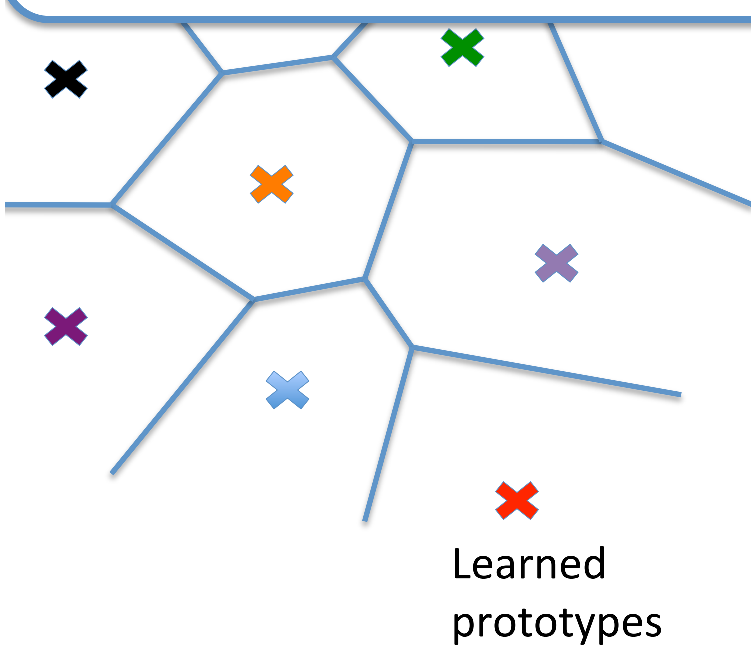
Advantages of DL

- Deep learning can learn the features useful for a particular task automatically
  - Can use **unlabeled** data to learn the features
- Can learn distributed representations

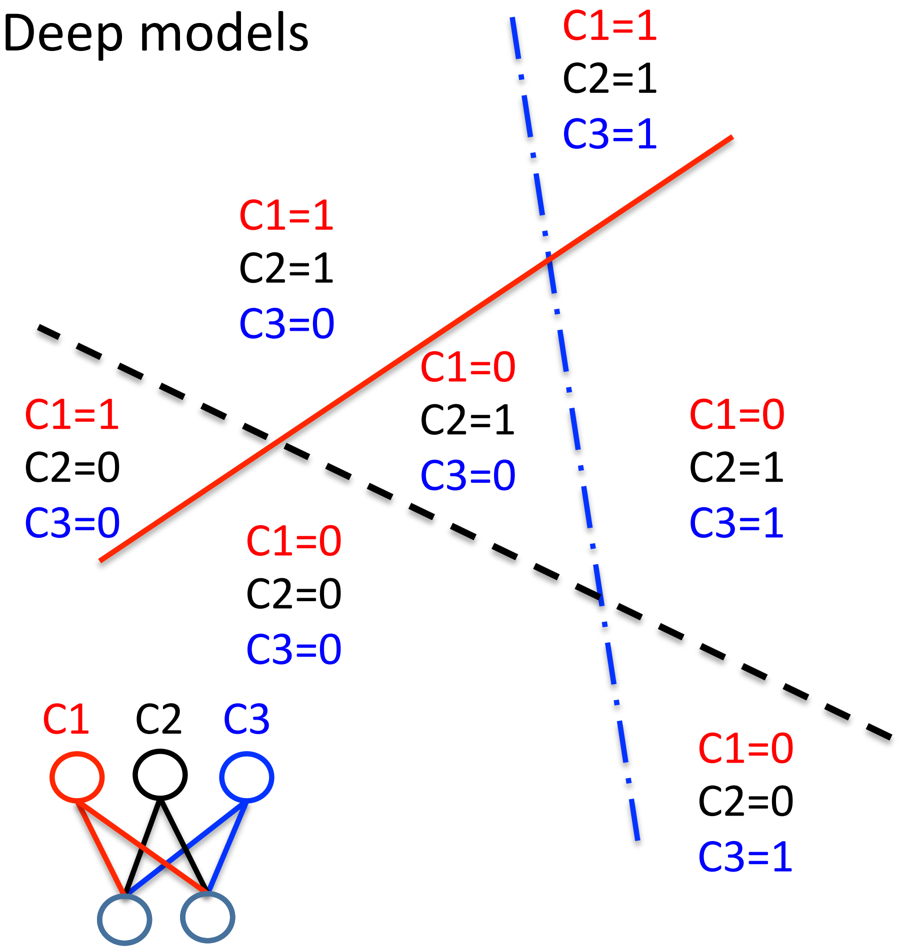
Local vs. Distributed Representations

- Clustering, Nearest Neighbors, RBF SVM, local density estimators

- Parameters for each region.
- # of regions is linear with # of parameters.



- RBMs, Factor models, PCA, Sparse Coding, Deep models

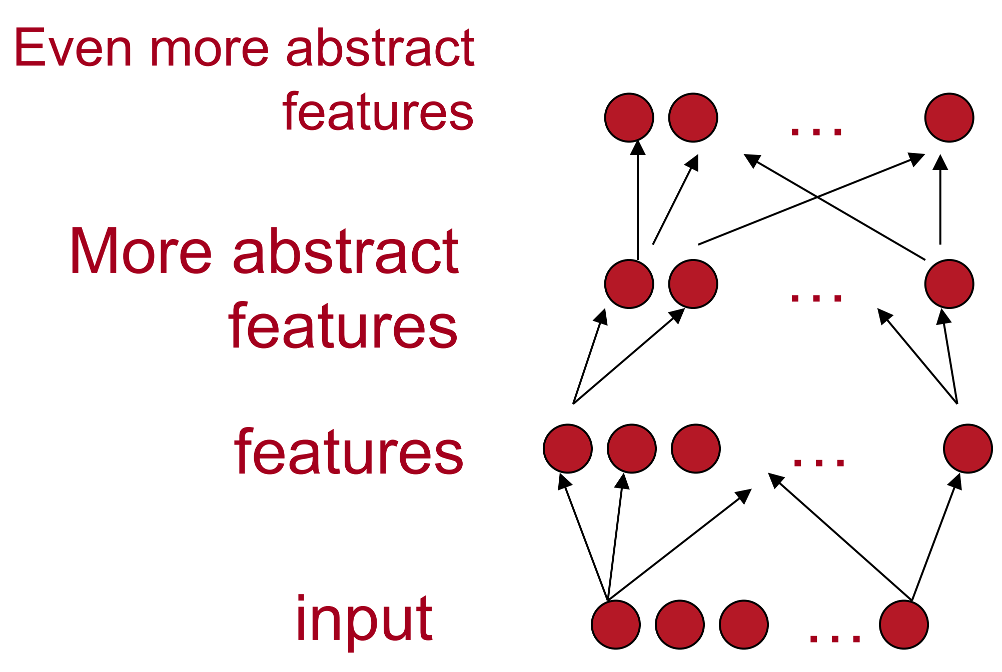


Bengio, 2009, Foundations and Trends in Machine Learning

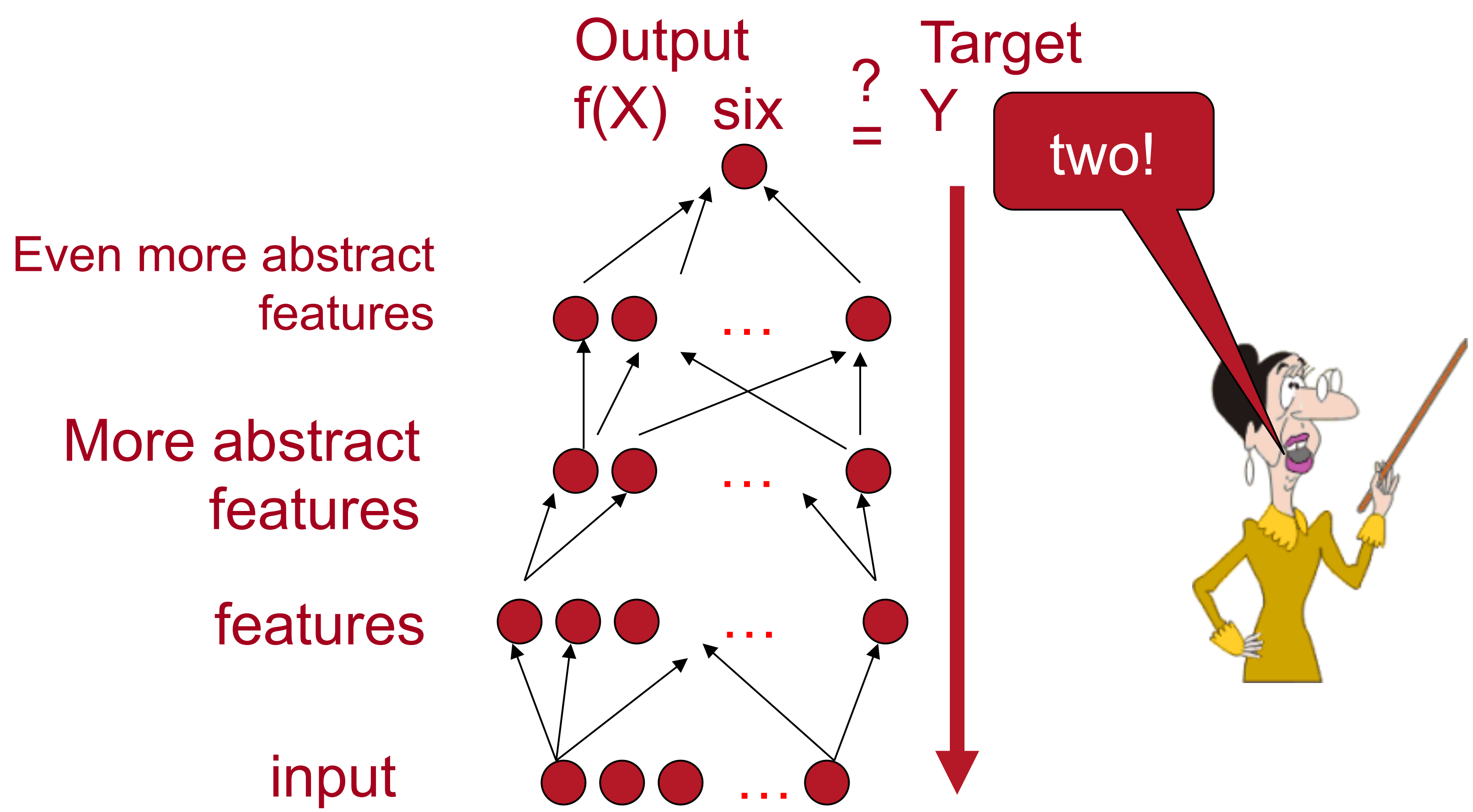
The breakthrough!

- It was shown that by learning two layers at a time, and then stacking those to create a deep neural network was an effective method for overfitting.
- greedy layer-wise training
    - *A Fast Learning Algorithm for Deep Belief Nets*, Hinton et al., Neural Computing, 2006.

unsupervised pre-training



supervised post-training

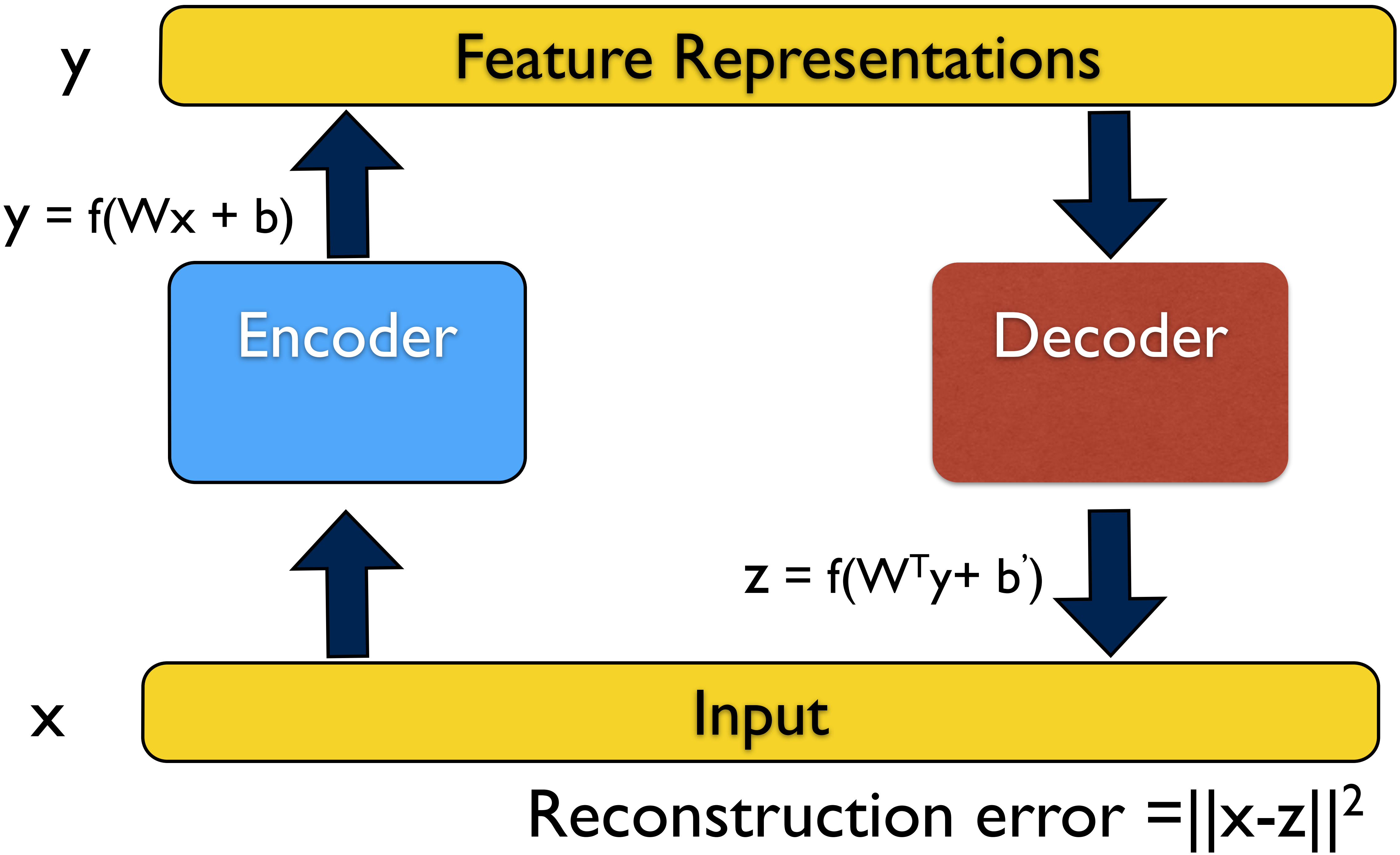


slide credit: Bengio KDD 2014

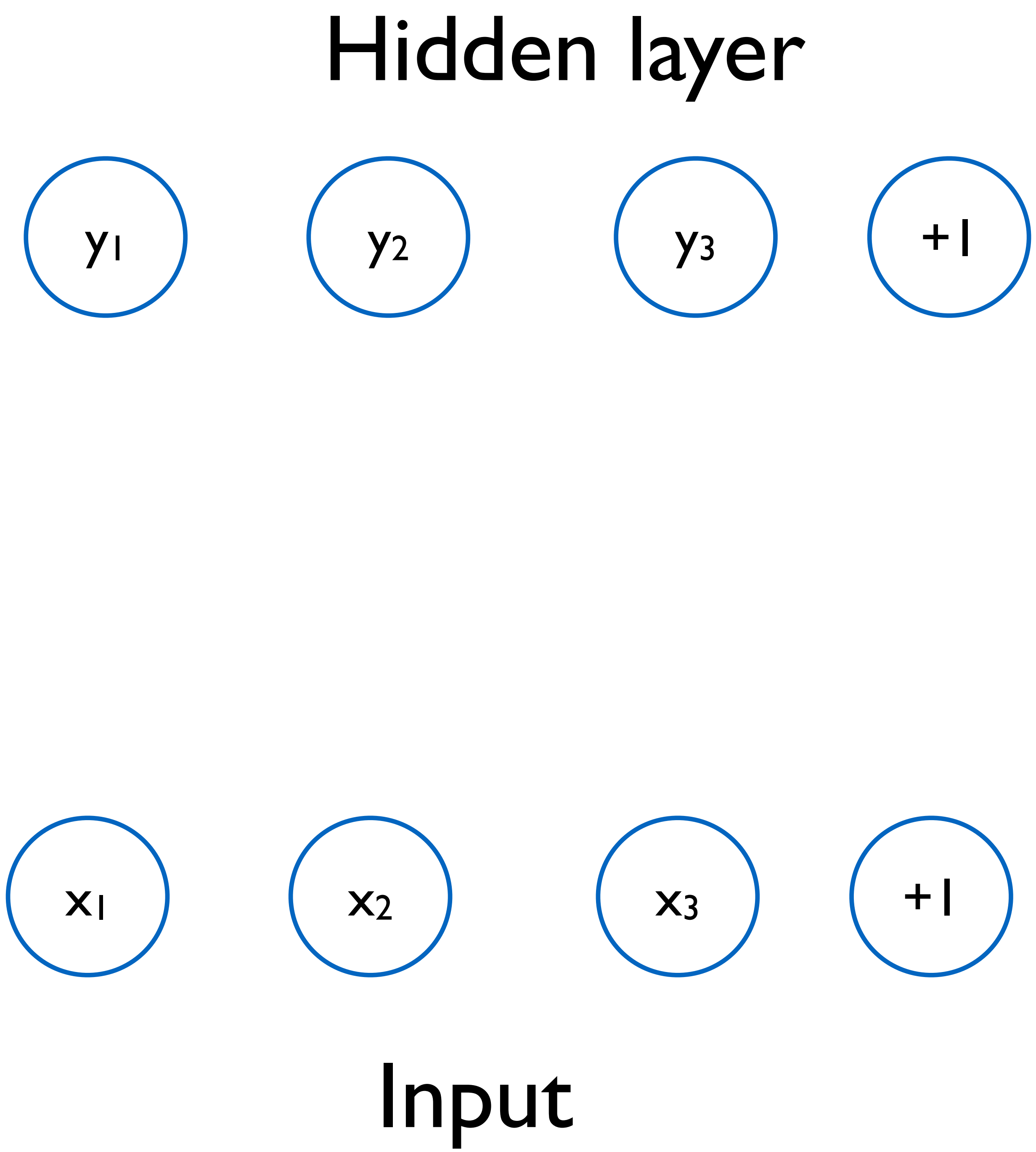
Deep Learning Methods

- Two main techniques exist
- Autoencoders (AE)
  - A non-probabilistic method
  - Easier to implement
  - Theoretical analysis is difficult (although some work has been done lately)
- Restricted Boltzman Machine (RBM)
  - A probabilistic method
  - Under certain conditions it could be shown that both RBMs and AEs are optimizing the same objective

Autoencoder

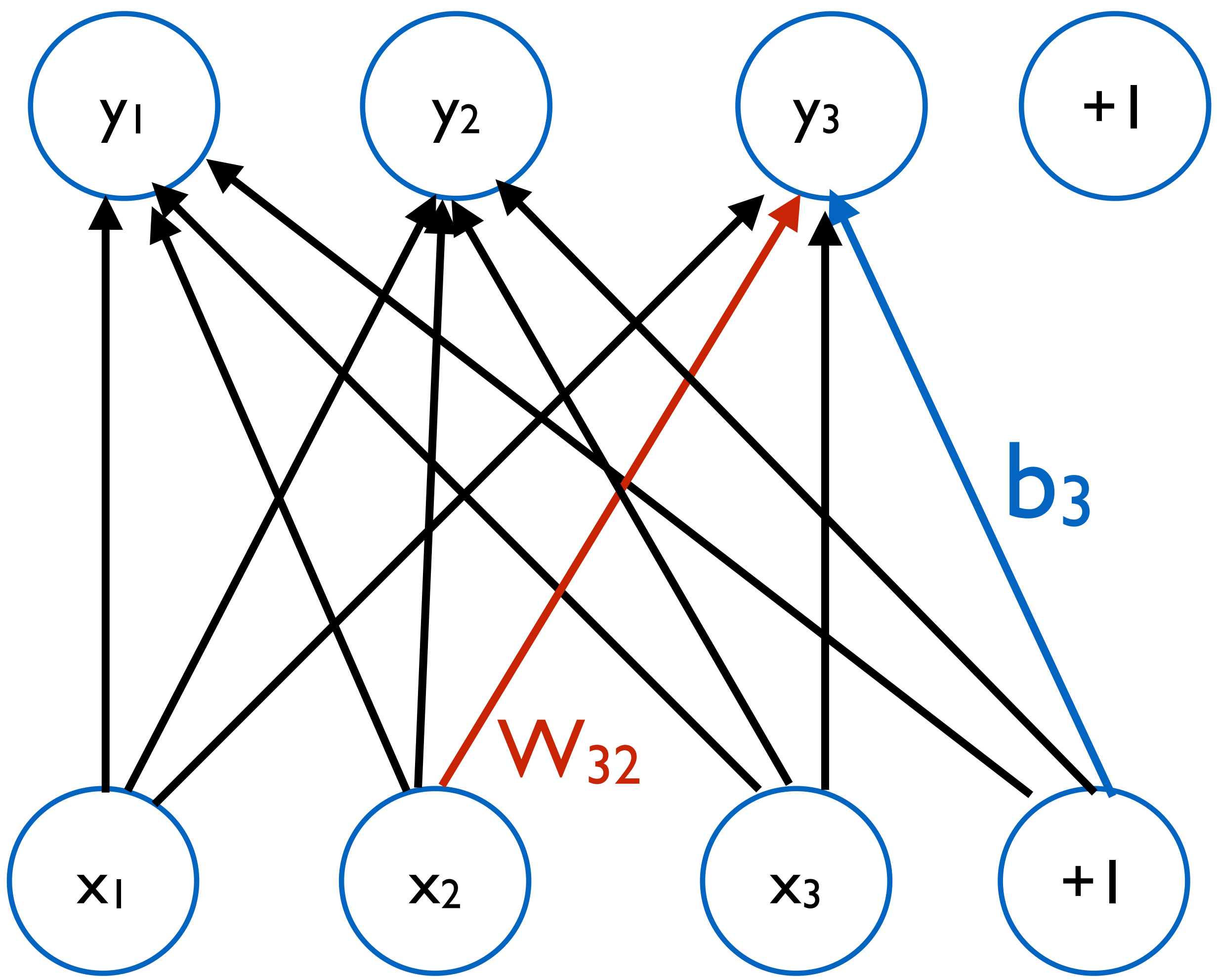


Autoencoder



Autoencoder

hidden layer



encoder

input layer

$w_{ij} : x_j \longrightarrow y_i$

Details

- By using the transpose matrix W' for the decoder where W is the encoder matrix, we can reduce the number of parameters in the model. (less likely to overfit)
- b and b' are respectively encoding and decoding bias terms.
- Bias terms can be incorporated as features into the autoencoder by setting a feature that is always ON.
- The non-linear function f is performing some elementwise non-linear operation on each element of a vector.
- Without non-linearity, autoencoders are equivalent to PCA.

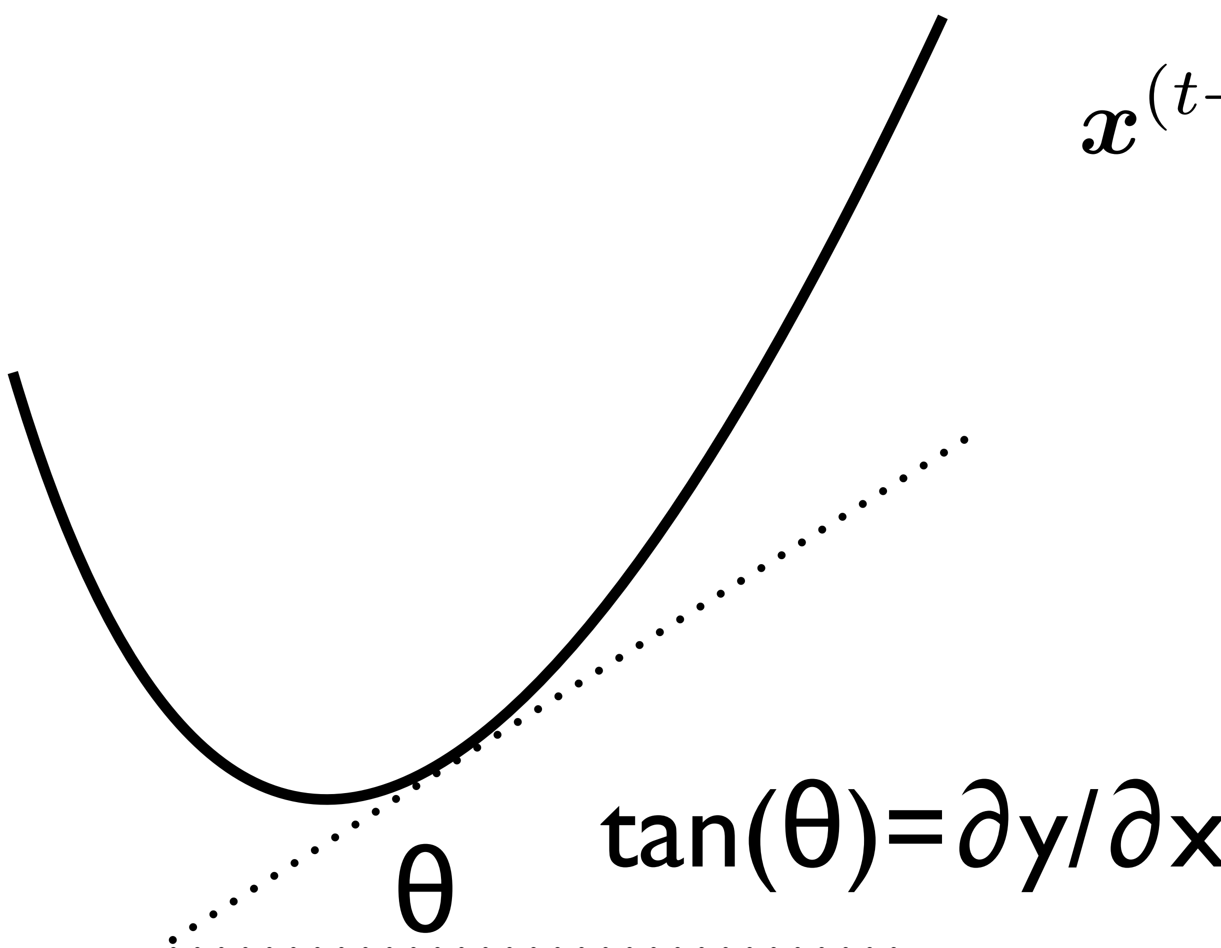
Training procedure

1. encode the input x using the encoder
   1. Calculate $Wx + b$, and insert it in f
2. Let the output of (1) be y. Insert y into the decoder and reconstruct the input
   1. Calculate $W^T y + b'$, and insert it in f
3. Let the output of (2) be z. Compare z and x.
   1. The loss function to be used is problem specific. For real values a popular loss function is the squared loss. For binary values use the cross-entropy error function.
4. Adjust the parameters (W, b, b') such that the loss computed in (3) is minimized.
   1. Compute the partial derivative of the loss w.r.t. each parameter and apply the stochastic gradient descent method.

Stochastic Gradient Descent (SGD)

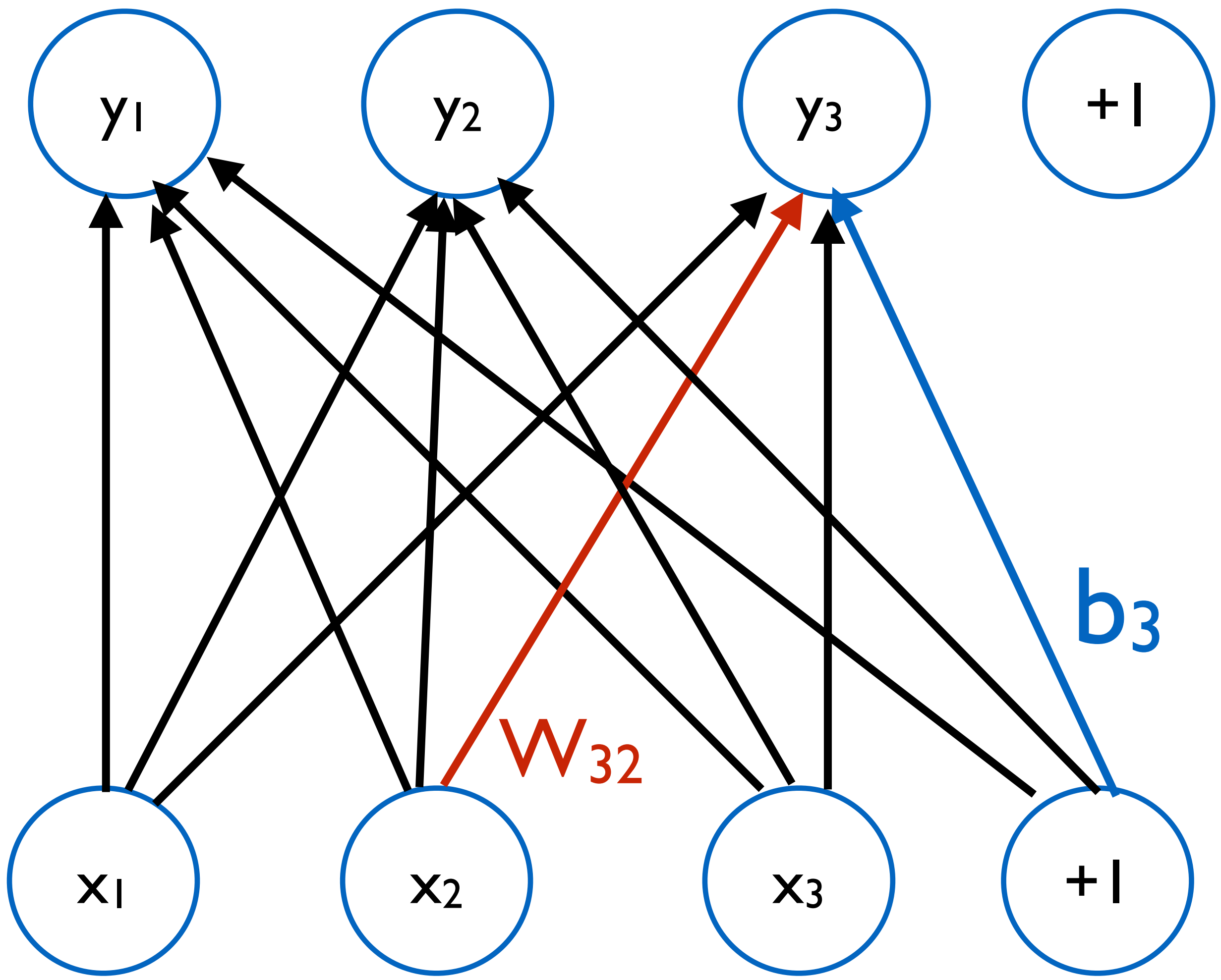
- We have already seen SGD in Perceptron, logistic regression and multi-layer neural networks.
- Move in the opposite direction of the gradient of the loss

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \frac{\partial E(\boldsymbol{x}, \boldsymbol{z})}{\partial x}\Big|_{t=t}$$



gradient is given by the derivative

Example



$$y_1 = f(x_1 W_{11} + x_2 W_{12} + x_3 W_{13} + b_1)$$
$$y_2 = f(x_1 W_{21} + x_2 W_{22} + x_3 W_{23} + b_2)$$
$$y_3 = f(x_1 W_{31} + x_2 W_{32} + x_3 W_{33} + b_3)$$

$$f(t) = \frac{1}{1 + \exp(-t)}$$

In matrix form…

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \qquad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} \qquad \boldsymbol{b}' = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \end{pmatrix}$$

$$\boldsymbol{y} = f(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

Decoder becomes…

$$z_1 = f(y_1 W_{11} + y_2 W_{21} + y_3 W_{31} + b'_1)$$
$$z_2 = f(y_1 W_{12} + y_2 W_{22} + y_3 W_{32} + b'_2)$$
$$z_3 = f(y_1 W_{13} + y_2 W_{23} + y_3 W_{33} + b'_3)$$

$$\boldsymbol{z} = f(\mathbf{W}^\top \boldsymbol{y} + \boldsymbol{b}')$$

squared loss $$||\boldsymbol{x} - \boldsymbol{z}||^2 = \sum_{k=1}^{d} (x_k - z_k)^2$$

Parameter update

Let us consider the update of w_{12}

$$\frac{\partial\,||\boldsymbol{x}-\boldsymbol{z}||^2}{\partial w_{12}} = -2(\boldsymbol{x}-\boldsymbol{z})\frac{\partial z_2}{\partial w_{12}}$$

$$\frac{\partial z_2}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} f(w_{12j}y_1 + w_{22}y_2 + w_{32}y_3 + b_2')$$

Here

$$t = w_{12j}y_1 + w_{22}y_2 + w_{32}y_3 + b_2'$$

$$\frac{\partial f(t)}{\partial w_{12}} = \frac{\partial f(t)}{\partial t}\frac{\partial t}{\partial w_{12}} = \frac{\exp(-t)}{(1+\exp(-t))^2}y_1 = \sigma(t)(1-\sigma(t))y_1$$

$$\boxed{\frac{\partial\,||\boldsymbol{x}-\boldsymbol{z}||^2}{\partial w_{12}} = -2(\boldsymbol{x}-\boldsymbol{z})\sigma(t)(1-\sigma(t))y_1}$$

$$w_{12}^{(n+1)} = w_{12}^{(n)} + 2\eta(\boldsymbol{x}-\boldsymbol{z})\sigma(t)(1-\sigma(t))y_1$$

References

- Deep Learning Tutorial
  - http://deeplearning.net/tutorial/gettingstarted.html
  - git clone git://github.com/lisa-lab/DeepLearningTutorials.git
- You need Theano
  - http://deeplearning.net/software/theano/install.html
- Dependencies Python >= 2.6, g++, python-dev, NumPy, SciPy, BLAS)
- Can be installed via sudo apt-get install in Debian/Ubuntu or for Mac OSX brew/macports.