Deep Learning

St. Louis Machine Learning & Data Science Meetup

Justin Brandenburg
Data Scientist

Dec 6, 2016

Machine Learning

Machine learning is the technique of pattern classification and prediction based on input data.

- What exactly is that input data?
- Can it use any data?
- Limitations?

Deep Learning

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using artificial neural network architectures composed of multiple non-linear transformations.

- “Deep" in “Deep Learning" means that the neural network has more than 2 layers
- “Deep" is also associated with the fact that the model makes use of unlabeled data
- “Deep" means that there is no need for human-invented features

An Explanation

Deep Learning is the process of applying artificial neural network technologies to solve problems

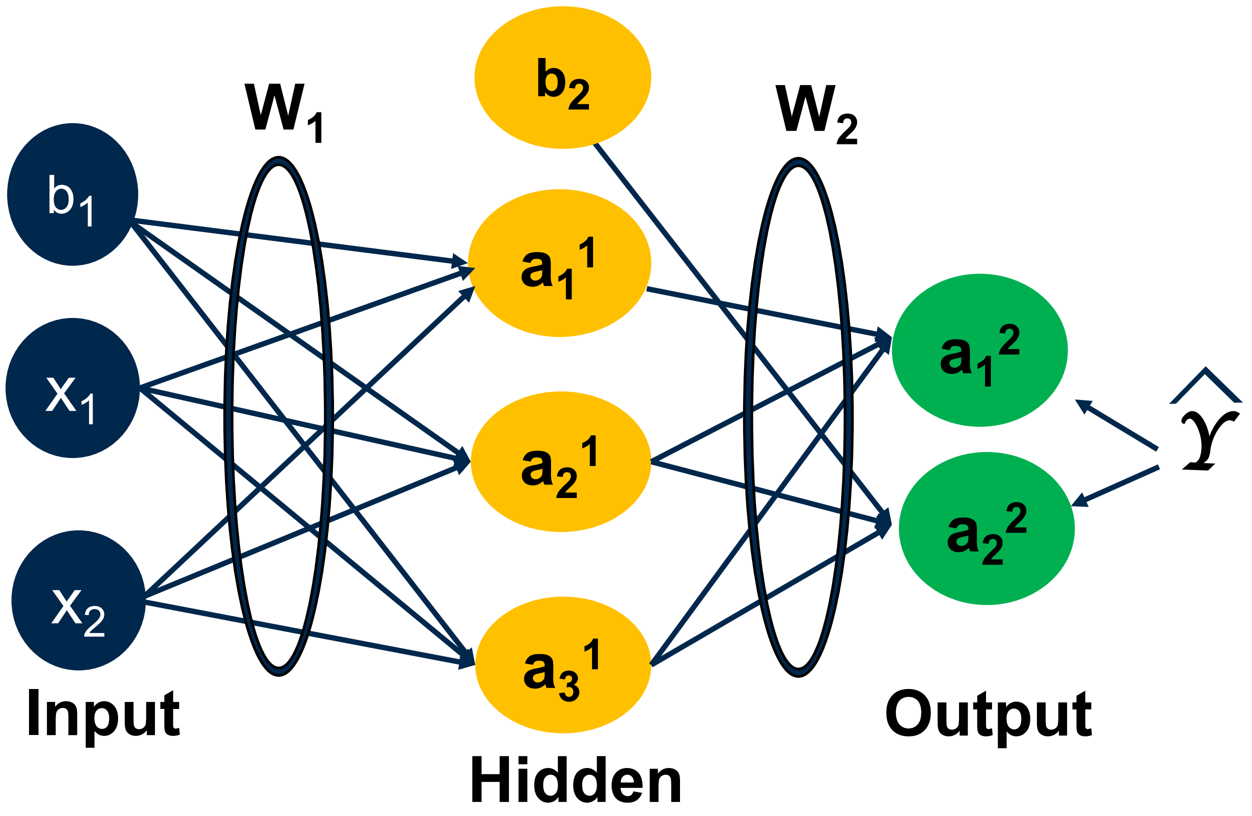
- A deep neural network is a neural network with multiple hidden layers trained sequentially with unsupervised learning techniques
- Attempts to learn prominent features from the input data and reduce the task of the task of building a feature extractor for every category of data.
- Able to model very complex nonlinear functions
- Well suited to highly dimensional problems, where the number of potential features is very large

Artificial Neural Networks

- Inspired by the study of brains and the nervous system
- Consist of a network of nodes (“neurons”) connected by directed graphs (“synapses”)
- **MULTILAYER PERCEPTRON:** a neural network architecture that has one or more hidden layers
  - Most widely used in predictive analytics
  - Are ***feedforward*** networks => a neuron in one layer can accept input from any neuron in a previous layer but cannot accept input from neurons in the same layer

Artificial Neural Networks

- 3-layer neural network with one input layer, one hidden layer, and one output layer.
- The number of nodes in the input layer is determined by the dimensionality of our data
- The number of nodes in the output layer is determined by the number of classes we have



x	y
(0.71, 0.91)	1
(0.25, 0.44)	0
(0.34, 0.61)	1

ANN-Activation Functions

- Activation functions for the hidden units are needed to introduce non-linearity into the network
- There are different types of units based on the activation functions
  - **Sigmoid**
  - **Rectified linear units (ReLU)**
  - **Hyperbolic tangent units (Tanh)**
  - **Softmax**

ANN - Forward Propagation

- Performing a dot-product on the inputs, X, with the weights, W_1 between the first and second layer

$$z_1 = X \bullet W_1 + b_1$$

- Apply an activation function to z_1

$$a_1 = \tanh(z_1)$$

- Performing a dot-product on the activation function output, a_1, and the next set of weights, W_2

$$z_2 = a_1 \bullet W_2 + b_2$$

- Apply an activation function to z_2 to get final output

$$a_2 = \hat{\boldsymbol{\gamma}} = \mathrm{softmax}(z_2)$$

From http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/

MAPR

ANN-Cost Function

- We need to determine how well our parameters (W1, W2, b1, b2) classified our output
- We use a cost function to measure our error:

Quadratic Cost (Squared Error Function)

$$C = L(\gamma, \widehat{\gamma}) = \sum \frac{1}{2} (\gamma - \widehat{\gamma})^2$$

Cross-Entropy

$$C = L(\gamma, \widehat{\gamma}) = -\frac{1}{n} \sum [\gamma \ln a + (1 - \gamma) \ln(1 - a)]$$

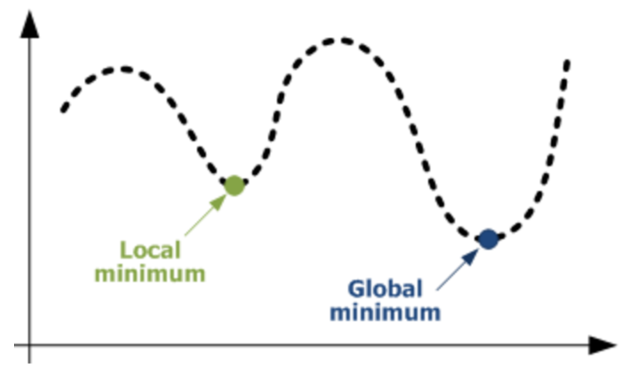
ANN-Training & Optimizing

- To train our NN and improve our estimates, we need to minimize our errors
- Use **Backpropagation** to compute the gradients (vector of derivatives) of the layer parameters with respect to the prediction error

$$\frac{\partial C}{\partial W_1}, \frac{\partial C}{\partial W_2}, \frac{\partial C}{\partial b_1}, \frac{\partial C}{\partial b_2}$$

- Goal is to find the global minimum
- Use **Gradient Descent** (Batch)

Non-convex



ANN-Backpropagation

- BP algorithm iterates through many cycles (**epochs**) of two processes.
- First process is the feed forward
- Second Process:
  - The difference between the network's output signal and the true value results in an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors.
  - Use the derivative of each neuron's activation function to identify the gradient in the direction of each of the incoming weights
  - The gradient suggests how steeply the error will be reduced or increased for a change in the weight.
  - The algorithm will attempt to change the weights that result in the greatest reduction in error by an amount known as the learning rate.

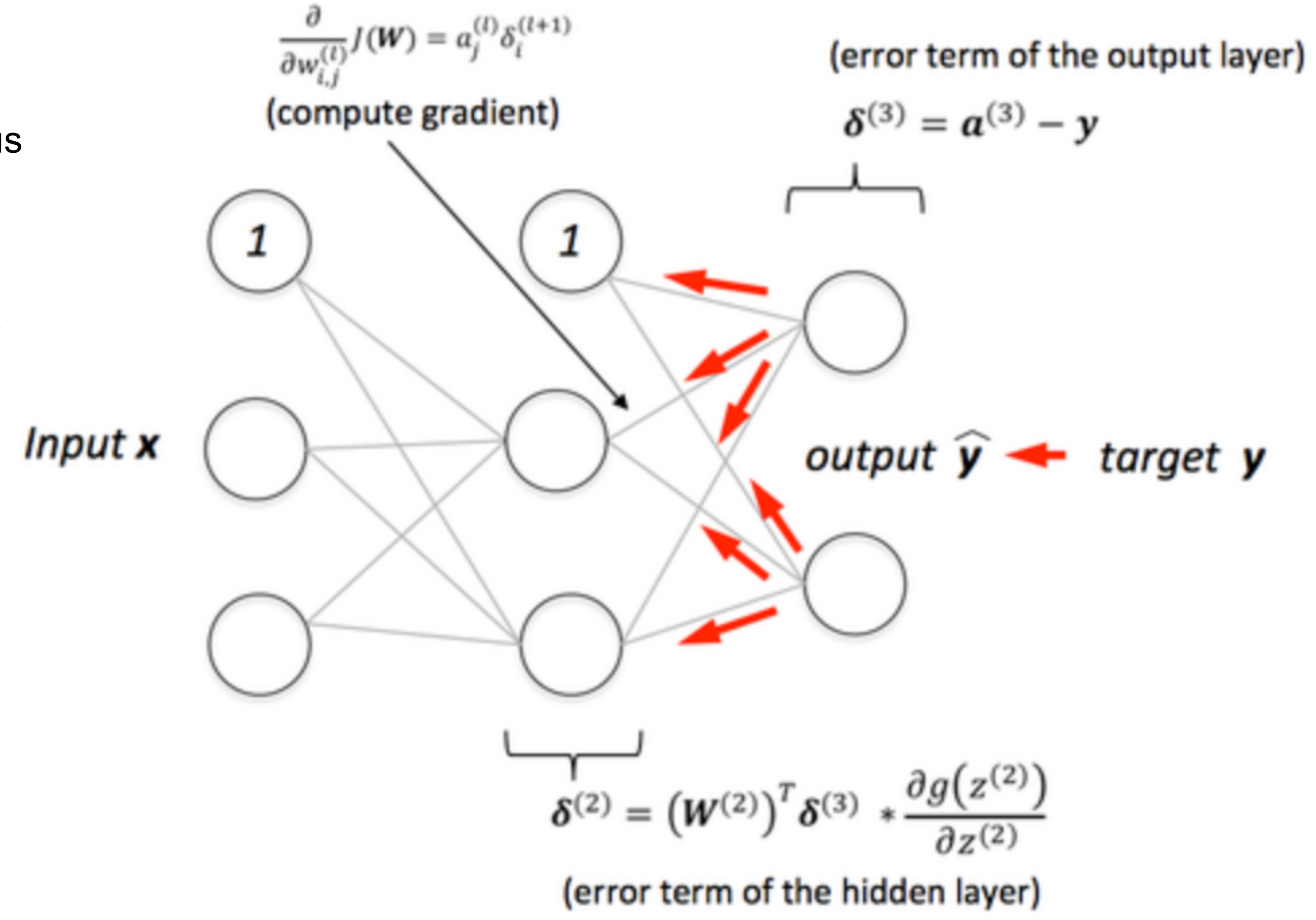
ANN-Backpropagation

Strengths:

- Adaptable to classification and continuous outcomes
- Capable of modeling more complex patterns than nearly any algorithm
- Makes few assumptions about the data's underlying relationships

Weaknesses:

- Extremely computationally intensive and slow to train
- Prone to overfitting training data
- Black box model, difficult to interpret


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

Visual from http://sebastianraschka.com/faq/docs/visual-backpropagation.html

Considerations in ANN

- We can choose the dimensionality (the number of nodes) of the hidden layer
- The more nodes we put into the hidden layer the more complex functions we will be able fit
- More computation is required to make predictions and learn the network parameters
- Bigger number of parameters could mean overfitting our data

ANN to Deep Learning

- We move from neural networks to Deep Learning when we start working with multiple hidden layers (more than 2)
- Deep neural network can fit it, more accurately, with fewer parameters than a neural network
- Two most popular Deep Neural Nets:

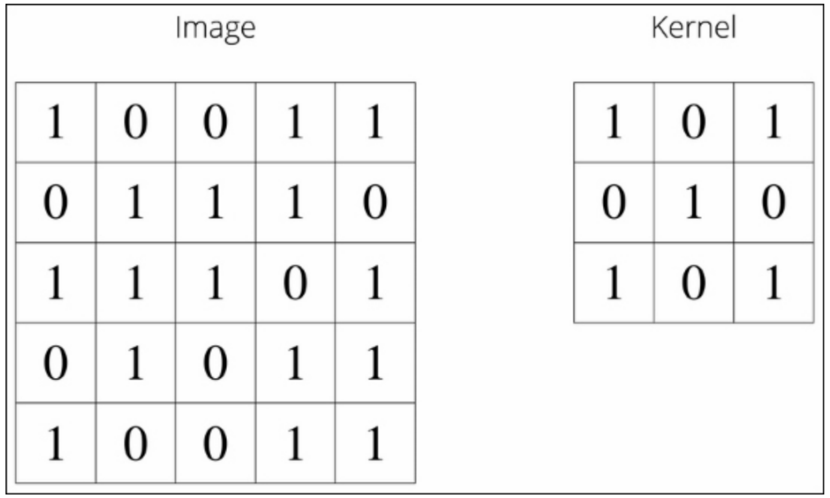
  1. **Convolutional Neural Networks (CNNs)**
  2. **Recurrent Neural Networks (RNNs)**

Convolutional Neural Network

- **Convolutional Neural Network (CNN)**
  - Feedforward network that has proven extremely successful for image analysis
  - Pushing the field of computer vision with focus on object detection
- Four main steps in CNNs:
  1. **Convolution** - applying a filter (kernel, a matrix) over an image (Height (pixels) x Width (pixels) x Channels (RGB = 3)) and computing the dot product to generate a feature map
  2. **Non Linearity** – apply activation function (ReLU) to feature map
  3. **Pooling or Sub Sampling** – reduces the dimensionality of each feature map but retains the most important information
     - When pooling over a map, the output will stay the same if you shift image by a few pixels
  4. **Classification** (Fully Connected Layer) – traditional ANN that utilizes an activation function (Softmax) for classification
     - Utilizes the backpropagation discussed earlier for training
     - The loss for each kernel is calculated, enabling the individual weight adjustment of every kernel as needed

Convolutional Neural Network

Convolution

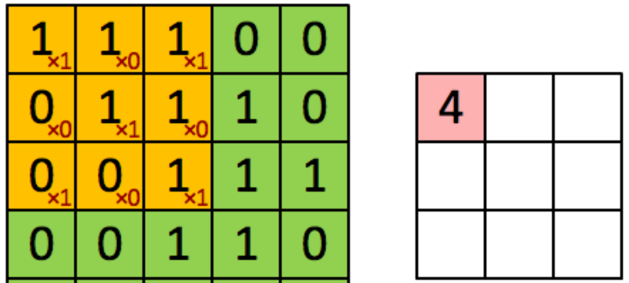


Image

1	0	0	1	1
0	1	1	1	0
1	1	1	0	1
0	1	0	1	1
1	0	0	1	1

Kernel

1	0	1
0	1	0
1	0	1



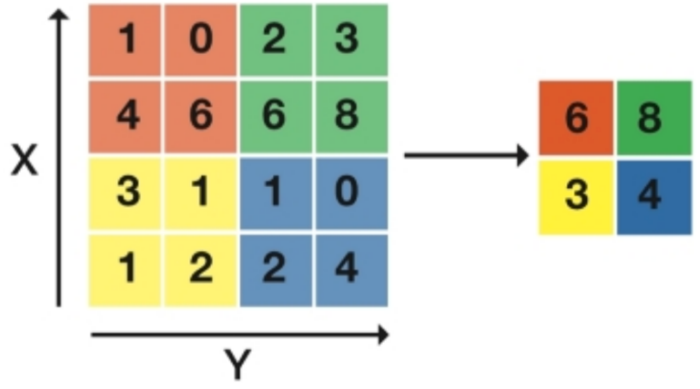
$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

4		

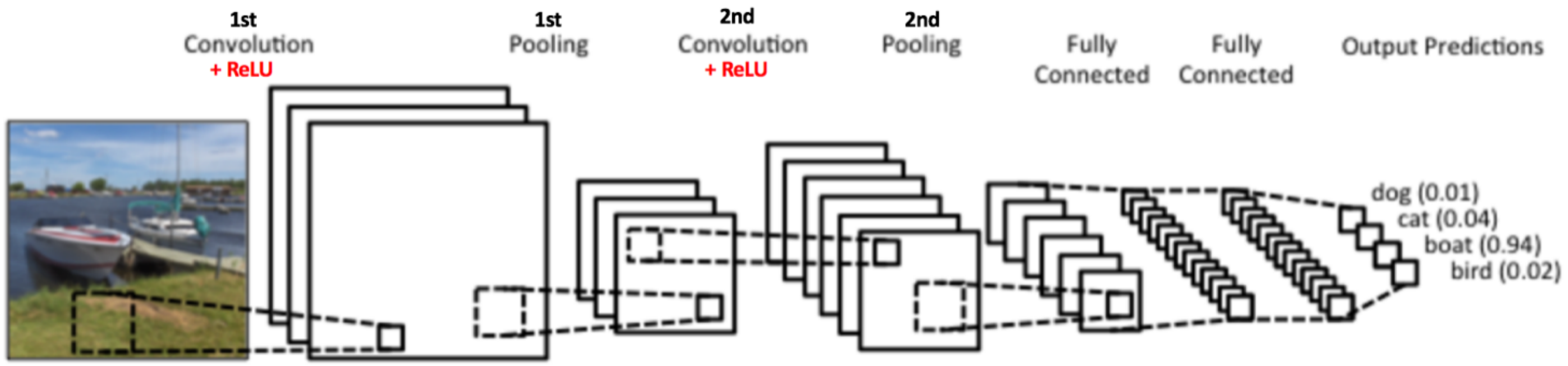
Image

Convolved Feature

Max-Pooling



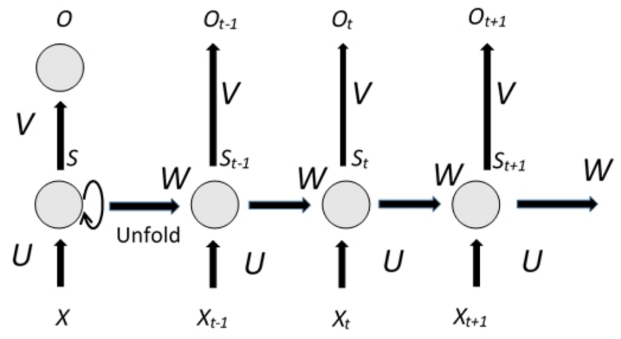
Full CNN Process



Convolution with 3×3 Filter. Source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

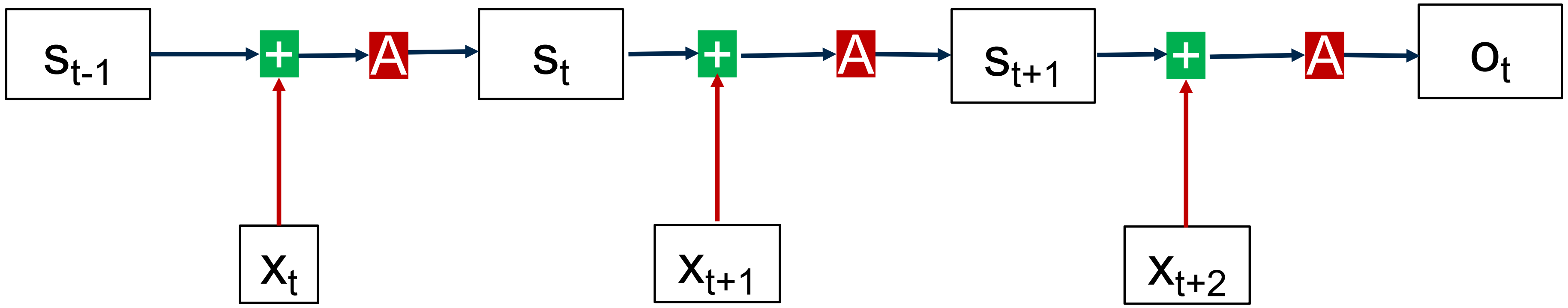
Recurrent Neural Network

- **Recurrent Neural Network (RNN)**
  - Assumes output is dependent on input and past information has an impact on current and future results
    - Great for Time Series Forecasting, Language Modeling, Sentiment Classification, Machine Translation
  - Used with the assumption that structure is important
    - Ex: **NATIONALS** DEFEAT **CARDINALS** and not **CARDINALS** DEFEAT **NATIONALS**
  - RNN always uses the same parameters which improves the training time
  - Uses a form of backpropagation for training



Recurrent Neural Network

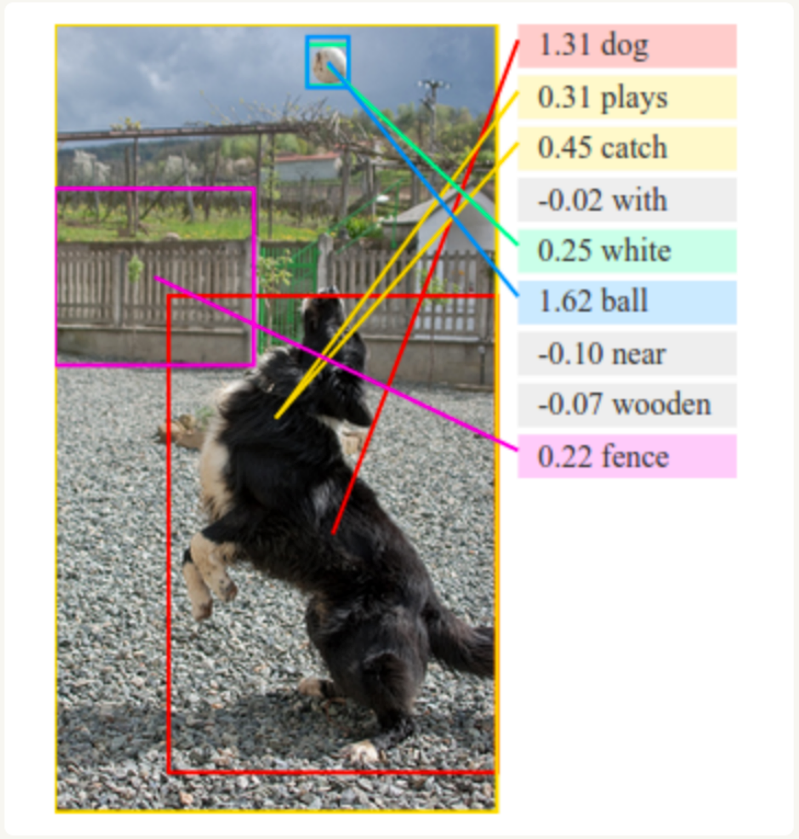
- x_t is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence.
- s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step
- A usually is a nonlinearity such as tanh or ReLU. $s_{\{-1\}}$, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- o_t is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary

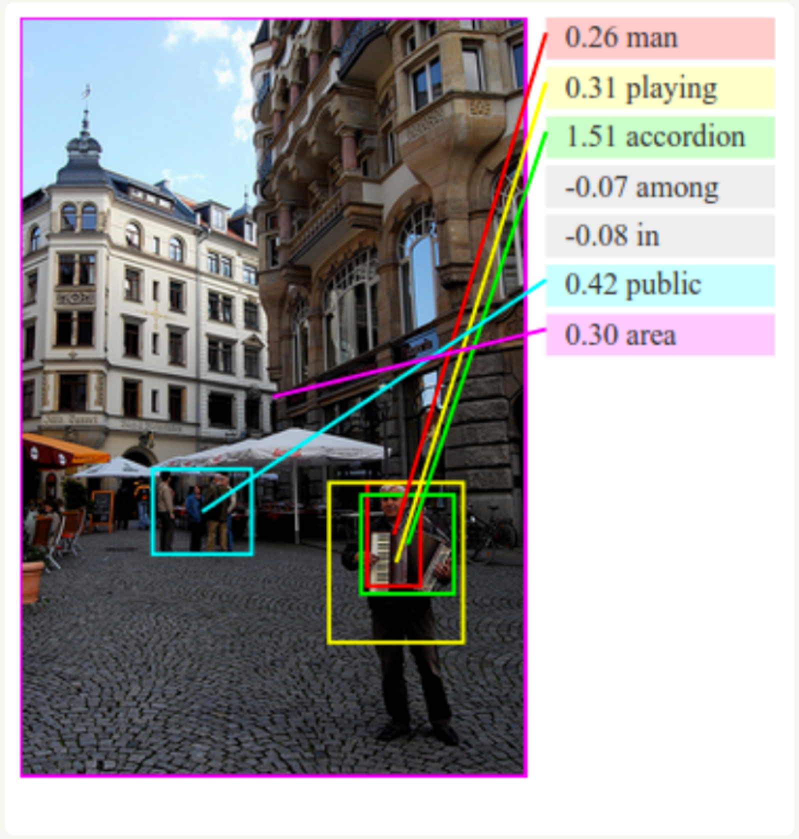


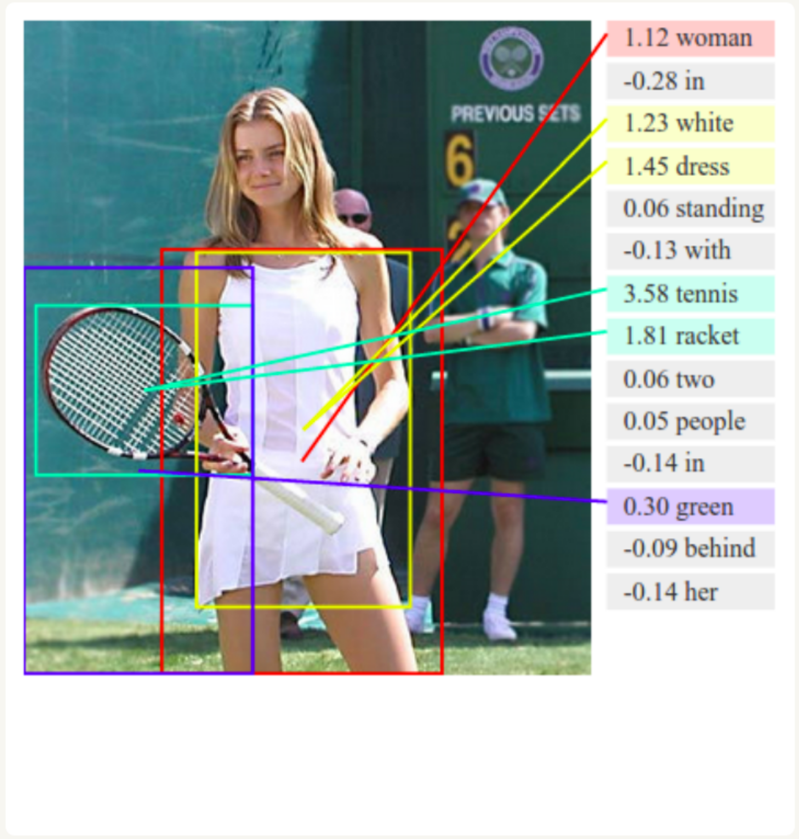
Recurrent Neural Network

- Vanishing/Exploding Gradient Problem:
  - RNN tend to be very deep
  - If the gradient descends too quickly before all inputs are entered (for example, a long text file)
  - Vanishing - occurs when you use a Sigmoid or Tanh function,
  - Exploding - ReLU
- Long Short-Term Memory (LSTM) RNN
  - Employs gating units that prevent Vanishing and Exploding Gradients
  - Gates act on the signals they receive and they block or pass on information based on its strength and import, which they filter with their own sets of weights
  - Effective in Time Series Analysis
    - Forecasting
    - Anomaly Detection

CNN & RNN







Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

GPU vs CPU

- The CPU is good at fetching small amounts of memory quickly while the GPU is good at fetching large amounts of memory
- A single GPU might have thousands of cores while a CPU usually has no more than 12 cores. Although GPU cores are slower than CPU cores, they more than make up for that with their large number and faster memory if the operations can be parallelized
- GPUs were designed to handle such vector and matrix operations in parallel unlike a single core CPU that would handle matrix operations in serial form processing one element at a time
- On a CPU, Caffe showed the best parallelization results and TensorFlow also can exploit the capabilities offered by multiple threads.

Deep Learning vs SVM vs Random Forest

- SVMs and Random Forest are easier to train and easier to understand
- SVMs are less prone to over fitting your data
- Random Forest can handle outliers and multi class outcomes
- SVMs can be multiclass but often you still have to train for each class
- Random Forest complexity grows with the number of trees in the forest
- SVMs require a good amount of hyper-parameter tuning and it becomes computationally expensive with you have many classes

Deep Learning Libraries

- There are many good and tested libraries for deep learning in popular programming languages:

Caffe
Chainer
DL4J
Deeplearning4j
H_2O

MINERVA
mxnet
Purine

K
KERAS
Microsoft
CNTK
MatConvNet

TensorFlow
theano
torch

Wrapping Up

- The goal behind deep learning is to automatically learn the features from data
  - Algorithms that do the feature engineering for us to provide deep neural network structures with meaningful information so that it can learn more effectively.
- When choosing to implement a Deep Learning Framework
  - Expected output (classification, feature extractor)
  - What is your hardware and software
    - CNN and RNN are computationally expensive
  - Start with simplest methods first
    - If logistic regression works, use it
  - Do you have enough data?
  - How does this affect your workflow and data pipeline?

Questions?

MapR Technologies

Justin Brandenburg
jbrandenburg@mapr.com

MapR Blog
https://www.mapr.com/blog/

MapR Academy
http://learn.mapr.com/