Course 395: Machine Learning - Lectures

Lecture 1-2: Concept Learning (M. Pantic)

Lecture 3-4: Decision Trees & CBC Intro (M. Pantic & S. Petridis)

Lecture 5-6: Evaluating Hypotheses (S. Petridis)

Lecture 7-8: Artificial Neural Networks I (S. Petridis)

Lecture 9-10: Artificial Neural Networks II (S. Petridis)

Lecture 11-12: Artificial Neural Networks III (S. Petridis)

Lecture 13-14: Genetic Algorithms (M. Pantic)
**Dropout**

- We don’t modify the error function but the network itself
- During training neurons are randomly dropped out
- The probability that a neuron is present is $p$

From Dropout: A simple way to prevent neural networks from overfitting by Srivastava et al., JMLR 2014
**Dropout**

- Dropout prevents overfitting because it prevents neurons from co-adapting too much. Each neuron should create useful features on its own without relying on other hidden units to correct its mistakes.

- Typical values for $p$: 0.8/0.5 for input/hidden neurons.
- Test time: outgoing weights of a neuron are multiplied by $p$.

From Dropout: A simple way to prevent neural networks from overfitting by Srivastava et al., JMLR 2014
Dropout - Tips

• If a network with n neurons in the hidden layer works well for a given task then a good dropout network should have n/p neurons.

• Dropout introduces a significant amount of noise in the gradients, a lot of gradients cancel each other → you should use higher learning rate (and maybe higher momentum)

• More epochs are needed

• The above heuristics do not always work!
Data Augmentation

• One of the best ways to avoid overfitting is more data

• So we can artificially generate more data, usually a bit noisy, so we introduce more variation

• We should apply operations that correspond to real-world variations.

• For images: flip left-right, rotate, random cropping, etc
Data Normalisation

- It is not desirable that some inputs/features are orders of magnitude larger than other inputs. Why?

- Map each input/feature to \([-1/0, +1]\)

- Min value is mapped to -1/0

- Max value is mapped to 1
Data Normalisation

- Standardize inputs to mean=0 and 1 std. dev.=1
  \[ y = \frac{x - x_{\text{mean}}}{x_{\text{std}}} \]

- Useful for continuous inputs/targets

- It’s called z-normalisation

- Scaling is needed if inputs take very different values. If e.g., they are in the range [-3, 3] then scaling is probably not needed
Data Normalisation

- $x_{\text{mean}}, x_{\text{std}}$ are computed on the training set and then applied to the validation and test sets.

- It is not correct to normalise each set separately.
Image Normalisation

- When the input data are images then you can simply remove the mean image computed on the training set.

- Alternatively, you can compute the mean and standard deviation of all the pixels in each image and z-normalise each image independently.

- In case of videos, it’s usually better to apply the same normalisation to all frames in the video.
Monitoring the learning process

- If loss increases or oscillates then the learning rate is too high
- If loss goes down slowly the learning rate is low

Find a learning rate value at which the loss on the training data immediately begins to decrease.

It’s a good idea to turn off regularisation at this point

Monitoring the learning process

Other tips

• Compute the mean and standard deviation of hidden neurons activations for all examples in a mini-batch

• They should be different than 0 (this is important when ReLu is used since the neurons can easily die)

• For each layer compute the norm of the weights and the norm of the weight updates Δw.

• The ratio norm(Δw) / norm(w) should be 0.01 – 0.0001

• If ratio is significantly different then something could be wrong
Hyperparameter Optimisation

- Once a good initial learning rate value is found then we can optimise the hyperparameters on the validation set.
- Network architecture: number of layers, number of neurons per layer.
- Learning rate: when to start decaying, type of decay.
- Regularisation: type of regularisation, values for regularisation parameters.
- Training algorithm, SGD+Momentum, Adam, RMSprop.
- Maybe we wish to optimise again the initial learning rate.
(Hyper)Parameters / Weights

• (Hyper)Parameters are what the user specifies, e.g. number of hidden neurons, learning rate, number of epochs etc

• They need to be optimised

• Weights: They are also parameters but they are optimised automatically via gradient descent
Deep NNs

3-layer feed-forward network

4-layer feed-forward network

- Two ways to train
- A lot of data (data augmentation), ReLu, dropout etc
- Pre-training: weights are initialised to a good starting point
  - Restricted Boltzmann Machines or Stacked Denoising Autoencoders
  - Backpropagation is used to fine-tune the weights
Convolutional Neural Networks

• Convolutional Neural Networks (CNNs) have been very successful in computer vision


• Improved by LeCun et al., “Gradient-Based Learning Applied to Document Recognition”, Proc. IEEE, 1998
Convolutional Neural Networks

- Became popular in 2012 after winning the ImageNet competition

- “ImageNet Classification with Deep Convolutional Neural Networks”, by Krizhevsky et al., NIPS 2012

- Tricks: Data augmentation, Dropout, ReLu + GPUs
ImageNet Competition – Object Classification

- Classification of 1000+ objects
- State-of-the-art before 2012: ~26%
- New state-of-the-art in 2012 with deep networks: ~15%
Convolutional Neural Networks

- It’s a deep network = many layers
- Each layer is either a convolutional layer or subsampling layer
- Final layers are fully connected layers
Convolutional Neural Networks

- **Convolution**

  From: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

- **Max Pooling**

  From: http://cs231n.github.io/convolutional-networks/#pool
Convolutional Neural Networks

From: Peeman et. al, Speed sign detection and recognition by convolutional neural networks

From: Taigman et. al, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR 2014
Convolution Types

- Images: 2D convolution

- Videos: we can use 2D convolutions on each image

- We can also stack together a few frames (e.g., 3 – 5) and use a 3D convolution

- Audio signal: 1D convolution
CNN Architectures

- AlexNet
- VGG16, VGG19
- Inception
- ResNet 18, 34, 50, 101, 152
- DensetNet 121, 161, 169, 201
Residual Networks (ResNet)

- Deep CNN with shortcut connections
- Makes easier training of deeper networks.
- Variations: DenseNet, Wide ResNet

From: He et al., Deep Residual Learning for Image Recognition, CVPR 2016
Deep Networks for Time Series

- Deep feedforward NNs/CNNs are good at various tasks but not at handling time series data
- Recurrent Neural Networks are suitable for time series
- They also suffer from the vanishing gradient problem
LSTMs

• A type of recurrent network that can be effectively trained is the Long-Short Term Memory Recurrent Neural Network (LSTM-RNN). Introduced in 1990s

• We replace the neuron with a memory cell

• There are input, output and forget gates which control when information flows in / out of the cell and when to reset the state of the cell
LSTMs

From LSTM: A search space odyssey by Greff et al., arXiv Mar 2015
CNNs vs LSTMs

- CNNs are good at extracting features from raw data (images, audio waveform etc) but they do not model temporal dynamics.

- LSTMs are good at modelling time series but they do not extract features.

- We can add a softmax layer to turn them into classifiers.

- If we jointly want to extract features, model temporal dynamics, and perform classification (video classification, speech recognition) then we can combine CNNs + LSTMs + softmax.
End-to-end Learning

- This is called end-to-end learning because the input is raw data (one end) and the output is the classification label (other end).

- We do not intervene at feature extraction or classification, the deep network learns to model the pipeline from one end to the other end.
Data Types

Image:
- 2D spatial data
- 2D CNNs
- No temporal information

Video:
- 3D spatiotemporal data
- 2D images in time
- We can use 2D/3D CNNs + LSTM
- Frame rate: 25/30 frames per second

Audio:
- 1D temporal data
- There is no spatial information
- 1D CNN + LSTM
- Sampling rate: 44.1 kHz
Data Types

- In images/videos we extract features per frame/group of frames.

- In audio there is no notion of frame, so we define a window with length $K$ (e.g., 40) ms as our frame.

- We also use overlapping frames with stride = 10ms.
Data Types

- Above values for window length and stride are usually used with traditional features.

- When CNNs are used different window size/stride might be used, e.g., 5ms and 0.25ms.

- Note that frame rate of audio and video are different!!

Traditional Features: Mel Frequency Cepstral Coefficients (MFCCs)

CNN Features: 1D CNNs
Image/Video

- ResNet extract features directly from the images.
- BLSTMs model temporal dynamics in each stream.
- Such architectures significantly outperform traditional approaches (feature extraction + classification).
- You can come up with several variants of this architecture.
Audio

- ResNet extract features directly from the raw waveform.
- BLSTMs model temporal dynamics in each stream.
- Use of 1D CNNs is still an active research area.
- Usually MFCCs + BLSTMs work equally well.
End-to-end Audiovisual Fusion

- ResNets extract features directly from the images and the audio waveform, respectively.

- BLSTMs model temporal dynamics in each stream.

- Top BLSTMs perform fusion and model joint temporal dynamics.
Traditional Audiovisual Fusion


- Handcrafted audio/visual features are extracted then fusion takes place, e.g., by feature concatenation
- Similar approach was also used for emotion recognition
End-to-end Audiovisual Fusion - Training

- It’s impossible to train the entire architecture from scratch.
- We first train each stream (audio/visual) independently in 2 steps.
End-to-end Audiovisual Fusion - Training

First train the ResNet

Fix ResNet, Train BLSTMs

Fine-tune entire stream

First train the ResNet

Fix ResNet, Train BLSTMs

Fine-tune entire stream
End-to-end Audiovisual Fusion - Training

- Fix audio/visual streams and train top BLSTM layers only.
- Fine-tune the entire network.
End-to-end Audiovisual Fusion - Results

<table>
<thead>
<tr>
<th>Stream</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (End-to-End)</td>
<td>97.7</td>
</tr>
<tr>
<td>A (MFCC)</td>
<td>97.7</td>
</tr>
<tr>
<td>V (End-to-End)</td>
<td>83.0</td>
</tr>
<tr>
<td>V [15]</td>
<td>76.2</td>
</tr>
<tr>
<td>V [19]</td>
<td>61.1</td>
</tr>
<tr>
<td>A + V (End-to-End)</td>
<td>98.0</td>
</tr>
</tbody>
</table>

- Results on audiovisual speech recognition, goal is to recognize 500 words (data from BBC TV).
- AV model results in small improvement in clean audio conditions and significant improvement in noisy audio conditions.