ESE 546, FALL 2020

MODULE 1 SUMMARY

SHEIL SARDA [SHEIL@SEAS],

Lecture outlines and key takeaways for Module 1

(1) What is intelligence?
   (a) Key components of intelligence
   (b) Intelligence: The Beginning (1942-50)
   (c) Intelligence: Reloaded (1960-2000)
   (d) Intelligence: Revolutions (2006-)
   (e) A summary of our goals in this course

   Key takeaways:
   - Perception, Cognition and Action are the three parts any intelligent, autonomous agent possesses.
   - Progress in deep learning is driven by cheap computational resources and data availability

(2) Linear Regression, Perceptron, Stochastic Gradient Descent
   (a) Problem setup for machine learning
   (b) Linear regression
   (c) Perceptron
   (d) Stochastic Gradient Descent

   Key takeaways:
   - Designing an accurate predictor for in-sample is trivial, since a hash map will do. We want predictors that generalize to new data outside the training set.
   - SGD is a simple optimization technique used in the perceptron algorithm where weights are updated every time the perceptron makes mistakes on a datum.

(3) Kernels, Beginning of neural networks
   (a) Digging deeper into the perceptron
   (b) Creating nonlinear classifiers from linear ones
   (c) Kernels
   (d) Learning the feature vector
   (e) Deep feed-forward networks

   Key takeaways:
   - Kernels are powerful because they do not require you to think of the feature and parameter spaces.

- Mercer's theorem says that as long as your function of comparing two samples in the data set satisfies the basic properties of a kernel function, there exists some feature space which your function implicitly constructs.

(4) Deep fully-connected networks, Backpropogation
   (a) Deep fully-connected networks
   (b) The backpropogation algorithm

   Key takeaways:
   - A deep network creates new features by composing older features.
   - With deep learning, the level of transferable insights from specific fields like NLP, speech processing, etc. has increased, and the bar for entering a new field is much lower.
   - Provided the deep network has enough number of layers and enough features at each layer, it can fit any dataset.
   - ReLU activation functions are the most popular, compared to threshold, logistic, hyperbolic tangent, etc.
   - Backpropogation is an algorithm for computing the gradient of the loss function for a deep network.
   - The backprop gradient is shared equitably among the different quantities that took part in the forward computation.
   - Forward and backward functions exist for every layer, including activation layers.

(5) Convolutional Architectures
   (a) Basics of the convolution operation
   (b) How are convolutions implemented?
   (c) Convolutions for multi-channel images in a deep network
   (d) Translational equivariance using convolutions
   (e) Pooling to build translational invariance

   Key takeaways:
   - Convolutions work in the same way for two-dimensional or three-dimensional input signals. The kernel will be a matrix of size $k \times k$ versus $k \times k \times k$ respectively.
   - Just like fully-connected layers, we can also stack up convolutions. The effective receptive field increases as we go up the layers.
   - Convolutions are the most heavily used operator in a deep network. We therefore need to implement them as efficiently as we can.
   - If you translate the signal by δ, then the output of convolution is also translated by the same amount. This property is called equivariance, and it holds for 2D convolutions.
   - Pooling is an operation that smears out the features locally in the neighborhood of each pixel.
   - Max pooling has a side-benefit of reducing the number of operations in a deep network and the number of parameters by sequentially reducing the size of the feature map with layers.

- Too much pooling will dramatically reduce the signal in the input image.

(6) Data augmentation, Loss functions
   (a) Data augmentation
   (b) Loss functions

   Key takeaways:
   - Augmenting the data means to create variants of each datum in some simple way such that we know that its label is unchanged.
   - Most popular data augmentations techniques are changing brightness, contrast, cropping the image to simulate occlusions, flipping or jittering the image, warping the image using a projection, zooming into a section, etc.
   - The goal of data augmentation is similar to replacing fully-connected layers with convolutions and pooling. Both make the model invariant to translations.
   - By being invariant to a larger set of nuisances than necessary, we waste model parameters and risk getting a larger test error. In general, the nuisances that the model should be invariant to depends upon the application.

(7) Bias-Variance Trade-off
   (a) Bias-Variance Decomposition
   (b) Weight Decay
   (c) Dropout
   (d) Batch-Normalization

   Key takeaways:
   - If we want to measure how well a model works on new data, we are interested in the population risk not the empirical risk (training loss).
   - We do not have access to a lot of different datasets to measure the bias or the variance. This is why the bias-variance trade-off, although fundamental in statistics and a great thinking tool, is of limited direct practical value.
   - For deep networks, the classical bias-variance trade-off becomes a "double descent" curve where the population risk keeps decreasing even if we fit very large models on relatively small datasets.
   - Cross-validation trains k different models, each time a fraction $(k-1)/k$ of the data is used as the training set and the remainder is used as the validation set. The validation performance of k models obtained by this process is averaged and used as a score to evaluate a model design.
   - Restricting the space of models that the training process searches over to fit in the data is called regularization.

Table of lecture and recitation topics:

Lecture	**Topic**
1	Introduction, History of deep learning
Rec 1	Python, Numpy, Google Colab; ngork, mounting Google Drive, wand.db
2	Linear Regression, Perceptron, stochastic gradient descent
Rec 2	PyTorch I: Syntax, basics of autograd, various modules, layout of the library
3	Kernel methods
4	Beginning of neural networks
Rec 3	PyTorch II
5	Backpropagation
6	Convolutional architectures
Rec 4	Using the AWS cloud
7	Data Augmentation, Loss functions
8	Dropout, Batch-Normalization
Rec 5	Neural architectures: hall of fame
9	Recurrent, Attention-based architectures