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

ESE 546, FALL 2020

MODULE 2 SUMMARY

SHEIL SARDA [SHEIL@SEAS]

Lecture outlines and key takeaways for Module 2

(1) Background on Optimization, Gradient Descent (Chapter 9)
   - (a) Convexity
   - (b) Introduction to Gradient Descent
     - (i) Conditions for optimality
     - (ii) Different types of convergence
   - (c) Convergence rate for gradient descent
     - (i) Some assumptions
     - (ii) GD for convex functions
     - (iii) Gradient descent for strongly convex functions
   - (d) Limits on convergence rate of first-order methods

   Key takeaways:
   - Monotonicity of the gradient implies convexity.
   - The loss function l is always be a function of the entire dataset.
   - For gradient descent, if we pick the step-size $\eta \leq \frac{1}{L}$, the gradient descent always improves the value of the function with each iteration and also improves the distance of the weights to the global minimum at each iteration.
   - Strong convexity enables fewer iterations to converge. Compared to the $\mathcal{O}(1/\epsilon)$ iterations required for convex functions, strongly convex functions require only $\mathcal{O}(\log(1/\epsilon))$ iterations.
   - Nesterov's lower bound suggests the existence of gradient-based algorithms for convex functions which require $\mathcal{O}(1/\epsilon^2)$ iterations.

(2) Accelerated Gradient Descent (Chapter 10)
   - (a) Polyak's Heavy Ball Method
     - (i) Polyak's method can fail to converge
   - (b) Nesterov's method
     - (i) Yet another way to write Nesterov's updates
     - (ii) How to pick the momentum parameter?

   Key takeaways:

- We can think of the gradient applied to the weight at time t as a force that acts on a particle to update its position between time steps. This particle has no inertia, so the force applied directly affects its position.
- If we give the particle a point mass and some inertia, instead of the force directly affecting the position, we can apply Newton's second law of motion $F = ma$.
- The caveat with relying on inertia to make progress is overshooting behavior around the global minimum since inertia is often very different from the gradient. This results in oscillating behavior.
- Nesterov's method removes the oscillation problem of Polyak by incorporating dampening or friction like in the case of a simple harmonic oscillator.

(3) Stochastic Gradient Descent (Chapter 11)
   (a) SGD for least-squares regression
   (b) Convergence of SGD
      (i) Typical assumptions in the analysis of SGD
      (ii) Convergence rate of SGD for strongly-convex functions
      (iii) When should one use SGD in place of GD?
   (c) Accelerating SGD using momentum
      (i) Momentum methods do not accelerate SGD
   (d) Understanding SGD as Markov Chain

   Key takeaways:
   - It is difficult to do gradient descent if the number of samples n is large because the gradient is a summation of a large number of terms.
   - Epochs is a construct introduced in the deep learning libraries for book-keeping purposes, allowing apples-to-apples comparisons between different algorithms.
   - After $w^t \in (w_{min}, w_{max})$ (the zone of confusion), there is no real convergence of the weights.
   - If the learning rate is large, SGD makes quick progress outside of the zone of confusion but bounces around a lot inside the zone of confusion.
   - If the learning rate is too small, SGD is slow outside of the zone of confusion but does not bounce around too much inside the zone.

Table of lecture and recitation topics:

Lecture	**Topic**
14	Gradient Descent
15	Midterm Exam
Rec 9	Midterm Discussion
16	Momentum (heavy-ball, Nesterov), Adam, Early stopping
17	Stochastic Gradient Descent I
Rec 10	Tricks of Trade in training neural networks

ESE 546, FALL 2020
MODULE 3 SUMMARY

SHEIL SARDA [SHEIL@SEAS]

Lecture outlines and key takeaways for Module 3

(1) Stochastic Gradient Descent (Chapter 11)
   (a) SGD for least-squares regression
   (b) Convergence of SGD
      (i) Strongly convex functions
      (ii) What is the appropriate notion of convergence?
      (iii) Descent Lemma for SGD
      (iv) Typical assumptions in the analysis of SGD
         (A) Stochastic gradients are unbiased
         (B) Second moment of gradient norm does not grow too quickly
      (v) Descent Lemma for SGD with additional assumptions
      (vi) Convergence rate of SGD for strongly convex functions
      (vii) Optimality gap for SGD (Theorem)
      (viii) Heuristic for training neural networks
      (ix) Convergence rate of SGD for decaying step-size (Theorem)
      (x) Convergence rate for mini-batch SGD
      (xi) When should one use SGD in place of Gradient Descent?
   (c) Accelerating SGD using momentum
      (i) Polyak-Ruppert averaging
      (ii) Momentum methods do not accelerate SGD
      (iii) Why do we use Nesterov's method to train neural networks?
   (d) Understanding SGD as a Markov Chain
      (i) Gradient flow
      (ii) Markov chains
      (iii) Invariant distribution of a Markov chain
      (iv) Time spent at a particular state by the Markov chain
      (v) A Markov chain model of SGD
         (A) Transition probability of SGD
         (B) Variance of SGD weight updates
         (C) SGD is like GD with Gaussian noise
      (vi) The Gibbs distribution

        (vii) Convergence of a Markov chain to its invariant distribution
        (viii) KL Divergence monotonically decreases (Lemma)

(2) Shape of the energy landscape of neural networks (Chapter 12)
    (a) Introduction
    (b) Deep Linear Networks
    (c) Extending the picture to deep networks
(3) Object Detection & Image Segmentation (Recitation 11)
    (a) Object Detection Framework
    (b) R-CNN
    (c) Fast R-CNN
    (d) Faster R-CNN
    (e) You Only Look Once (YOLO)
    (f) Image Segmentation
    (g) Mask R-CNN
(4) Generalization Bounds (Recitation 10)
    (a) PAC learning
    (b) Union and Chernoff Bounds
        (i) Union Bound (or Boole's Inequality)
        (ii) Measure Concentration Inequalities
    (c) Generalization Bounds and Uniform convergence
    (d) VC-dimension

Table of lecture and recitation topics:

Lecture	Topic
18	Stochastic Gradient Descent I
19	Stochastic Gradient Descent II
Rec 11	Vignette: Object Detection
20	Markov Chains
Rec 12	Generalization Bounds

ESE 546, FALL 2020
MODULE 4 SUMMARY

SHEIL SARDA [SHEIL@SEAS]

Lecture outlines and key takeaways for Module 4

Shape of the energy landscape of neural networks (Chapter 12)

(1) Introduction
(2) Deep Linear Networks
  (a) Least squares solution
  (b) Projection of a vector onto a matrix
  (c) Back to deep linear networks
  (d) Critical points of B if A is fixed
  (e) Critical points of A if B is fixed
  (f) Critical points of (A, B)
  (g) If W is a critical point then it can be written as a projection of the least squares solution on the subspace spanned by some p eigenvectors of Σ
  (h) If W is the global minimum for a two-layer network then it is a projection of the solution for a single-layer network onto the subspace spanned by the top p eigenvectors of Σ
  (i) There are exponentially many saddle points for a two-layer network
  (j) No local minima in a deep linear network, all minima are global and not unique
  (k) All the previous results are true for multi-layer linear networks
(3) Extending the picture to deep networks

Key takeaways

- Saddle points are critical points but which are neither local minima nor local maxima
- Weight decay changes the number of global minima, and only the ones that have the smallest L2 norm remain in the energy landscape. It also reduces the number of saddle points
- Essentially, the objective of deep networks is not convex, but current results indicate that it is quite benign and this is perhaps the reason why it is so easy to train them

Generalization performance of machine learning models (Chapter 13)

(1) The PAC-Learning Model
  (a) PAC-learnable hypothesis class
  (b) Learning Monotone Boolean Formulae
(2) Concentration of Measure

    (a) Union Bound (or Boole's Inequality)
    (b) Weak Law, Strong Law and Central Limit Theorem
    (c) Chernoff Bound
(3) Uniform Convegence
(4) Vapnik-Chernovenkis (VC) dimension
    (a) Shattering a set of inputs
    (b) Bounds on the VC-dimension of deep neural networks

Key takeaways

- PAC-Learning assumes nothing about how low L is going to use the data, e.g. whether it runs SGD or what surrogate loss it uses or even whether it performs Empirical Risk Minimization
- The PAC-Learning model is very abstract and we should expect only qualitative answers from this theory
- The set of inputs D is shattered by the concept class if we can achieve every possible labeling out of the 2^n labelings using the concept.
- If the VC-dimension of a concept class is finite, then this class has the uniform convergence property. THerefore, we can learn this concept class agnostically in the PAC framework with training data
- The number of samples required to learn a concept scales linearly with the VC-dimension V
- The gap in applying VC-theory to deep networks steps from the need for uniform convergence: we may not need that the empirical and population risk are close for all hypotheses in the class

Variational Inference (Chapter 14)

(1) The model
(2) Some technical basics
    (a) Variational calculus
        (i) Picking the domain and objective in variational optimization
        (ii) Choosing a functional to measure the distance between q and p
    (b) Laplace approximation
    (c) Digging deeper into KL-Divergence
(3) Evidence Lower Bound (ELBO)
    (a) Parameterizing ELBO
(4) Gradient of the ELBO
    (a) The Reparameterization Trick
    (b) Score function estimator of the gradient
    (c) Gradient of the remaining terms in ELBO
(5) Some comments

Key takeaways

- The latent factors of data are not known to us if we do not take part in the generative process. Nature is in charge of generating the data and our goal here is to guess the parameters of this generative model
- Since the KL-divergence is zero if and only if the two distributions are equal, we are never going to be able to minimize it completely. Essentially, the crux of variational inference boils down to picking a good family of distributions Q that makes this optimization easy
- Our goal in generative modeling is computing Nature's true posterior distribution of latent factors
- The prior p will be a mean-field Gaussian distribution. The prior has no parameters in some cases
- The reparameterization trick allows us to obtain the backpropogation gradients across the sampling operations via a creative use of the Laplace approximation of the distribution q
- The key difference between the reparameterization trick and the score-function estimator is that in the latter we do not need to make sure that the gradient can be back-propogated across the sampling operation

Generative Adversarial Networks (Chapter 15)

(1) Two-sample tests and Discriminators
(2) Building the Discriminator in a GAN
(3) Building the Generator of a GAN
(4) Putting the discriminator and generator together
    (a) Training a GAN
    (b) Solving min-max problems is difficult
    (c) A harsh discriminator inhibits the training of the generator
(5) How to perform valiation for a GAN
    (a) Frechet Inception Distance (FID)
    (b) Maximum Mean Discrepency (MMD)
(6) The zoo of GANs

Key takeaways

- Models for which we can calculate the likelihood of sample being from a distribution are called explicit generative models. An alternative class of generative models which are implicit only give a sample x but do not report its likelihood.
- A good statistic is the one that lets us distinguish between data that comes from Nature's distribution and data that is synthesized by our generative model. This statistic is the discriminator.
- A GAN is solving the min-max problem: generators minimize the objective and discriminator maximize the objective

- For variational generative models, if the log-likelihood of new images is similar to the log-likelihood of images in the training data then the new images are good as last as far as the model is concerned
- GANs can often end up memorizing the training data and generate very realistic images that are essentially the same as those in the training data

Table of lecture and recitation topics:

Lecture	Topic
21	SGD III, Background on Principal Component Analysis
Rec 12	Generalization bound using uniform convergence
22	Deep Linear networks
Rec 13	Vignette: Deep reinforcement learning
23	Background: Information theory and Variational Inference
24	Auto-Encoders, ELBO I
Rec 14	Recap of info theory, variational auto-encoders
25	ELBO II
26	Bayesian neural networks
27	Generative Adversarial Networks, Recap of post-midterm topics