Draft Notes for Deep Learning

Moonsu Kang

Aug 2016

This is note taken from Deep Learning book by Ian Goodfellow. http://www.deeplearningbook.org/
Due to lack of time, I have copied and pasted large part of his book instead of writing notes on my own words. I have read through whole book but haven't got to cover in notes as I was travelling back then.

Introduction

Deep Learning: This book is about a solution to intuitive problems that we face with using computer to solve. Machine Learning allows computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones.
AI systems need the ability to acquire their own knowledge, by extracting patterns from raw data. This capability is known as machine learning. The performance of machine learning algorithms depends heavily on the representation of the data they are given. Each piece of information included in the representation of the patient is known as a **feature**.
One solution to this problem is to use machine learning to discover not only the mapping from representation to output but also the representation itself. This approach is known as representation learning. Through representation we are interested is separating **factors of variation** that affect the way that data is shown. Disentangle factors of variation and discard unneccessary ones is definitely a hard task. **Deep Learning** solves this problem in representation learning by introducing representations that are expressed in terms of other, simpler representations. It allows a computer to build complex concepts out of simpler concepts.
To summarize, deep learning is an approach to AI. Specifically, it is a type of machine learning, a technique that allows computer systems to improve with experience and data. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler

concepts, and more abstract representations computed in terms of less abstract ones.
Part I introduces basic mathematical tools and machine learning concepts. Part II describes the most established deep learning algorithms that are essentially solved technologies. Part III describes more speculative ideas that are widely believed to be important for future research in deep learning.

0.0.1 Connectionist movement

Distributed representation: the idea that each input to a system should be represented by many features, and each feature should be involved in the representation of many possible inputs. **Back-propagation**, **LSTM**: long short-term memory. The wave of neural networks research popularized the use of the term deep learning from 2006. The third wave began with a focus on new unsupervised learning techniques and the ability of deep models to generalize well from small datasets, but today there is more interest in much older supervised learning algorithms and the ability of deep models to leverage large labeled datasets.

0.0.2 Achievements

In recent years, it has seen tremendous growth in its popularity and usefulness, due in large part to more powerful com- puters, larger datasets and techniques to train deeper networks. One of achievements of deep learning is its extension to the domain of reinforcement learning. In the context of reinforcement learning, an autonomous agent must learn to perform a task by trial and error, without any guidance from the human operator. Deep learning has also significantly improved the performance of reinforcement learning for robotics. The years ahead are full of challenges and opportunities to improve deep learning even further and bring it to new frontiers.

I. Basics

1 Linear Algebra

For $\boldsymbol{Ax} = \boldsymbol{b}$ to be $\boldsymbol{x} = \boldsymbol{A^{-1}b}$, $\boldsymbol{A^{-1}}$ needs to exist, it needs to be full-rank. The span of a set of vectors is the set of all points obtainable by linear combination of the original vectors. Determining whether $\boldsymbol{Ax} = \boldsymbol{b}$ has a solution thus amounts to testing whether b is in the span of the columns of A. This particular span is known as the column space or the range of A.
In order for the system $\boldsymbol{Ax} = \boldsymbol{b}$ to have a solution for all values of $b \in \mathbb{R}^m$ (full rank), it requires that the column space of A be all of $\mathbb{R}^m$. 1) Having $n \gg m$ is necessary, and 2) m number of columns, out of n, should be linearly

independent.
In order for matrix to have an inverse, in addition $\boldsymbol{Ax = b}$ has at most one solution for each value of $\boldsymbol{b}$. To do so, the matrix can have at most m columns.

Eigendecomposition: decompose a matrix into a set of eigenvectors and eigenvalues.

$$\boldsymbol{Av} = \lambda \boldsymbol{v}$$
$$\boldsymbol{A} = \boldsymbol{V} diag(\lambda) \boldsymbol{V^{-1}}$$

where v is an Eigenvector of A and λ is eigenvalue. A matrix whose eigenvalues are all positive is called positive definite. A matrix whose eigenvalues are all positive or zero-valued is called positive semidefinite.

Singular Value Decomposition The SVD allows us to decompose a matrix into singular vectors and singular values. SVD is more generally applicable as any real matrix can have singular value decomposition.

$$\boldsymbol{A = UDV^T}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are definted to be orthogonal matrices, while $\boldsymbol{D}$ is known to be a diagonal matrix. Elements along the diagonal of $\boldsymbol{D}$ are known as singular values. Columns of $\boldsymbol{U}$ are known as left-singular vectors; of $\boldsymbol{v}$ are right-singular vectors. Lef-signular vectors are the eigenvectors of $\boldsymbol{AA^T}$.

Moore-Penrose Pseudoinverse: Pseudoinverse of A is defined as a matrix:

$$\begin{aligned} \boldsymbol{A}^{+} &= \lim_{a \to 0} (\boldsymbol{A^T A} + \alpha \boldsymbol{I})^{-1} \boldsymbol{A^T} \\ &= \boldsymbol{V D^{+} U^T} \end{aligned}$$

Pseudoinverse of A is helpful with finding solutions for cases when $\boldsymbol{A}$ is not square.

Trace Operator: gives the sum of all of diagonal entries of a matrix

$$Tr(\boldsymbol{A}) = \sum_i \boldsymbol{A_{i,i}}$$

Determinant: The determinant of a square matrix, denoted $det(\boldsymbol{A})$, is a function mapping matrices to real scalars. The determinant is equal to the product of all the eigenvalues of the matrix. The absolute value of the determinant can be thought of as a measure of how much multiplication by the matrix expands or contracts space.

2 Probability and Information Theory

There are three possible sources of uncertainty: 1. Inherent stochasticity in the system being modeled

2. Incomplete observability
3. Incomplete modeling
Probability theory provides a set of formal rules for determining the likelihood of a proposition being true given the likelihood of other propositions.A probability distribution is a description of how likely a random variable or set of random variables is to take on each of its possible states.

2.0.1 Distributions

Multivariate Normal Distribution:

$$N(x, \mu, \Sigma) = \sqrt{\frac{1}{(2\pi)^n det(\Sigma)}} exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$$

Exponential Distribution: a probability distribution with a sharp point at $x = 0$.

$$p(x; \lambda) = \lambda \mathbf{1}_{\boldsymbol{x} \geq \mathbf{0}} exp(-\lambda x)$$

Laplace distribution: probability distribution that allows us to place a sharp peak of probability mass at an arbitrary point μ

$$Laplace(x; \mu, \gamma) = \frac{1}{2\gamma} exp(-\frac{|x-\mu|}{\gamma})$$

Mixtures of Distributions: mixture distribution is made up of several component distributions. On each trial, the choice of which component distribution generates the sample is determined by sampling a component identity from a multinoulli distribution:

$$P(x) = \sum_i P(c = i) P(x|c = i)$$

A common type of mixture model is the Gaussian mixture model, in which the components $p(x|c = i)$ are Gaussians. Each component,i, has a separately parametrized mean $\mu^(i)$ and covariance $\Sigma^(i)$, along with prior probability $\alpha_i = P(c = i)$. **softplus function**: softplus function is softened version of max(0,x) function.

$$\varsigma(x) = log(1 + exp(x))$$

2.0.2 Measure Theory

One of the key contributions of measure theory is to provide a characterization of the set of sets that we can compute the probability of without encountering paradoxes.Measure theory provides a rigorous way of describing that a set of points is negligibly small. Such a set is said to have "measure zero." Another useful term from measure theory is "almost everywhere." A property that holds almost everywhere holds throughout all of space except for on a set of measure zero. Because the exceptions occupy a negligible amount of space, they can be safely ignored for many applications. Some important results in probability theory hold for all discrete values but only hold "almost everywhere" for continuous values.

2.0.3 Informational Theory

Information Theory quantifies how much information is present in a signal. In this textbook, we mostly use a few key ideas from information theory to characterize probability distributions or quantify similarity between probability distributions. - Likely events should have low information content; Less likely events should have higher information content - Independent events should have additive information **self-information of an event** $x = x$

$$I(x) = -logP(x)$$

Self-information deals only with a single outcome. We can quantify the amount of uncertainty in an entire probability distribution using the **Shannon entropy**:

$$H(x) = \mathbf{E}_{x \sim P}[I(x)] = -\mathbf{E}_{x \sim P}[logP(x)]$$

Kullback-Leibler (KL) divergence: measure how different two distributions are over x

$$D_{KL}(P \| Q) = \mathbf{E}_{x \sim P}[log \frac{P(x)}{Q(x)}]$$

A quantity that is closely related to the KL divergence is the **cross-entropy**, which is similar to the KL divergence but lacking the term on the left

$$\begin{aligned} H(P, Q) &= H(P) + D_{KL}(P \| Q) \\ &= \mathbf{E}_{x \sim P} logQ(x) \end{aligned}$$

Minimizing the cross-entropy with respect to Q is equivalent to minimizing the KL divergence, because Q does not participate in the omitted term.

2.0.4 Structured Probabilistic Models

Machine learning algorithms often involve probability distributions over a very large number of random variables. Using a single function to describe the entire joint probability distribution can be very inefficient (both computationally and statistically).Instead of using a single function to represent a probability distribution, we can split a probability distribution into many factors that we multiply together. Factorizations can greatly reduce the number of parameters needed to describe the distribution. When we represent the factorization of a probability distribution with a graph, we call it a **structured probabilistic model** or graphical model. There are two main kinds of structured probabilistic models: directed and undirected. Both kinds of graphical models use a graph G in which each node in the graph corresponds to a random variable, and an edge connecting two random variables means that the probability distribution is able to represent direct interactions between those two random variables. Directed models use graphs with directed edges, and they represent factorizations into conditional probability distributions.

$$p(x) = \Pi_i p(x_i | P_{a_g}(x_i))$$

Undirected models use graphs with undirected edges, and they represent factorizations into a set of functions; unlike in the directed case, these functions are usually not probability distributions of any kind. Factors are just functions, not probability distributions. The output of each factor must be non-negative, but there is no constraint that the factor must sum or integrate to 1 like a probability distribution. We therefore divide by a normalizing constant Z, defined to be the sum or integral over all states of the product of the ϕ functions, in order to obtain a normalized probability distribution:

$$p(x) = \frac{1}{Z}\Pi_i \phi^{(i)}(C^{(i)})$$

3 Numerical Computation

3.0.1 Overflow and Underflow

Under- flow occurs when numbers near zero are rounded to zero. Many functions behave qualitatively differently when their argument is zero rather than a small positive number. Overflow occurs when numbers with large magnitude are approximated as ∞. Most readers of this book can simply rely on low- level libraries that provide stable implementations but good to keep in mind.

3.0.2 Poor Conditioning

Conditioning refers to how rapidly a function changes with respect to small changes in its inputs. For function $f(x) = \boldsymbol{A^{-1}}x$, conditional number is equal to $max_{i,j}|\frac{\lambda_i}{\lambda_j}|$. This is the ratio of the magnitude of the largest and smallest eigenvalue.

3.0.3 Gradient Based Optimization

Partial Derivative:gradient generalizes the notion of derivative to the case where the derivative is with respect to a vector: the gradient of f is the vector containing all of the partial derivatives, denoted $\nabla_x f(x)$.

Directional Derivative: the slope of the function f in direction u. the directional derivative is the derivative of the function $f(x + au)$ with respect to a , evaluated at $a = 0$.

Jacobian matrix: Matrix that contains all of the partial derivatives of a function whose input and output are both vectors.

$$J_{i,j} = \frac{d}{dx_j} f(x)_i$$

where $f : \mathbb{R}^m \to \mathbb{R}^n$ and $\boldsymbol{j} \in \mathbb{R}^{n\times m}$.

Hessian matrix: Matrix that contains second derivatives, where

$$H(f)(x)_{i,j} = \frac{d^2}{dx_i dx_j} f(x)$$

Equivalently, the Hessian is the Jacobian of the gradient. In multiple dimensions, there can be a wide variety of different second derivatives at a single point. The condition number of the Hessian measures how much the second derivatives vary. When the Hessian has a poor condition number, gradient descent performs poorly. It also makes it difficult to choose a good step size. This issue can be resolved by using information from the Hessian matrix to guide the search. The simplest method for doing so is known as **Newton's method**. Newton's method is based on using a second-order Taylor series expansion to approximate $f(x)$ near some point $x^(0)$:

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T \nabla_x f(x^{(0)}) + \frac{1}{2}(x - x^{(0)})^T H(f)(x^{(0)})(x - x^{(0)})$$

$$x^* = x^{(0)} - H(f)(x^{(0)})^{-1} \nabla_x f(x^{(0)})$$

Iteratively updating the approximation and jumping to the minimum of the approximation can reach the critical point much faster than gradient descent would. This is a useful property near a local minimum, but it can be a harmful property near a saddle point.
Optimization algorithms such as gradient descent that use only the gradient are called first-order optimization algorithms. Optimization algorithms such as New- ton's method that also use the Hessian matrix are called second-order optimization algorithms.

Lipschitz continuous: A Lipschitz continuous function is a function f whose rate of change is bounded by a Lipschitz constant L:

$$\forall x, \forall y, |f(x) - f(y)| \leq L \|x - y\|_2$$

This property allows us to quantify our assumption that a small change in the input made by an algorithm such as gradient descent will have a small change in the output.

3.0.4 Constrained Optimization

The Karush–Kuhn–Tucker (KKT) approach provides a very general solution to constrained optimization. With the KKT approach, we introduce a new function called the generalized Lagrangian or generalized Lagrange function.The generalized Lagrangian is then defined as:

$$L(\boldsymbol{x}, \lambda, \alpha) = f(x) + \sum_i \lambda_i g^{(i)}(x) + \sum_j \alpha_j h^{(j)}(x)$$

where $g^{(0)}, h^{(0)}$ are equality/inequality constraints and λ, α are KKT multipliers.

$$\mathbb{S} = \{x | \forall i, g^{(i)}(x) = 0 \text{ and } \forall j, h^{(j)} \leq 0\}$$

4 Machine Learning Basics

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

The error incurred by an oracle making predictions from the true distribution $p(x, y)$ is called the Bayes error.