Contents

1 Deep Learning Survey ................................. 1
  1.1 Network Architectures ................................ 1
  1.2 Problem Domains ..................................... 2
  1.3 Types of Problems .................................. 2
  1.4 Engineering concerns ............................... 2

2 Deep Learning Problems ............................... 2
  2.1 Standard Computer Vision problems ............... 2
  2.2 Natural Language Processing (NLP) Problems ..... 3
  2.3 Datasets ............................................. 3
  2.4 Networks .......................................... 3

3 Optimization ........................................... 3
  3.1 Gradient Descent .................................... 4
  3.2 Momentum based methods .......................... 5
  3.3 Adaptive approaches: ............................... 5

1 Deep Learning Survey

Conceptually these 4 sections: network architecture, problem domains, problem types, and engineering concerns create a 4D grid describing the field of DL.

1.1 Network Architectures

- Multi-Layer Perceptron (MLP)
  - network has fully connected layers

- Convolutional Neural Nets (CNNs)
  - useful for images
  - spatial regularity is embedded in the network architecture

- Recurrent Neural Nets (RNN)
  - CNNs with a sense of internal state over time
  - array and weight sharing are over time

- Graph Neural Nets (GNN)
  - nearby items are more related

- Transformers
can access input data elsewhere and weight share

- We can tune these networks with various levels of specificity but for the scope of this class we will focus on common underlying problems that may occur.

1.2 Problem Domains

- Vision
- Natural Language Processing (NLP)
- Control

For the scope of this class, we will explore certain domains to build intuition and experience designing networks and understand trade offs. Additionally research in this field is commonly in one of these domains so literacy is a plus.

1.3 Types of Problems

- Regression - to predict reals numbers
- Classification - to categorize
- Generation - to make/synthesize
- Recommendation (including conditional generation) - often to commercialize and make money

Deep learning aims to identify underlying regularities for these problems.

1.4 Engineering concerns

- Optimizer choice
- Regularization (augmentation, normalization, explicit, weight-sharing)
- Pre-training and self-supervision
  - We know learning models like data, so can we use external data to enhance the ML model? Theoretically, the network is able to learn regularities that are present elsewhere a large dataset. Once the actual data is presented, the network can focus on optimizing the nuances in the data.
- Scaling
  - larger models (more layers, units, data) tend to work better but it needs to be trained first
  - the network is tweaked to work and also scaled to run on various components and parallel clusters
  - the network is also often scaled down to run on devices for deployment
- Experimentation
- Debugging

2 Deep Learning Problems

2.1 Standard Computer Vision problems

1. Object classification - What?
2. Object location - Where?
3. Object detection - What and where?
4. Semantic segmentation - scene understanding (Can we modify architecture to be better than building a classified for each pixel?)
5. Style transfer - ex. change a picture to an impressionist painting
6. GANs - making fake realistic images (Can you generate images from a class?)

7. Unpaired data testing - How does network perform with data that is not paired? Appropriately paired data generally works well.
   - Example: draw the outline of bread and tell the network to make a cat

2.2 Natural Language Processing (NLP) Problems

- OpenAI GPT-2
  - NLP was previously rule based, but now networks can learn patterns of language
  - Example: the network learns words, grammatical types, sentence structure, flow, and pragmatics but does not necessarily learn how to reflect reality

2.3 Datasets

- CIFAR-10 and CIFAR-100 - images with 10 or 100 datasets
- imageNet - images with 1,000 classes

2.4 Networks

- AlexNet
  - classic medium depth network
  - widely known to be the first NN to attain state of the art results on ImageNet challenge
- ResNet
  - very deep, trainable network
  - does not include a large FC layer at the end, instead just average pools over all positions and has 1 linear layer
  - network development was drive by trying to improve the optimizer
  - network is leading to super-human performance
- fully convolutional networks
  - low-res (but high-depth) processing in the middle integrates context from the entire image
  - up-sampling at the end turns these low-res feature vectors into high-res per pixel predictions
- U-Net architecture
  - concatenate activations from conv layers to upsampling layers
- RNNs
  - How time oriented data can be digested and used for different problems
- Transformers
  - aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease

3 Optimization

In this section, a brief overview of important ideas in numerical optimization algorithms are presented. For a detailed understanding of what each of the methods does, refer to the course material of EE 227C.

- Three important optimization algorithms
  - Gradient Descent (learning rate)
  - Momentum based methods
  - Adaptive approaches (ADAM)
Singular value Decomposition (SVD)

Let us recall the singular value decomposition of a matrix $X$. For real matrices $X$, its singular value decomposition can be written as

$$X = U \Sigma V^T \text{ with } U \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{n \times n} \text{ and } \Sigma \in \mathbb{R}^{m \times n}.$$  \hfill (1)

The matrices $U$ and $V$ are orthonormal matrices and satisfy the properties $U^T U = I_{m \times m}, V^T V = I_{n \times n}$. The matrix $\Sigma$ is a collection of singular values of $X$ along its diagonal. The singular values of $X$ are the positive square roots of non-zero eigenvalues of $XX^T$ or $X^T X$. If $X$ has rank $r$ then there would be $r$ singular values of $X$.

Let the singular values of $X$ be denoted by $\sigma_i$ for $i \in \{1, \ldots, r\}$. The matrix $\Sigma$ can be written as

$$\Sigma = \begin{bmatrix} \text{diag}(\sigma_1 \ldots \sigma_r) & 0_{r \times (n-r)} \\ 0_{(m-r) \times r} & 0_{(m-r) \times (n-r)} \end{bmatrix}$$

We can also see that columns of $U$ are the extended eigenvectors of $XX^T$ and similarly the columns of $V$ are the extended eigenvectors of $X^T X$.

Least Squares

The least squares approximate solution of the equation $Xw = y$ can be found using the following optimization problem.

$$w^* = \arg\min_w ||Xw - y||_2^2$$  \hfill (2)

Utilizing the SVD of $X$ The same optimization problem can be formulated using change of coordinates as follows

$$\min_w ||U \Sigma V^T w - y||_2^2 = \min_w ||(U \Sigma V^T w - U^T y)||_2^2$$  \hfill (3)

Notice that the norm of a vector doesn’t change when it is rotated without stretching. Therefore,

$$\min_w ||U (\Sigma V^T w - U^T y)||_2^2 = \min_w ||\Sigma \tilde{w} - \tilde{y}||_2^2$$  \hfill (4)

where $V^T \tilde{w} = \tilde{w}$ and $U^T y = \tilde{y}$.

3.1 Gradient Descent

The gradient descent update equation for the optimization problem in (2) with a learning rate $\eta$ can be written as

$$w_{t+1} = w_t - 2\eta X^T (y - Xw_t)$$  \hfill (5)

Similarly, the gradient descent update equation for the equivalent optimization problem in (4) can be written as

$$\tilde{w}_{t+1}[i] = \tilde{w}_t[i] - 2\eta \Sigma^T (\Sigma w_t[i] - \tilde{y})$$  

$$= (1 - 2\eta \sigma_i^2) \tilde{w}_t[i] + 2\eta \Sigma^T \tilde{y}$$  \hfill (6)

Notice that the update rule is just written for the $i^{th}$ element of $w$. For the stability of the difference equation in (6), we need

$$1 - 2\eta \sigma_i^2 > -1 \ \forall \ i$$

$$\Rightarrow \eta < \frac{1}{\sigma_i^2} \ \forall \ i$$

$$\Rightarrow \eta < \frac{1}{\sigma_{\max}^2}$$  \hfill (7)

It can be seen from the above choice of $\eta$, $1 - 2\eta \sigma_{\text{text}}^2$ can be close to 1 and the convergence might take an extremely long time to converge. One of the ideas to improve the speed of convergence is to use the concept of ‘momentum’ inspired from the ‘Proportional + Integral (PI) action controller’ which is described in the next section.
3.2 Momentum based methods

**Idea:** Find a way to make the learning rate bigger without causing trouble for the large singular values.

**Observation:** The weights associated with the large singular values oscillate at high frequency as the learning rate is increased. So, to dampen the oscillations out, a low pass filter can be added. It is known from circuit analysis that a low pass filter outputs an exponential average of the input.

**Implementation:**

\[
\begin{align*}
\tilde{w}_{t+1}[i] &= \tilde{w}_t[i] - \eta a_{t+1}[i] \\
a_{t+1}[i] &= (1 - \beta)a_t[i] + \beta("current gradient")
\end{align*}
\]  

(8)

Here \(\beta\) controls how fast we average. For momentum based methods, both the weight and internal state average gradient are evolving. Note, there are several ways to mathematically implement this circuit. Here we define the average gradient (a) to be itself for a flat line.

3.3 Adaptive approaches:

There’s a limit to how much the learning rate can be increased even by the momentum based methods. Momentum still respects SVD and the movement along the directions that are small is still small. The idea in adaptive approaches is to change the learning rates for different singular values (different directions) - More about this in the next lecture.