Ash

March 12, 2018

Contents

List of Figures

Chapter 1

Introduction

Very brief background Deep learning good for large data set. Motivation is to work with less training examples
Existing work on meta learning and continuous learning
Gap the literature
Describe the Problem A system that can do continual learning. Refer to this doc: https://paper.dropbox.com/doc/A
classes-an-existing-classifier-RdKxXHh7M9OWbHvEvCCsV We want it to be scalable with respect to the number of classes So it should work faster than nearest neighbour based approaches for large number of classes
High level how you will solve it and why it is different from existing work
Brief description of experimental setup

Chapter 2

Background

TODO: Week 2

2.1 Hand Engineered and Learnt Features

TODO: Week 2
Features are a general term for characteristic attributes which exist across all samples in a data-set or domain. These features were traditionally hand-engineered by machine learning experts, carefully selecting the base-components of which the data-set in question appears to comprise.
A fundamental problem with hand-engineered features is that it imposes human knowledge onto a problem to be solved by a computer. Furthermore, key features for complex data such as images and video are incredibly difficult to ascertain – especially if desiring generic, transferable features. With the rise of neural networks – specifically CNNs, which will be discussed in detail later – feature-learning has become the norm. This essentially takes the task of feature engineering and solves it in a data-driven manner.

2.2 Supervised Learning

TODO: Week 2
Supervised learning is a machine learning strategy whereby the target solution is presented after each training iteration. This differs from unsupervised learning in that unsupervised learning has no direct target to learn from and is used to find underlying commonalities or patterns in data.
Supervised learning is the most commonly used method for image and video tasks, as typically the objective is to perform tasks where the target is well-defined. Common supervised learning tasks are *image classification* - where the objective is to assign an input image a label from a fixed set of categories; *localisation* - where the objective is to produce the coordinates of an object of interest from the input image; and *detection* - which combines the previous two tasks.
There are a multitude of supervised and unsupervised learning problems, but as with almost all meta-learning strategies I will focus primarily on the supervised task of image classification using neural networks.

2.3 Optimisation

TODO: Week 2

2.3.1 Loss

The process of optimising a machine learning system is to present it with a target of some sort – either in the supervised or unsupervised setting – and compute a numerical quantity called *loss* or *cost*. The loss is a scalar value which is representative of how "badly" the system has performed inference given the input image. It is the system's objective to minimise this value through some optimisation algorithm. The loss function is specifically chosen for the task at hand, *cross-entropy* being a common choice for image classification tasks and *mean-squared (L2) error* for localisation.

Symbols

Before discussing loss functions, it's important to understand the inputs and outputs of a machine learning system when performing image classification.
For a system that makes predictions between N classes, its input is a vector of image features $\boldsymbol{x}$ - usually the raw pixel values. The system's output is a vector $\hat{\boldsymbol{y}}$ of length N, where each of the output values $\hat{\boldsymbol{y}}_i$ is a score for class i being the correct answer. The loss $\mathcal{L}$, as described above, is a function of the predictions $\hat{\boldsymbol{y}}$ and the target values $\boldsymbol{y}$.
The target values are typically encoded as a *one-hot* vector of length N, which is all zeros with a 1 in the position of the correct class.

Softmax

As the system's outputs aren't normalised and thus cannot be interpreted as a true confidence measure, the outputs normally go through a softmax function σ.

$$\sigma(\hat{\boldsymbol{y}})_i = \frac{e^{\hat{y}_i}}{\sum_{k=1}^{N} e^{\hat{y}_i}} \tag{2.1}$$

The softmax function (eq 2.1) squashes the arbitrary scores into a vector such that its values sum to 1 and are each in the range $[0, 1]$. The resultant values can be interpreted as the probability of the input image falling into each of the classes.

Cross-Entropy Loss

With the machine learning system producing a normalised probability distribution across classes, those values need be compared with the targets to produce a scalar loss value. Cross-entropy loss, otherwise known as *log loss*, penalises for differences between predicted values and targets, with the penalty growing harsher for further-away predictions as demonstrated in fig 2.1.
For a vector of predictions $\hat{\boldsymbol{y}}$ and a one-hot target vector $\boldsymbol{y}$, the cross-entropy loss is:

$$H(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{k=1}^{N} y_k log(\hat{y}_k) \tag{2.2}$$

For the special case of *binary* cross-entropy (as shown in fig 2.1) where number of classes N is 2, the network's outputs are generally reduced to a single scalar value and cross entropy calculated as:

$$H(\hat{y}, y) = -(ylog(\hat{y}) + (1-y)log(1-\hat{y}) \tag{2.3}$$

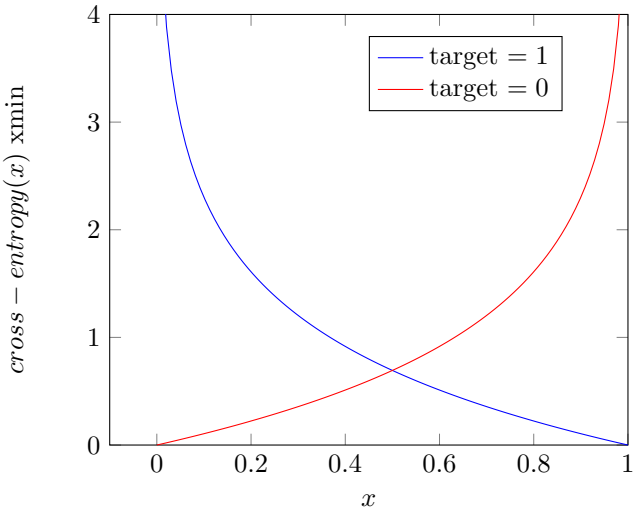

Figure 2.1: Binary Cross-Entropy

Mean-Squared Error (L2)

Mean-squared error is used for regression tasks, where the objective is to predict a quantitative value rather than a measure of probability. As L2 loss minimises the average error between the predictions $\hat{\boldsymbol{y}}$ and targets $\boldsymbol{y}$, the system learns to make predictions which lie in the mean position of these, which is generally ideal for regression tasks where there is one solution.

$$L2(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{k=1}^{N} (y_k - \hat{y}_k)^2 \tag{2.4}$$

2.4 Neural Networks

Artifical Neural networks (*ANNs*) are machine learning systems loosely inspired by the functioning of the biological neural networks of the brain. They are composed of artificial neurons which transmit signals from one another in the form of a non-linear function of the sum of the incoming inputs.
TODO THE FOLLOWING
- Explain why linearity sucks - Explain give motivation for layers - Describe activation functions
If we look at the structure of a generic neuron f_1 (**see fig TODO**) we see that it is composed of two components - a weight w_1 and a bias b_2. Passing a value x through a neuron f_1 is equivalent to computing the linear function $f_1(x) = w_1 x + b_1$.

If the outputs of this neuron are passed into a similar neuron f_2, we end up with the composite function

$$(f_2 \circ f_1)(x) = (w_2(w_1 x + b_1) + b_2) \tag{2.5}$$
$$= w_2 w_1 x + b_1 w_2 + b_2 \tag{2.6}$$

which is still a linear function of x. This is true no matter how many neurons in a row, meaning that any number of linear-function neurons are only as good as one.

2.4.1 SGD

TODO: Week 2
ANNs typically use gradient-descent for their optimization, which is the process of blah blah loss curve etc. - Gradient descent
- Loss curve
- Why
- How
- Why use Sgd

2.4.2 Backprop

TODO: Week 2

2.4.3 Adaptive Optimizers

SGD with Momentum

Nesterov Accelerated Gradient

Adagrad

Adadelta

RMSprop

Adam

2.5 Dealing with Small Training Data Sets

2.5.1 Overfitting

2.5.2 Transfer Learning

2.5.3 Few-Shot Learning

2.5.4 Meta Learning

Break into meta training, testing, episodes, etc.

2.6 Continuous Learning

Catastrophic forgetting

2.7 Modern Deep Learning Architectures

2.7.1 Convolutional Neural Networks

2.7.2 Recurrent Neural Networks

Chapter 3

Related Works

3.1 Meta Learning

Approaches in meta learning with neural networks are generally groupd into three categories.

3.1.1 Model Based

3.1.2 Metric Based

3.1.3 Optimization Based

3.2 Continuous Learning

Chapter 4

Proposal