The How, What and Why of Deep Learning

Seminar 1: Introduction to ML

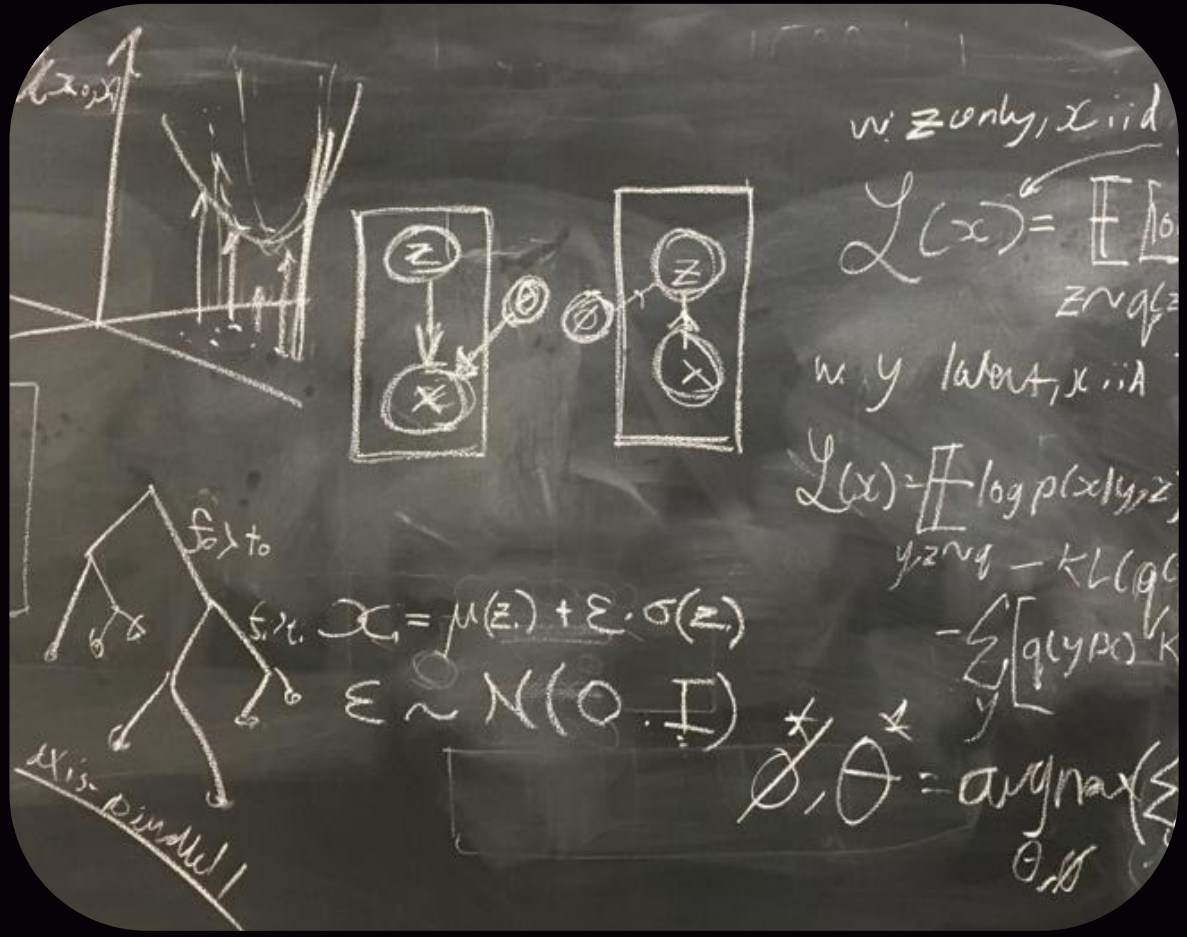
From Generalised Linear Models to Multi-Layer Perceptrons

Big Data Institute, Oxford
March 22nd, 2019

Alexander Camuto & Matthew Willetts

Course Timetable

- **22nd March**: Introduction: MLPs
- **5th April**: Computational Graphs: Implementations with Keras, Optimisation & Regularisation methods
- **19th April**: Convolutional NNs
- **3rd May**: RNNs and LSTMs
- **17th May:** Auto Encoders and NLP
- **31st May:** Deep Generative Models: VAEs & GANs, and their implementation with Tensorflow
- **14th June:** Deep RL & Frontiers in Deep Learning



Differences between Statistics and Machine Learning

- Statistics --> inferring values that cannot be directly observed
- ML --> more concerned with prediction

- ML less concerned with guarantees about results
  - Instead, judged by performance on held-out test set

- So far limited theoretical guarantees of DL method performance
  - But will discuss two papers: Universal Approximator Theorem and NFL Theorem

Supervised Learning I

We have a set of N data-points:

$$\mathrm{D}_N = \{(\mathbf{x}_i, y_i)\}$$
$$i \in (0, ..., N)$$

y is some class label

x could be:

- feature data (aka covariates)
- an image
- an entire time series
- some other data structure

We want to learn some mapping from data-space to a predictive distribution over classes

- This is an example of a *Discriminative Model* where we wish to obtain $\mathrm{p}(\mathrm{y}|\mathbf{x})$
- And we are going to be working with *parametric models* - train model = set parameters
- We then want our model to generalise well to new, unseen data
  - ie *good accuracy on the test set*

Supervised Learning II

Supervised learning in ML consists of:

1. Gathering a labelled training dataset, keeping back some test set
2. Choosing a (rich) parametric model, with lots of parameters θ.
3. Writing down an objective function/loss for the problem
4. Tuning the parameters of the model to minimise the loss over the training dataset
   a. Most commonly done by gradient descent of the loss wrt parameters
5. Evaluate model on test set

From Generalised Linear Models to Neural Networks I

Consider generlised linear model:

$$\eta(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x} + b$$
$$p(y|\mathbf{x};\theta) = p(y|g(\eta(\mathbf{x}));\theta)$$
$$\theta = \{\mathbf{w}, b\}$$

η(x) is the linear predictor, unknown w

g(·) in Stats is an *Inverse Link Function*. In ML, an *Activation Function*

$$\sigma(u) = \frac{1}{1+e^{-u}}$$

If g(·) is a *sigmoid function* σ(·), then we obtain logistic regression (binary classification)

We then optimise negative log-likelihood wrt w to get best predictions on training data

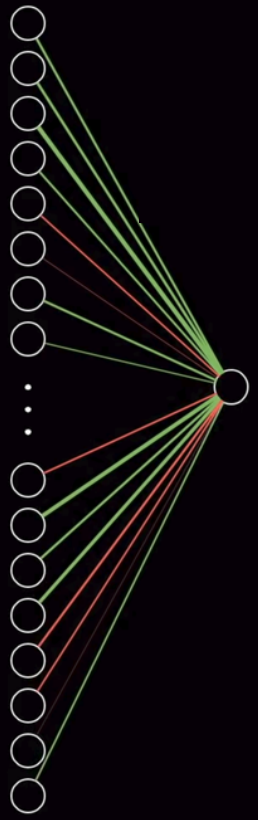
$$\mathcal{L}(\mathbf{x}, y; \theta) = -\log p(y|\mathbf{x};\theta)$$

From Generalised Linear Models to Neural Networks II

$$\mathcal{L}(\mathbf{x}, y) = -\log p(y|\mathbf{x})$$

$$\eta(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} + b$$
$$p(y|\mathbf{x}) = p(y|g(\eta(\mathbf{x})); \theta)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Represent:
--> x as a set of nodes
--> p(y|x) as a single node
--> w as edges from x to y

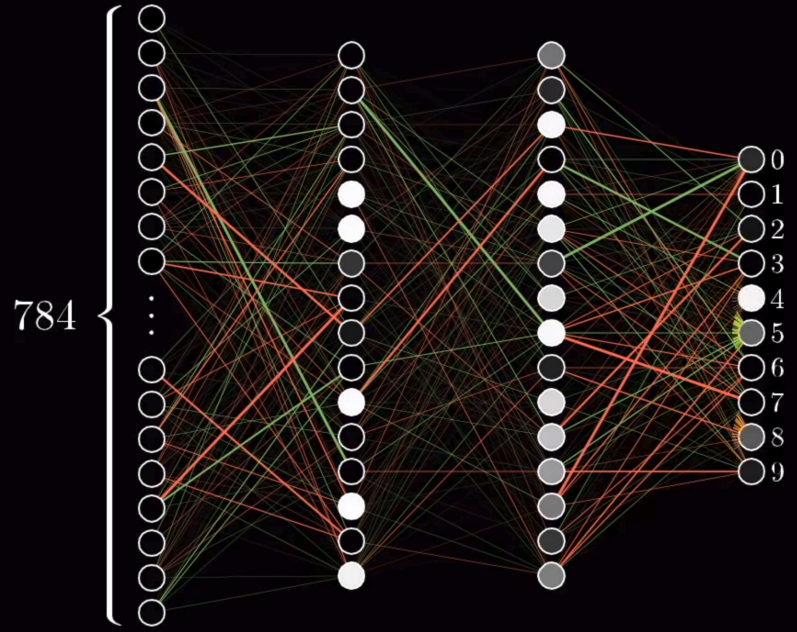
Here green means w>0, red<0

This is logistic regression, AKA a perceptron

If we have multi-class classification
--> w becomes a matrix, ie one row per class
--> have to normalise output to sum to one over all classes

Multilayer Perceptron (MLP)

→ 5



From Generalised Linear Models to Neural Networks III

$$\mathcal{L}(\mathbf{x}, y) = -\log p(y|\mathbf{x}) \qquad \eta(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} + b \qquad p(y|\mathbf{x}) = p(y|g(\eta(\mathbf{x})); \theta) \qquad \sigma(x) = \frac{1}{1 + e^{-x}}$$

We can compose these GLMs on top of each other, to build a hierarchy of GLMs

Interpret the output of inverse link/activation as a next set of covariates

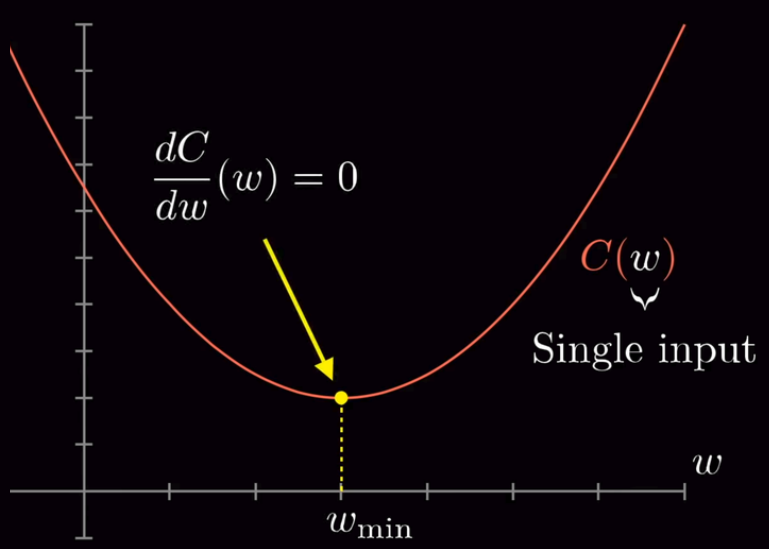
$$\mathbf{x}_{i+1} = g(\eta_{\theta_i}(\mathbf{x}_i)) \qquad \theta = \{\theta_i\}, i \in \{1, ..., \ell\}$$

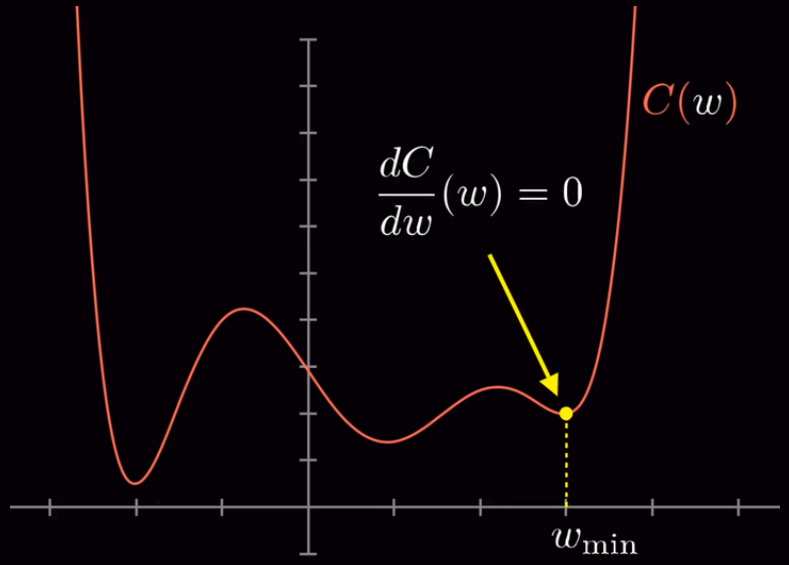
$$p(y|\mathbf{x}) = p(y|g(\eta_{\theta_\ell}(\mathbf{x}_\ell(g(\eta_{\theta_{\ell-1}}(\mathbf{x}_{\ell-1}(g(\eta_{\ell-2}(n_{\ell-1}(x_{\ell-2}...$$

- g(·) essential to prevent collapse to linear case
- How are we going to find $\theta^* = \arg\max_\theta [\mathcal{L}(\mathbf{x}, y; \theta)]$?
  - note - MLE estimate, not Bayesian wrt parameters
- For linear model, Newton's Method works, ie second order optimisation
- But in general, some form of gradient descent wrt parameters: $\theta_{t+1} = \theta_t + \rho(t) \nabla_\theta (\mathcal{L}(\mathbf{x}, y; \theta))|_{\theta=\theta_t}$

Training is Optimisation

We want our model to predict the correct labels on the training set.
Hope it will generalise to the test set!
Our loss is the sum of $\mathcal{L}(\mathbf{x}, y; \theta) = -\log p(y|\mathbf{x}; \theta)$ over our training set
Do gradient descent wrt θ





Training is Optimisation

In Non-Convex Optimisation, the Local Minima you reach will be determined by the objective function you have AND the optimisation routine used

Robbins–Monro and Stochastic Approximation

- $f(\theta)$ is a convex function of which we are trying to find a global optimum θ^* (maximum or minimum)
- $\nabla f(\theta)$ is the gradient of said function
- We generate iterations on θ of the form:

$$\theta_{t+1} = \theta_t - \rho_t \nabla(f(\theta_t))$$

- ρ(t) is a sequence of positive step sizes

Robbins–Monro and Stochastic Approximation

- Convergence to the unique solution at θ^* is guaranteed if:
  - f(θ) is uniformly bounded
  - The sequence of steps satisfies:

$$\sum_{t=0}^{\infty} \rho_t = \infty \quad \sum_{t=0}^{\infty} \rho_t^2 < \infty$$

- Forms the heart of gradient descent !

Example of Optimisation Strategy: Momentum

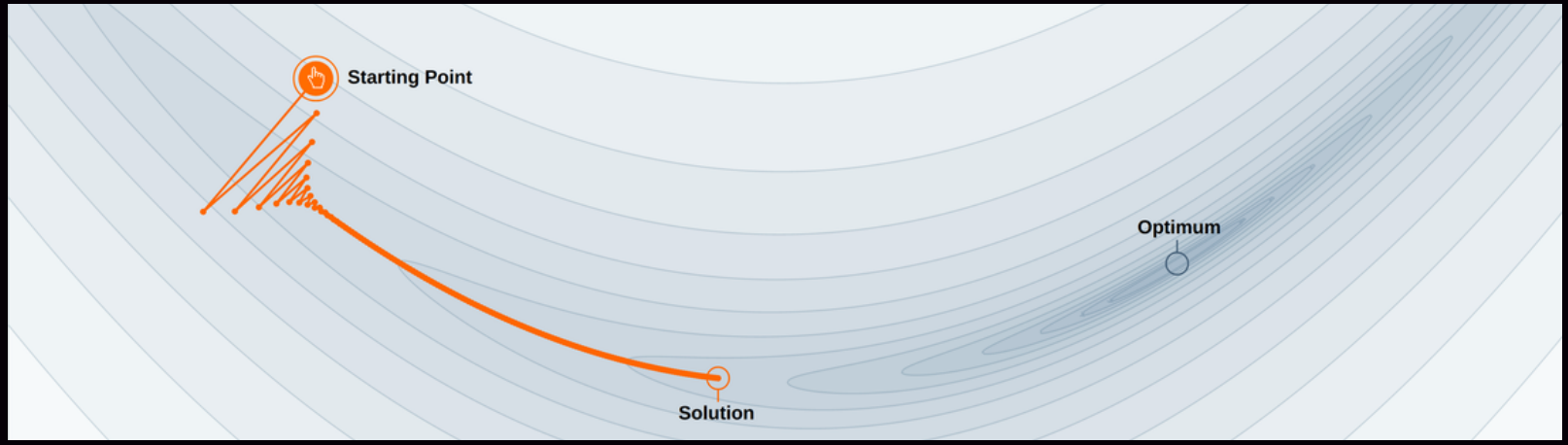
- Although gradient descent has many virtues, speed of convergence is not one of them.
- Momentum gives gradient descent a short-term memory, that limit our chances of converging to a local optimum.

$$z_{i+1} = \beta z_i + \nabla(f(\theta_i))$$
$$\theta_{i+1} = \theta_i - \rho z_{i+1}$$

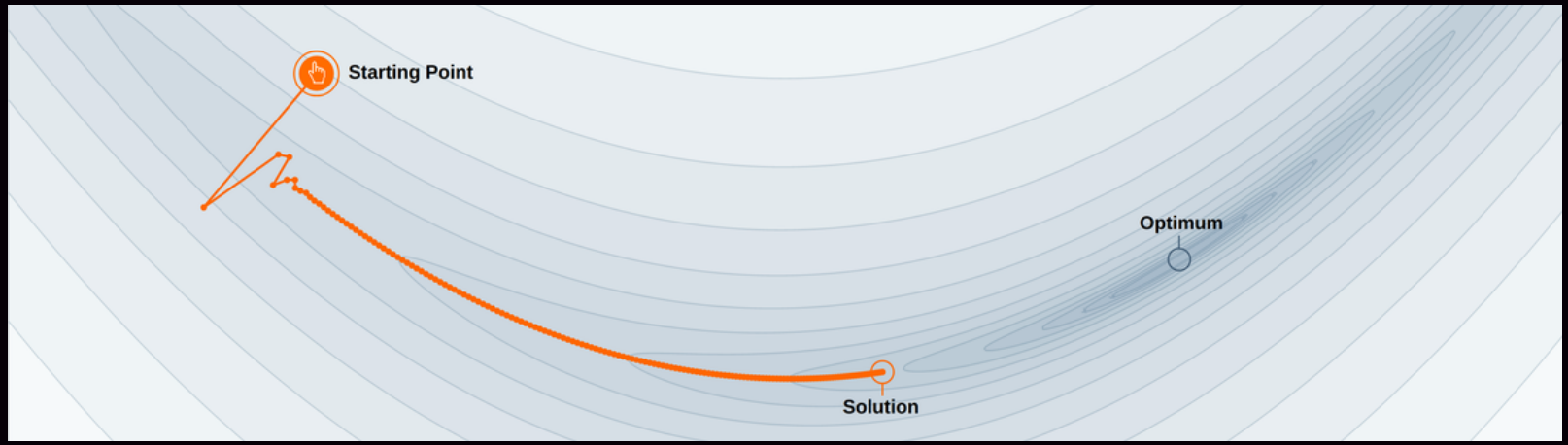
- ρ is the step size for learning.
- When β=0 , we recover gradient descent. As we increase its value, speed of convergence increases.

Example of Optimisation Strategy: Momentum

Without Momentum

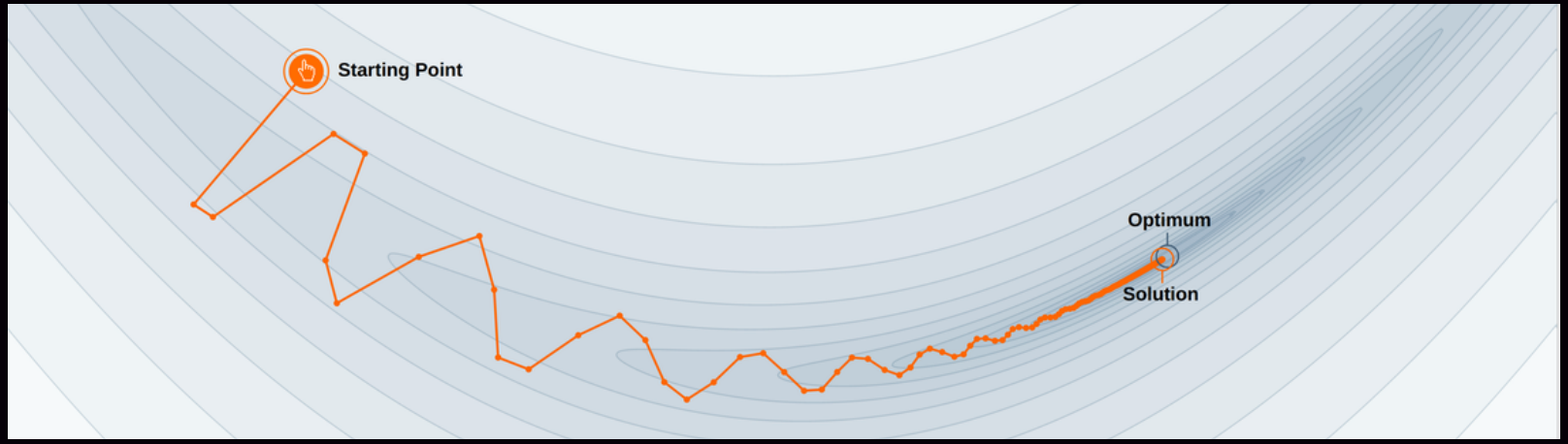


With Momentum

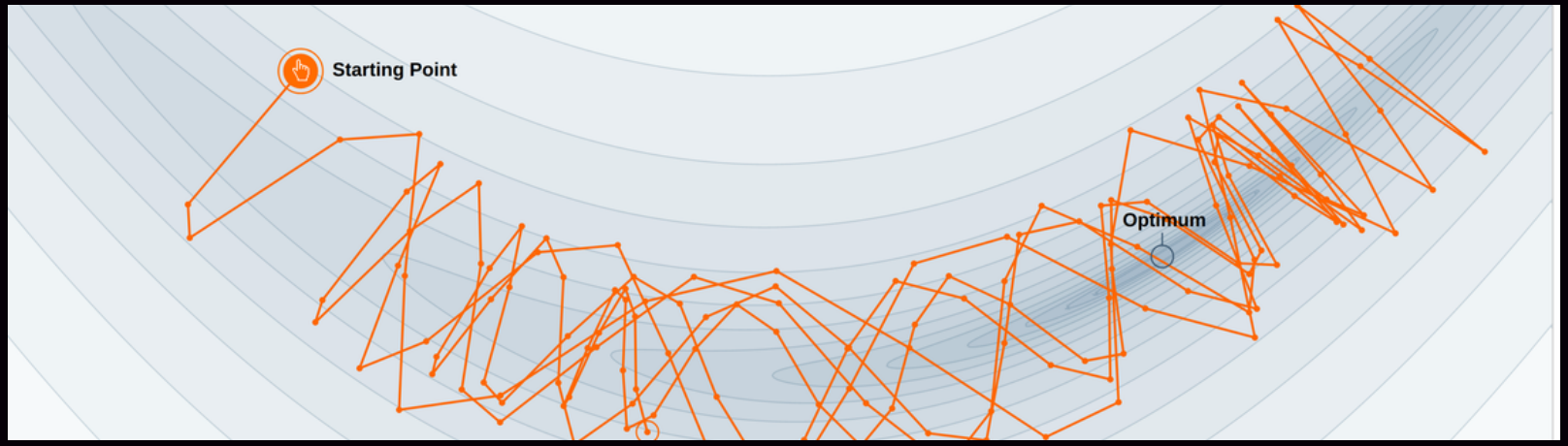


Example of Optimisation Strategy: Momentum

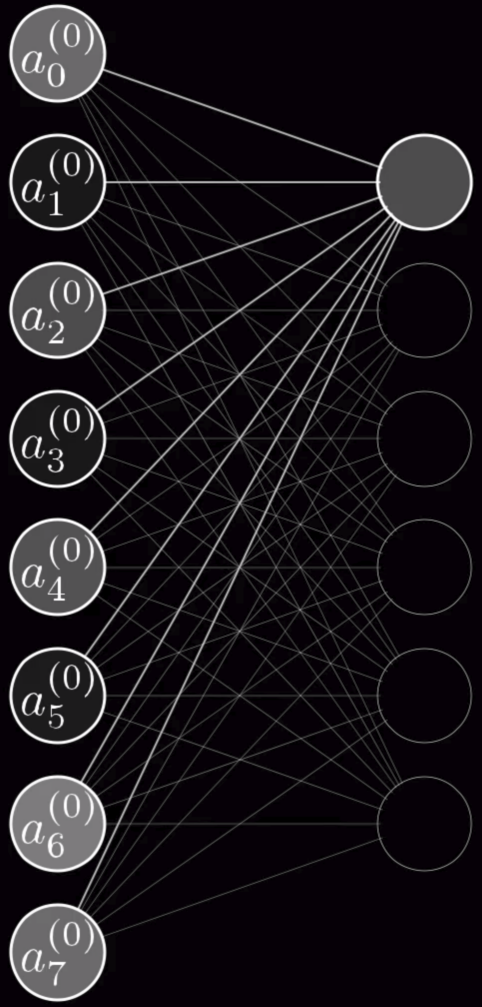
More Momentum



EVEN More Momentum



Activations



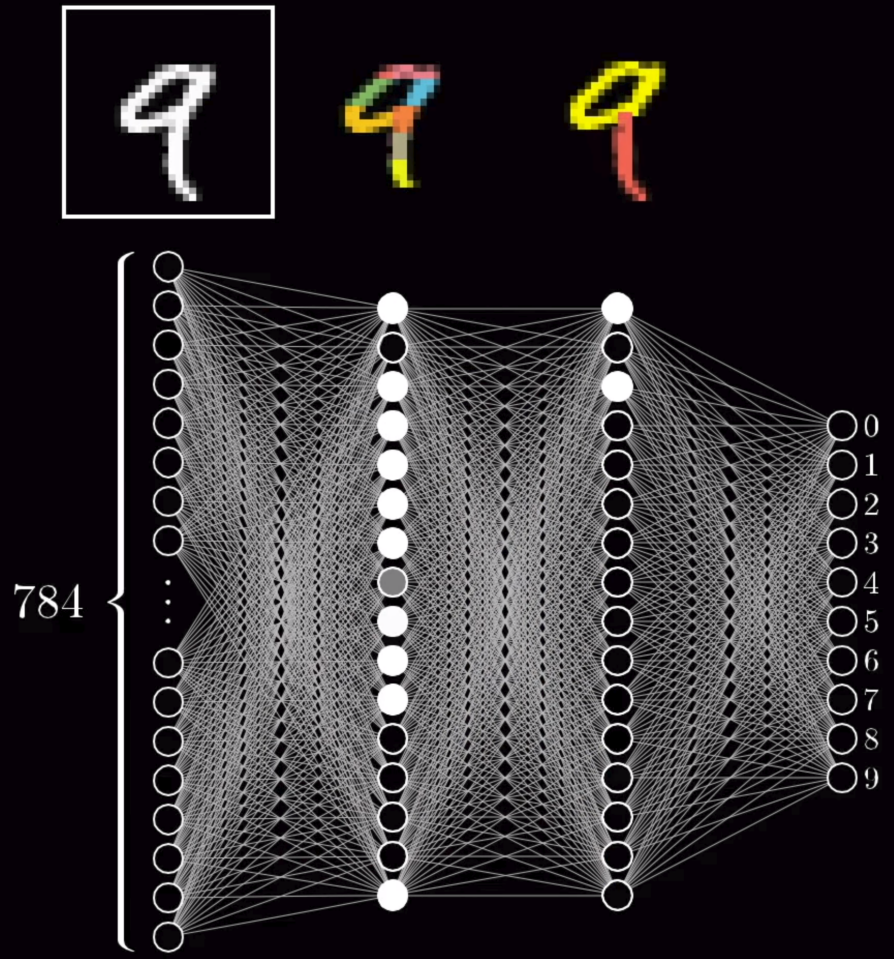
Sigmoid

$$a_0^{(1)} = \sigma\left(w_{0,0}\ a_0^{(0)} + w_{0,1}\ a_1^{(0)} + \cdots + w_{0,n}\ a_n^{(0)} + b_0\right)$$

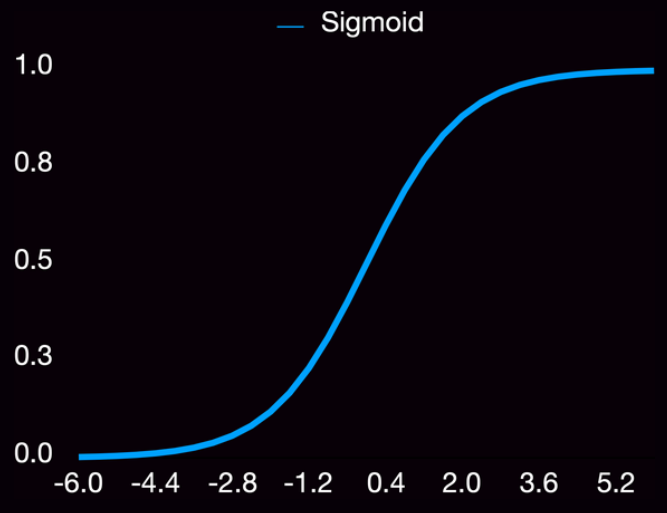
bias

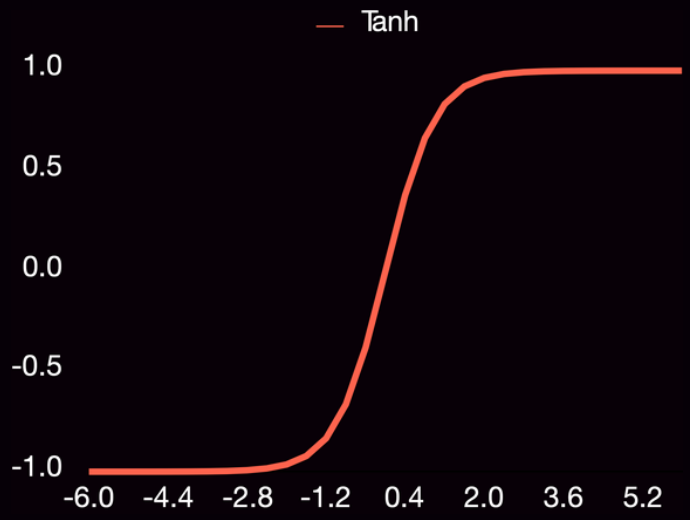
$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$

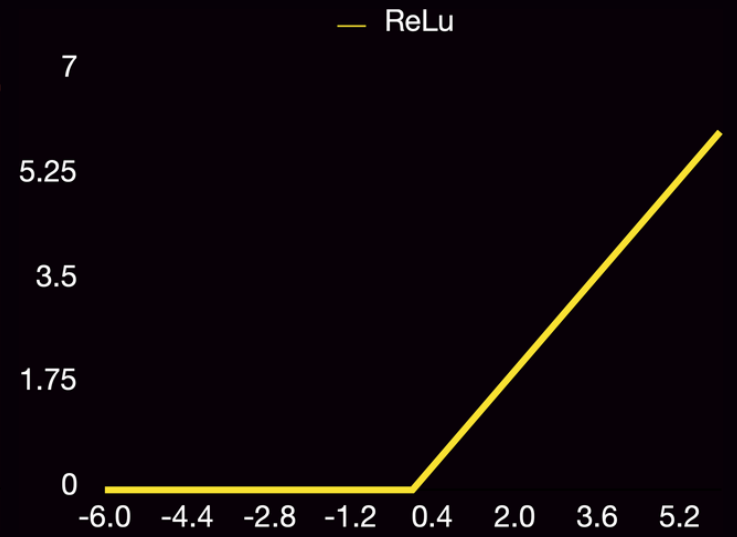
Neural Networks a practical example



Hidden Layer Activation Functions

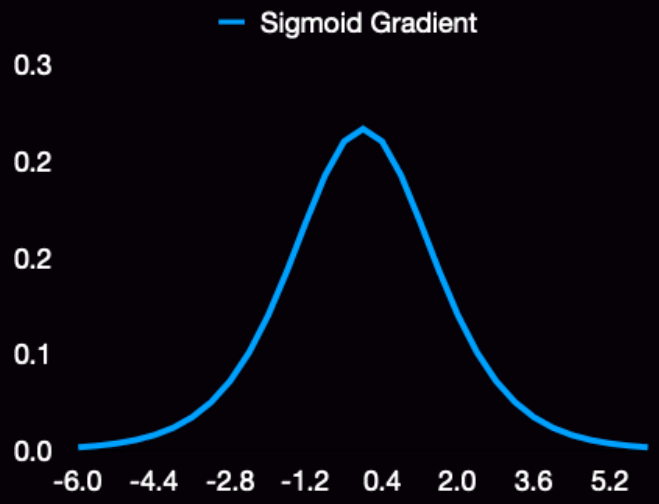


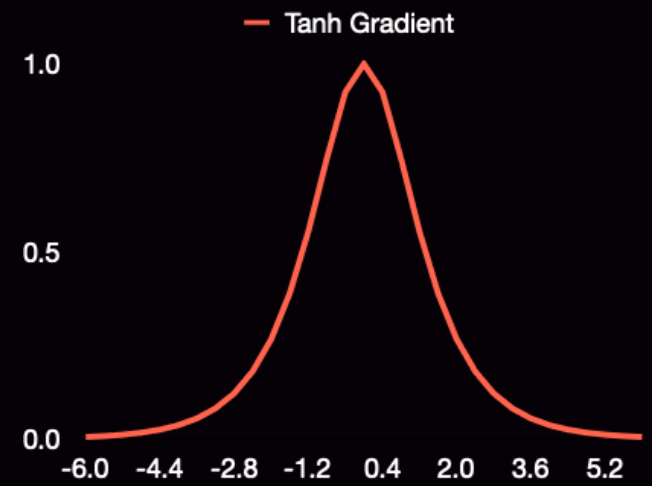


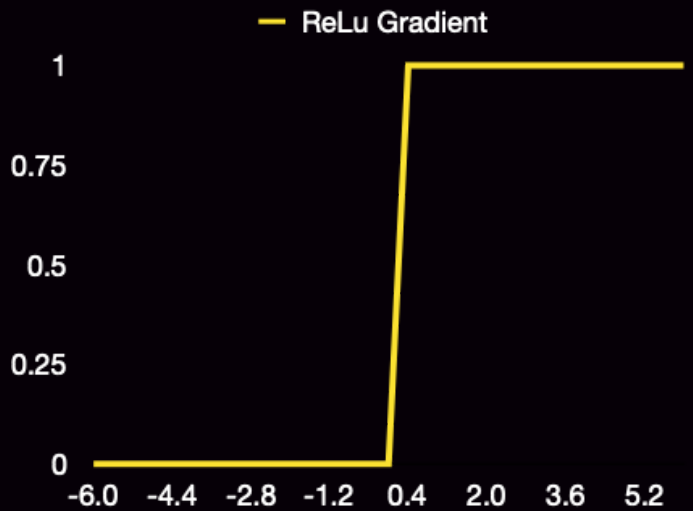


$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad tanh(x) = \frac{2}{1 - e^{-2x}} - 1 \qquad A(x) = max(0, x)$$

Hidden Layer Activation Functions







$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \qquad \tanh'(x) = \frac{2}{e^x + e^{-x}} \qquad A'(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

- **Note**: set discontinuity at 0 to be = 0

Output Layer Activation Functions

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\text{softmax}(x) = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$$

$$\text{softplus}(x) = \log(1 + \exp(x))$$

Paper: Universal Approximation Theorem

Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

1. Introduction

A number of diverse application areas are concerned with the representation of general functions of an n-dimensional real variable, $x \in \mathbb{R}^n$, by finite linear combinations of the form

$$\sum_{j=1}^{N} \alpha_j \sigma(y_j^{\mathrm{T}} x + \theta_j), \tag{1}$$

where $y_j \in \mathbb{R}^n$ and $\alpha_j, \theta \in \mathbb{R}$ are fixed. (y^{T} is the transpose of y so that $y^{\mathrm{T}}x$ is the inner product of y and x.) Here the univariate function σ depends heavily on the context of the application. Our major concern is with so-called sigmoidal σ's:

$$\sigma(t) \to \begin{cases} 1 & \text{as} \quad t \to +\infty, \\ 0 & \text{as} \quad t \to -\infty. \end{cases}$$

Such functions arise naturally in neural network theory as the activation function of a neural node (or *unit* as is becoming the preferred term) [L1], [RHM]. The main result of this paper is a demonstration of the fact that sums of the form (1) are dense in the space of continuous functions on the unit cube if σ is any continuous sigmoidal

* Date received: October 21, 1988. Date revised: February 17, 1989. This research was supported in part by NSF Grant DCR-8619103, ONR Contract N000-86-G-0202 and DOE Grant DE-FG02-85ER25001.

† Center for Supercomputing Research and Development and Department of Electrical and Computer Engineering, University of Illinois, Urbana, Illinois 61801, U.S.A.

Paper: No Free Lunch Theorem

ARTICLE Communicated by Steven Nowlan

The Lack of A Priori Distinctions Between Learning Algorithms

David H. Wolpert
The Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM, 87501, USA

This is the first of two papers that use off-training set (OTS) error to investigate the assumption-free relationship between learning algorithms. This first paper discusses the senses in which there are no a priori distinctions between learning algorithms. (The second paper discusses the senses in which there are such distinctions.) In this first paper it is shown, loosely speaking, that for any two algorithms A and B, there are "as many" targets (or priors over targets) for which A has lower expected OTS error than B as vice versa, for loss functions like zero-one loss. In particular, this is true if A is cross-validation and B is "anti-cross-validation" (choose the learning algorithm with largest cross-validation error). This paper ends with a discussion of the implications of these results for computational learning theory. It is shown that one *cannot* say: if empirical misclassification rate is low, the Vapnik–Chervonenkis dimension of your generalizer is small, and the training set is large, then with high probability your OTS error is small. Other implications for "membership queries" algorithms and "punting" algorithms are also discussed.

> "Even after the observation of the frequent conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience." David Hume, in *A Treatise of Human Nature*, Book I, part 3, Section 12.

1 Introduction

Much of modern supervised learning theory gives the impression that one can deduce something about the efficacy of a particular learning algorithm (generalizer) without the need for any assumptions about the target input–output relationship one is trying to learn with that algorithm. At most, it would appear, to make such a deduction one has to know something about the training set as well as about the learning algorithm.

Consider for example the following quotes from some well-known papers: "Theoretical studies link the generalization error of a learning

Paper: Deep Sparse Rectifier Neural Networks

Deep Sparse Rectifier Neural Networks

Xavier Glorot
DIRO, Université de Montréal
Montréal, QC, Canada
glorotxa@iro.umontreal.ca

Antoine Bordes
Heudiasyc, UMR CNRS 6599
UTC, Compiègne, France
and
DIRO, Université de Montréal
Montréal, QC, Canada
antoine.bordes@hds.utc.fr

Yoshua Bengio
DIRO, Université de Montréal
Montréal, QC, Canada
bengioy@iro.umontreal.ca

Abstract

While logistic sigmoid neurons are more biologically plausible than hyperbolic tangent neurons, the latter work better for training multi-layer neural networks. This paper shows that rectifying neurons are an even better model of biological neurons and yield equal or better performance than hyperbolic tangent networks in spite of the hard non-linearity and non-differentiability at zero, creating sparse representations with true zeros, which seem remarkably suitable for naturally sparse data. Even though they can take advantage of semi-supervised setups with extra-unlabeled data, deep rectifier networks can reach their best performance without requiring any unsupervised pre-training on purely supervised tasks with large labeled datasets. Hence, these results can be seen as a new milestone in the attempts at understanding the difficulty in training deep but purely supervised neural networks, and closing the performance gap between neural networks learnt with and without unsupervised pre-training.

1 Introduction

Many differences exist between the neural network models used by machine learning researchers and those used by computational neuroscientists. This is in part because the objective of the former is to obtain computationally efficient learners, that generalize well to new examples, whereas the objective of the latter is to abstract out neuroscientific data while obtaining explanations of the principles involved, providing predictions and guidance for future biological experiments. Areas where both objectives coincide are therefore particularly worthy of investigation, pointing towards computationally motivated principles of operation in the brain that can also enhance research in artificial intelligence. In this paper we show that two common gaps between computational neuroscience models and machine learning neural network models can be bridged by using the following linear by part activation : $\max(0, x)$, called the rectifier (or hinge) activation function. Experimental results will show engaging training behavior of this activation function, especially for deep architectures (see Bengio (2009) for a review), i.e., where the number of hidden layers in the neural network is 3 or more.

Recent theoretical and empirical work in statistical machine learning has demonstrated the importance of learning algorithms for deep architectures. This is in part inspired by observations of the mammalian visual cortex, which consists of a chain of processing elements, each of which is associated with a different representation of the raw visual input. This is particularly clear in the primate visual system (Serre et al., 2007), with its sequence of processing stages: detection of edges, primitive shapes, and moving up to gradually more complex visual shapes. Interestingly, it was found that the features learned in deep architectures resemble those observed in the first two of these stages (in areas V1 and V2 of visual cortex) (Lee et al., 2008), and that they become increasingly invariant to factors of variation (such as camera movement) in higher layers (Goodfellow et al., 2009).

Appearing in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

Acknowledgments

- Grant Sanderson for figures
- Gabriel Goh for the momentum simulations
- Chris Holmes and Gil McVean for arranging this seminar series