Imperial College London

Introduction to Neural Networks and Deep Learning

Deep Learning Course

Kevin Webster, Pierre Harvey Richemond

Course lecturers

Kevin Webster

- PhD from Imperial College in 2003
- Research interests in Deep Learning, Machine Learning & Dynamical Systems
- Previously Head of Research at music AI startup Jukedeck
- Co-founded ML consultancy FeedForward (feedforwardai.com) in November 2017

Pierre Richemond

- Researching his PhD in RL at the Data Science Institute at Imperial
- Helps to run the Deep Learning Network (www.dlnetwork.org)
- Studied electrical engineering, probability theory and stochastic processes and business management at ENST, Universite Paris VI - Ecole Polytechnique and HEC

Course information

Lecture times and location

Every Wednesday 4-6pm in Huxley 144 until 5th December

Plus Monday 15th October 3-5pm in Huxley 145

Programming/assignment drop in session

Every Tuesday 10-11am in Mathematics Learning Centre

Course website

https://www.deeplearningmathematics.com

Discussion forum

Facebook group (link on website) for questions / discussions during the course

GitHub repository

Coursework and tutorial notebooks are available on the course repository:

https://github.com/pukkapies/dl-imperial-maths

Contact

kevin.webster@imperial.ac.uk & p.richemond17@imperial.ac.uk

@kn_webster | @KloudStrife | @DLImperialMaths

Course aims

- Understand the background, goals and intuition behind Deep Learning, and the main developments of the field over the last 5 years
- Know the principles and design of the main types of neural network architecture
- Have an overview of some of the outstanding problems in Deep Learning
- Familiarise with Tensorflow and PyTorch and be able to build and train several types of neural network

Deep learning frameworks

(Tentative) syllabus (1/3)

Week 1. **Introduction to deep learning.** Overview and basic concepts. Supervised and unsupervised learning. Underfitting and overfitting. Typical problem tasks, example applications.

Week 2. **Neural networks optimisation.** The backpropagation method. Neural network optimisers: SGD, (Nesterov) momentum, Adagrad, RMSProp, Adadelta, Adam. Network initialisation strategies. Batch normalisation.

Week 3. **Convolutional neural networks.** Building blocks of CNNs: convolution, pooling operations, padding, strides, transposed convolutions. Capsule networks. CNN architectures: AlexNet, VGG, GoogLeNet/Inception, ResNet. Style transfer.

(Tentative) syllabus (2/3)

Week 4. **Introduction to reinforcement learning - part 1.** Key concepts: environment, agent, state, action, reward. Markov decision process. Bellman equation, dynamic programming. Q-learning.

Week 5. **Introduction to reinforcement learning - part 2.** Policy gradients. REINFORCE algorithm. Actor-critic paradigm. Entropy regularisation. TRPO, PPO.

Week 6. **Sequence models.** Recurrent neural networks, LSTM, GRU units. Bi-directional RNNs. Memory-augmented networks. Autoregressive models. Attention mechanisms.

(Tentative) syllabus (3/3)

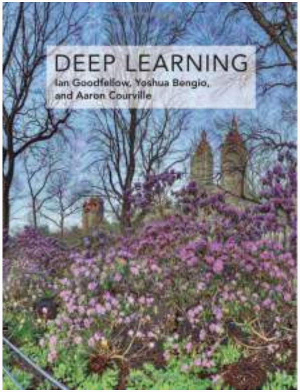
Week 7. **Generative Adversarial Networks (GANs).** Generator and discriminator networks. Mode collapse. Wasserstein GAN, Kantorovich-Rubinstein duality. Spectral normalisation.

Week 8. **Variational autoencoders (VAEs).** Encoder/decoder architecture. Variational inference and evidence lower bound (ELBO). Reparameterisation trick. IWAE, β-VAE.

Week 9. **Normalising flows.** Probability density change of variables. Autoregressive models as normalising flows. IAF, MAF, MADE, NICE, RealNVP, Glow, FFJORD.

Week 10. **Theories of deep learning.** Optimal transport, mean-field gradient flow, random matrix theory, spectral norm bounds.

Deep Learning book

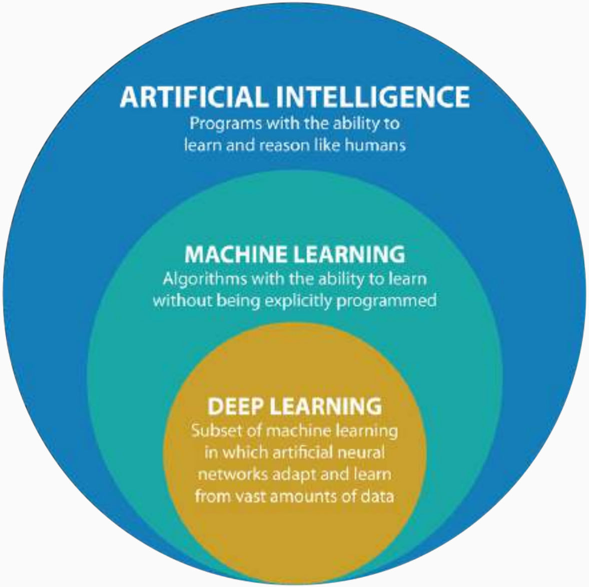


Deep Learning, Ian Goodfellow, Yoshua Bengio & Aaron Courville, 2016. MIT Press, http://www.deeplearningbook.org.

Outline

Outline

AI, ML and Deep Learning


ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

Birth of Artificial Intelligence

‘The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.’

- Proposal from the Dartmouth Conference, 1956

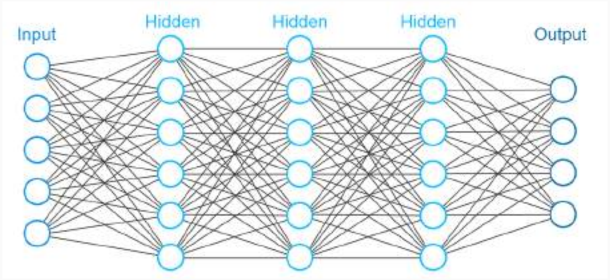
Machine Learning

‘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.’

- Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2.

Deep Learning

- Allow computers to learn from experience
- Avoids hand engineered knowledge
- Hierarchy of concepts
- Concepts are built on top of each other in layers



Deep Learning aims

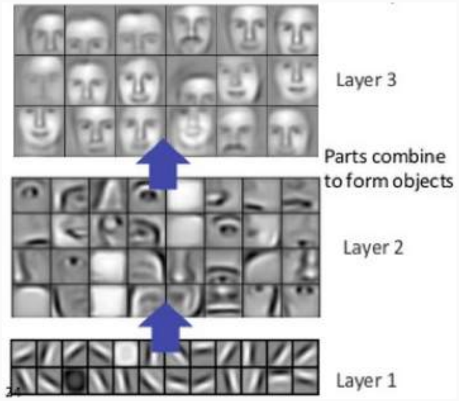
- AI applications aim to automate routine tasks, perform speech recognition, image analysis/understanding, medical diagnoses etc
- Early AI: describe problems with list of formal, mathematical rules.
- Real problem: solve tasks that are easy for people, intuitive, eg face recognition
- Machine Learning systems acquire their own knowledge from analysing data
- Representation learning aims to discover the best representations or features of the data
- Deep Learning layers distributed representations that encode high-level concepts with increasing depth

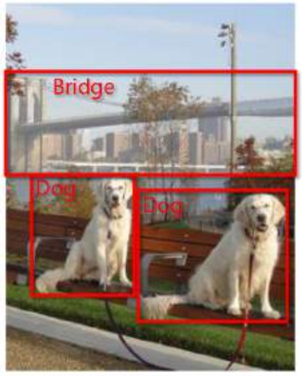
Historical trends in Deep Learning

- Deep Learning is not new! It has been rebranded from cybernetics, connectionism and artificial neural networks
- Decline in popularity from mid-1990s to 2006 when Geoffrey Hinton introduced the deep belief network
- Deep Learning is no longer guided by neuroscience and modelling the brain
- Resurgence of interest in Deep Learning as amount of available training data has increased
- Models have grown in size (and accuracy) as computer hardware and software has improved

Deep Learning applications

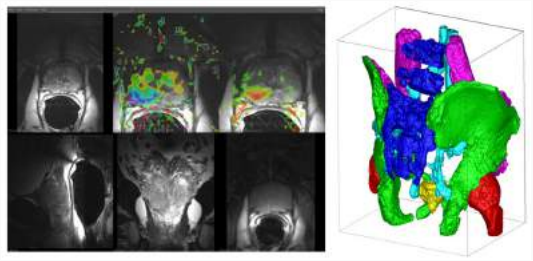
Image recognition

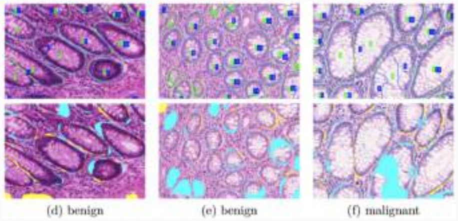




Deep Learning applications

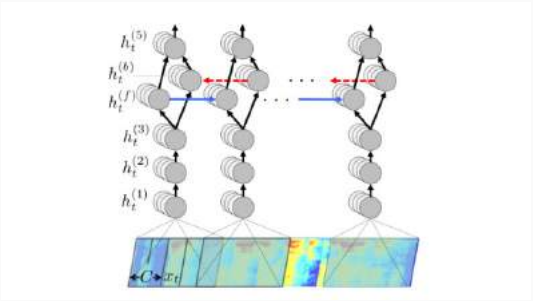
Healthcare

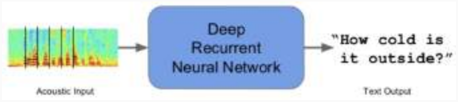




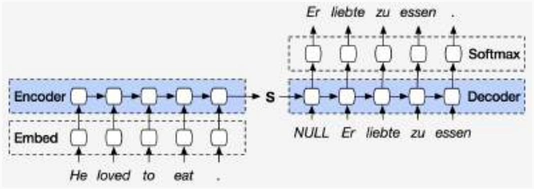
Deep Learning applications

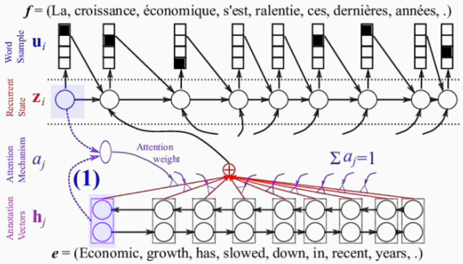
Speech recognition





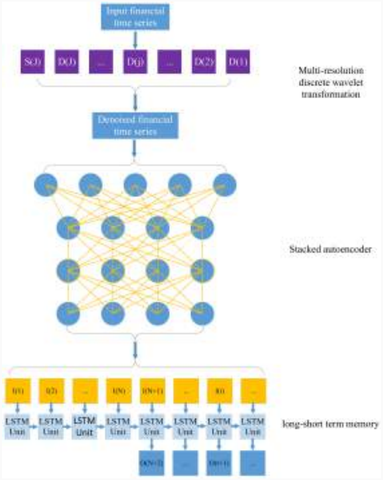
Machine translation

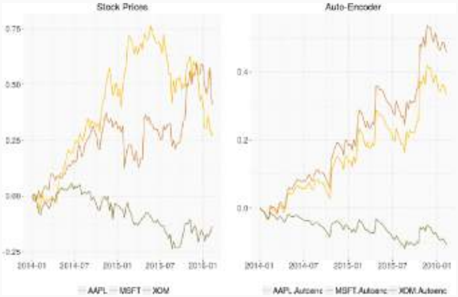




Deep Learning applications

Finance





Deep Learning applications

Image generation

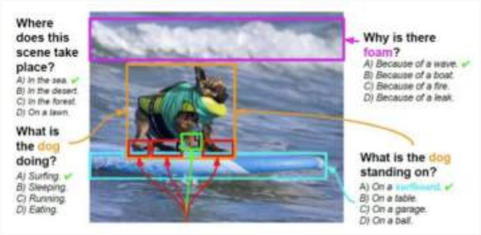
Image generation

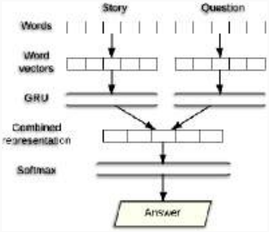
[video]

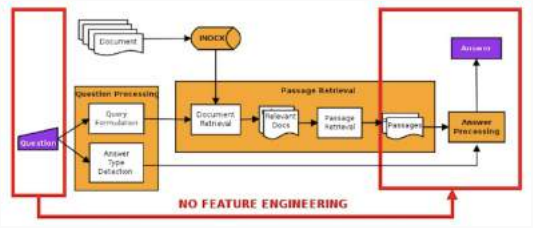
Deep Learning applications

Image generation

Question answering

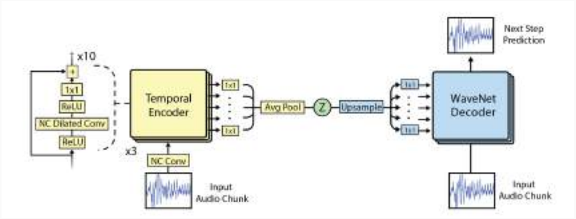


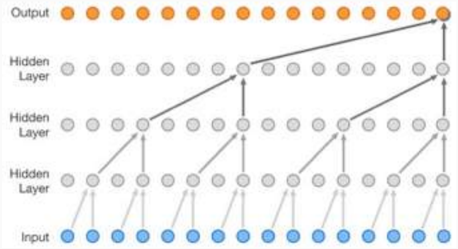




Deep Learning applications

Audio synthesis





[audio examples]

Outline

Mathematical requirements

- Linear algebra
  - Eigendecomposition
  - Singular Value Decomposition
- Probability theory
  - Marginal, joint & conditional probabilities
  - Bernoulli, categorical, Gaussian distributions...
  - Bayes' rule
- Information theory
  - Entropy & cross-entropy
  - Kullback-Leibler (KL) divergence
- Elementary calculus and optimization

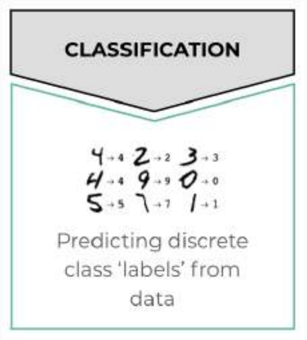
Machine Learning

Recall Mitchell's definition of a learning algorithm:

'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.'

What kinds of tasks T are machine learning algorithms suited to?

Machine Learning - task T



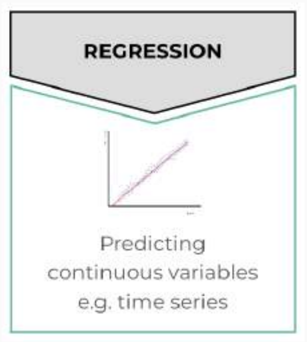
The data is generated according to the joint distribution $p(\mathbf{x}, \mathcal{C})$, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathcal{C} \in \{1, \ldots, k\}$.

The aim is to learn a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$ that classifies the data.

Typically split into a model of the conditional distribution $p(C|\mathbf{x})$ and some decision rule. This is a **discriminative** task.

Examples: object recognition, sentiment analysis.

Machine Learning - task T



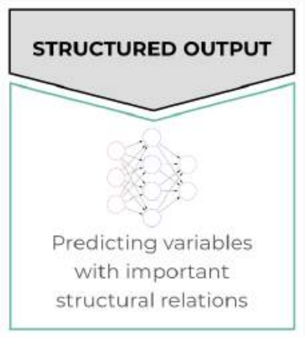
The data is generated according to the joint distribution $p(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}$. The data could be sequential in nature.

This task is similar to regression, but the output type of the model is different.

Could learn a deterministic mapping, or train a model to output the parameters of a distribution.

Examples: insurance claims, financial markets.

Machine Learning - task T



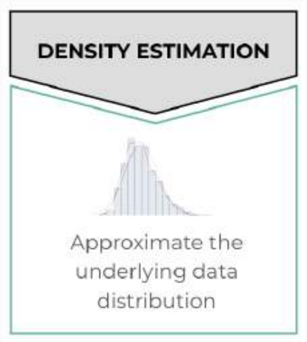
The data resides in a highly structured space, such as parse trees, graphs, or the space of valid sentences for a translation task.

Models typically output vector representations that encode structural objects.

As before, the model output could be deterministic or it could be a distribution over possible outputs.

Examples: OCR, transcription, language translation.

Machine Learning - task T



The data is generated according to $p(\mathbf{x}, \mathbf{y})$ or just $p(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}$.

The task is to learn the underlying (joint) probability density function that generated the data.

This **generative** modelling approach can be useful for other tasks, as well as generating new data examples similar to the training set.

Examples: speech synthesis, protein design.

Machine Learning - performance P

- Need a quantitative measure to train and evaluate the model
- Choosing a performance measure is not always trivial
- Measure depends on task and model output
    - Classification accuracy
    - Least squares error
    - Cross-entropy loss
    - Discounted sum of rewards
    - Mean average precision
    - ...
- Typically evaluate model performance on a held-out test set
- In addition, a separate validation set may be used

Machine Learning - experience E

Supervised and unsupervised learning are two broad categorisations of the type of experience, or data, that models are provided with.

- **Supervised** learning algorithms are provided with a training set of data features and associated labels
- **Unsupervised** learning algorithms are provided with a training set of data features, and are tasked with finding useful representations or structure in the data

Machine Learning - experience E

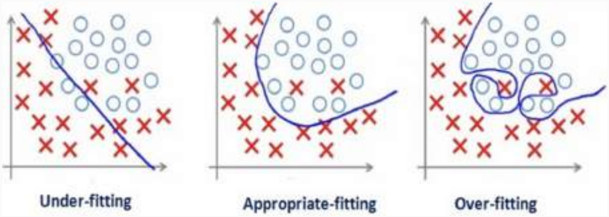
Some machine learning algorithms are provided with an interactive experience with which to learn from.

Reinforcement learning algorithms interact with an environment, and are provided with features of the environment, as well as a reward signal. The models must use this feedback from the environment to learn the optimal way to interact with the environment.

Machine Learning - generalization

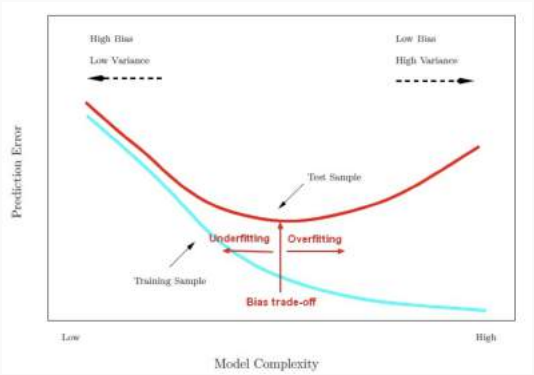
The ability to perform well on previously unseen inputs is **generalization**.

- Generalization is usually measured on a test set (separate to training set)
- Underlying assumption is that training set and test set are i.i.d.
- Parameters of the model are adjusted according to the training set
- Model **overfits** if it performs poorly on the test set
- Model **underfits** if the training error is too large



Machine Learning - generalization

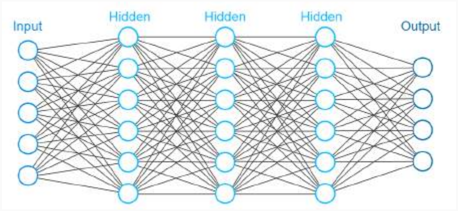
Overfitting/underfitting can be controlled through the model complexity and regularisation.



Statistical learning theory provides some estimates on generalization errors for certain classes of models.

Outline

Feedforward network / multilayer perceptron



Neural network with N layers $h_1, \dots, h_N$, where $h_i \in \mathbb{R}^{n_i}$. Input $\mathbf{x} = h_1$ and output $\mathbf{y} = h_N$. For $i = 1, \dots, N-1$,

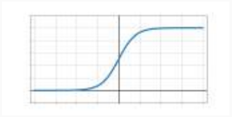
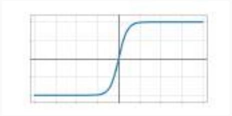
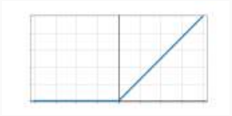
$$\hat{h}_{i+1} = W^{(i)} h_i + b^{(i)}, \qquad \text{(pre-activation)}$$

where $W^{(i)} \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b^{(i)} \in \mathbb{R}^{n_{i+1}}$.

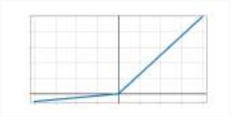
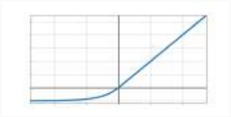
$$h_{i+1} = \sigma(\hat{h}_{i+1}), \qquad \text{(post-activation)}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is an **activation function** that is applied element-wise.

Activation functions

Activation function	Plot	Equation
sigmoid		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$
tanh		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
ReLU		$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$

Activation functions

Activation function	Plot	Equation
leaky ReLU		$f(x) = \begin{cases} \epsilon x, & x < 0 \\ x, & x \geq 0 \end{cases}$
ELU		$f(x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$

Output layers

Frequently used output activations / layers:

- **Sigmoid output layer** is useful to predict probabilities as the range is in $(0, 1)$.
- **Softplus output layer** is useful to predict e.g. Gaussian variance parameters, as the range is $(0, \infty)$:

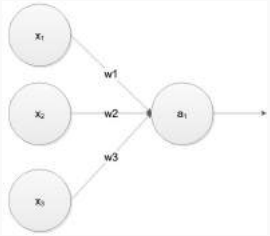
$$f(x) = \ln(1 + e^x)$$

- **Softmax output layer** is often used to predict parameters of a categorical distribution:

$$P(\text{Category } j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}},$$

where z_k $(k \in \{1, \ldots, K\})$ are the pre-activations.

Example: logistic regression



Logistic regression can be viewed as a simple neural network. The model prediction is given by

$$y = \sigma(\mathbf{w}^T \mathbf{x} + b),$$

where the inputs $\mathbf{x} \in \mathbb{R}^N$, the weights $\mathbf{w} \in \mathbb{R}^N$, bias $b \in \mathbb{R}$, σ is the sigmoid function and the output $y \in (0, 1)$.

Loss functions

Loss functions used in deep learning are often attempting to maximise log-likelihood. Typical per-example loss functions are:

- **Cross-entropy loss**. Frequently used in classification, with M mutually exclusive classes:

$$H_{\mathbf{y}'}(\mathbf{y}) := -\sum_{i=1}^{M} y_i' \log(y_i)$$

- **Mean squared error**. Frequently used in regression tasks, e.g. in predicting a d-dimensional output:

$$S := \frac{1}{d}\sum_{i=1}^{d}(y_i' - y_i)^2$$

Other loss functions are used for specific network architectures such as the VAE or GAN (covered in later lectures).

Regularisation

As mentioned earlier, regularisation is important to avoid overfitting. There are several approaches to regularisation:

- Model complexity
- Weight decay
- Patience/early stopping
- Dropout
- Weight sharing
- Ensemble predictions
- Dataset augmentation
- Noise robustness
- Multitask learning
- Adversarial training

Stochastic gradient descent

Most Deep Learning models are trained with (a variant of) stochastic gradient descent.

The cost function usually decomposes as a sum over the training examples in the training set:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are the parameters of the model, $(\mathbf{x}^{(i)}, y^{(i)})_{i=1}^{m}$ is the (labelled) training set, and L is the per-example loss.

Stochastic gradient descent

Gradient descent involves computing

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}),$$

which can be prohibitively expensive for large datasets.

Stochastic gradient descent approximates the expected gradient by sampling a **minibatch** of data examples $(\mathbf{x}^{(n_i)}, y^{(n_i)})_{i=1}^{m'}$, $m' \ll m$.

The estimated gradient is then

$$\mathbf{g} = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}^{(n_i)}, y^{(n_i)}, \boldsymbol{\theta}),$$

Stochastic gradient descent

Stochastic gradient descent then uses the estimated gradient to adjust the parameters:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \mathbf{g},$$

where ϵ is the learning rate.

- SGD makes efficient use of the dataset when training deep learning models
- The cost per SGD update does not depend on the training size m
- Provides a scalable way of training nonlinear models on large datasets

References i

Bishop, C. M. (2006).
Pattern Recognition and Machine Learning.
Springer.

Dwivedi, P. (2018).
Nlp – building a question answering model.
https://towardsdatascience.com/nlp-building-a-question-answering-model-ed0529a68c54.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
http://www.deeplearningbook.org.

References ii

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009).
The elements of statistical learning: data mining, inference, and prediction, 2nd Edition.
Springer series in statistics. Springer.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018).
Progressive growing of gans for improved quality, stability, and variation – official tensorflow implementation of the iclr 2018 paper.
https://github.com/tkarras/progressive_growing_of_gans.

Mitchell, T. M. (1997).
Machine learning.
McGraw Hill series in computer science. McGraw-Hill.

References iii

Uszkoreit, J. (2017).
Transformer: A novel neural network architecture for language understanding.
https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html.

van den Oord, A., Dieleman, S., and Zen, H. (2016).
Wavenet: A generative model for raw audio.
https://deepmind.com/blog/wavenet-generative-model-raw-audio/.

Wallner, E. (2017).
Colorizing b&w photos with neural networks.
https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/.

References iv

Zhang, M. (2018).
Biggan: A new state of the art in image synthesis.
https://medium.com/syncedreview/
biggan-a-new-state-of-the-art-in-image-synthesis-cf2ec56940