

TTIC 31230, Fundamentals of Deep Learning, Winter 2019

David McAllester

The Fundamental Equations of Deep Learning

Early History

1943: McCullock and Pitts introduced the linear threshold “neuron”.

1962: Rosenblatt applies a “Hebbian” learning rule. Novikoff proved the perceptron convergence theorem.

1969: Minsky and Papert publish the book *Perceptrons*.

The Perceptrons book greatly discourages work in artificial neural networks. Symbolic methods dominate AI research through the 1970s.

80s Renaissance

1980: Fukushima introduces the neocognitron (a form of CNN)

1984: Valiant defines PAC learnability and stimulates learning theory. Wins Turing Award in 2010.

1985: Hinton and Sejnowski introduce the Boltzman machine

1986: Rumelhart, Hinton and Williams demonstrate empirical success with backpropagation (itself dating back to 1961).

90s and 00s: Research In the Shadows

1997: Schmidhuber et al. introduce LSTMs

1998: LeCunn introduces convolutional neural networks (CNNs) (LeNet).

2003: Bengio introduces neural language modeling.

Current Era

2012: Alexnet dominates the Imagenet computer vision challenge.

Google speech recognition converts to deep learning.

Both developments come out of Hinton's group in Toronto.

2013: Refinement of AlexNet continues to dramatically improve computer vision.

2014: Neural machine translation appears (Seq2Seq models).
Variational auto-encoders (VAEs) appear.

Graph networks for molecular property prediction appear.

Dramatic improvement in computer vision and speech recognition continues.

Current Era

2015: Google converts to neural machine translation leading to dramatic improvements.

ResNet appears. This makes yet another dramatic improvement in computer vision.

Generative Adversarial Networks (GANs) appear.

2016: Alphago defeats Lee Sedol.

Current Era

2017: AlphaZero learns both go and chess at super-human levels in a matter of hours entirely from self-play and advances computer go far beyond human abilities.

Unsupervised machine translation is demonstrated.

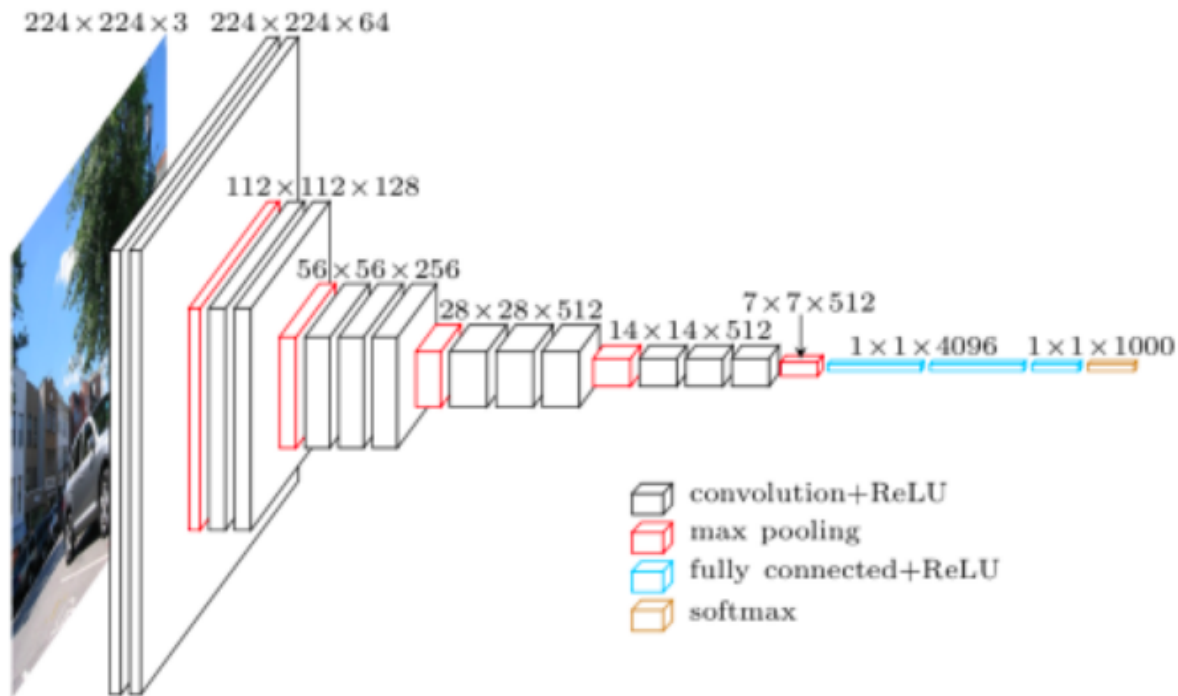
Progressive GANs.

2018: Unsupervised pre-training significantly improves a broad range of NLP tasks including question answering (but dialogue remains unsolved).

AlphaFold revolutionizes protein structure prediction.

What is a Deep Network?

VGG, Zisserman, 2014



Davi Frossard

138 Million Parameters

What is a Deep Network?

We assume some set \mathcal{X} of possible inputs, some set \mathcal{Y} of possible outputs, and a parameter vector $\Phi \in \mathbb{R}^d$.

For $\Phi \in \mathbb{R}^d$ and $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ a deep network computes a probability $P_\Phi(y|x)$.

The Fundamental Equation of Deep Learning

We assume a “population” probability distribution Pop on pairs (x, y) .

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} E_{(x,y) \sim \text{Pop}} - \ln P_{\Phi}(y|x)$$

This loss function $\mathcal{L}(x, y, \Phi) = -\ln P_{\Phi}(y|x)$ is called **cross entropy loss**.

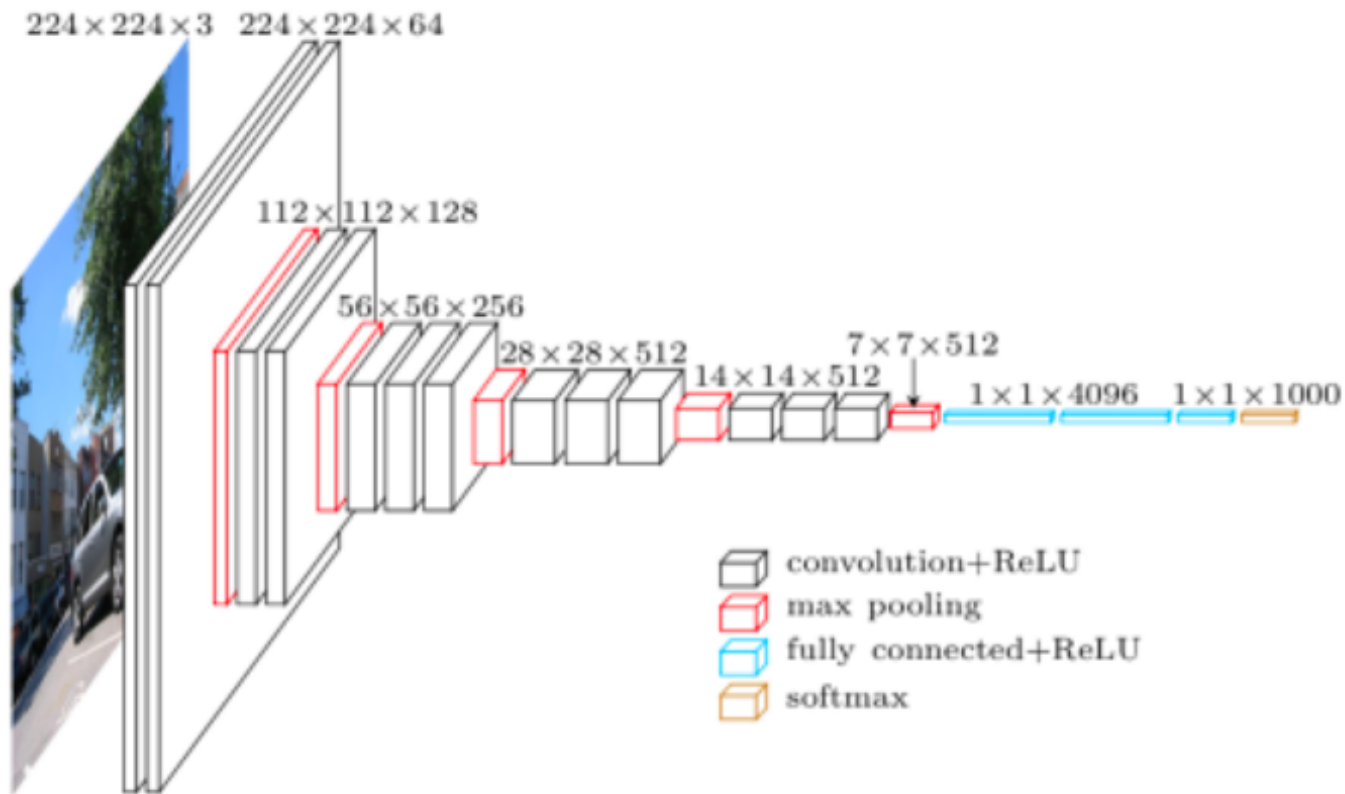
A Second Fundamental Equation

Softmax: Converting Scores to Probabilities

We start from a “score” function $s_{\Phi}(y|x) \in \mathbb{R}$.

$$\begin{aligned} P_{\Phi}(y|x) &= \frac{1}{Z} e^{s_{\Phi}(y|x)}; \quad Z = \sum_y e^{s_{\Phi}(y|x)} \\ &= \operatorname{softmax}_y s_{\Phi}(y|x) \end{aligned}$$

Note the Final Softmax Layer



Davi Frossard

How Many Possibilities

We have $y \in \mathcal{Y}$ where \mathcal{Y} is some set of “possibilities”.

Binary: $Y = \{-1, 1\}$

Multiclass: $Y = \{y_1, \dots, y_k\}$ k manageable.

Structured: y is a “structured object” like a sentence. Here $|Y|$ is unmanageable.

Binary Classification

We have a population distribution over (x, y) with $y \in \{-1, 1\}$.

We compute a single score $s_\Phi(x)$ where

for $s_\Phi(x) \geq 0$ predict $y = 1$

for $s_\Phi(x) < 0$ predict $y = -1$

Softmax for Binary Classification

$$\begin{aligned} P_{\Phi}(y|x) &= \frac{1}{Z} e^{ys(x)} \\ &= \frac{e^{ys(x)}}{e^{ys(x)} + e^{-ys(x)}} \\ &= \frac{1}{1 + e^{-2ys(x)}} \\ &= \frac{1}{1 + e^{-m(y)}} \quad m(y|x) = 2ys(x) \text{ is the margin} \end{aligned}$$

Logistic Regression for Binary Classification

$$\Phi^* = \operatorname{argmin}_{\Phi} E_{(x,y) \sim P_{\text{op}}} \mathcal{L}(x, y, \Phi)$$

$$= \operatorname{argmin}_{\Phi} E_{(x,y) \sim P_{\text{op}}} -\ln P_{\Phi}(y|x)$$

$$= \operatorname{argmin}_{\Phi} E_{(x,y) \sim P_{\text{op}}} \ln \left(1 + e^{-m(y|x)} \right)$$

$$\ln \left(1 + e^{-m(y|x)} \right) \approx 0 \quad \text{for } m(y|x) \gg 1$$

$$\ln \left(1 + e^{-m(y|x)} \right) \approx -m(y|x) \quad \text{for } -m(y|x) \gg 1$$

Log Loss vs. Hinge Loss (SVM loss)

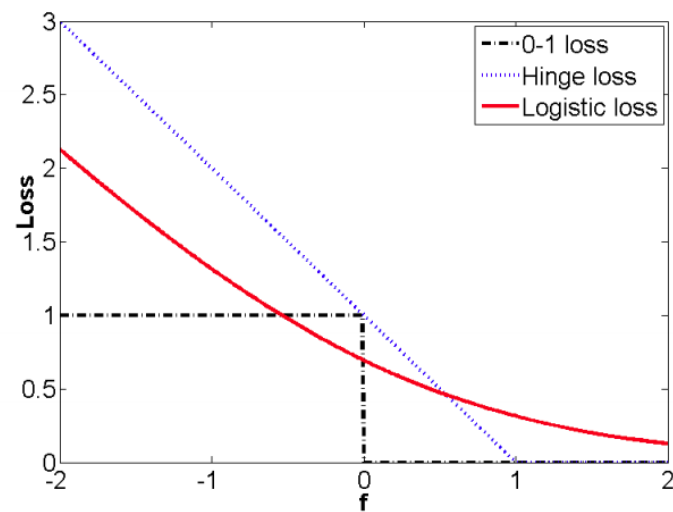


Image Classification (Multiclass Classification)

We have a population distribution over (x, y) with $y \in \{y_1, \dots, y_k\}$.

$$P_{\Phi}(y|x) = \underset{y}{\text{softmax}} \ s_{\Phi}(y|x)$$

$$\begin{aligned} \Phi^* &= \underset{\Phi}{\text{argmin}} \ E_{(x,y) \sim P_{\text{op}}} \mathcal{L}(x, y, \Phi) \\ &= \underset{\Phi}{\text{argmin}} \ E_{(x,y) \sim P_{\text{op}}} \textcolor{red}{- \ln P_{\Phi}(y|x)} \end{aligned}$$

Machine Translation (Structured Labeling)

We have a population of translation pairs (x, y) with $x \in V_x^*$ and $y \in V_y^*$ where V_x and V_y are source and target vocabularies respectively.

$$P_{\Phi}(w_{t+1}|x, w_1, \dots, w_t) = \underset{w \in V_y \cup \langle \text{EOS} \rangle}{\text{softmax}} s_{\Phi}(w \mid x, w_1, \dots, w_t)$$

$$P_{\Phi}(y|x) = \prod_{t=0}^{|y|} P_{\Phi}(y_{t+1} \mid x, y_1, \dots, y_t)$$

$$\begin{aligned} \Phi^* &= \underset{\Phi}{\operatorname{argmin}} E_{(x,y) \sim \text{Pop}} \mathcal{L}(x, y, \Phi) \\ &= \underset{\Phi}{\operatorname{argmin}} E_{(x,y) \sim \text{Pop}} -\ln P_{\Phi}(y|x) \end{aligned}$$

Fuundamental Equation: Unconditional Form

$$\Phi^* = \operatorname{argmin}_{\Phi} E_{y \sim P_{\text{op}}} - \ln P_{\Phi}(y)$$

Entropy of a Distribution

The entropy of a distribution P is defined by

$$H(P) = E_{y \sim P} [-\ln P(y)] \text{ in units of “nats”}$$

$$H_2(P) = E_{y \sim P} [-\log_2 P(y)] \text{ in units of bits}$$

Example: Let Q be a uniform distribution on 256 values.

$$E_{y \sim Q} [-\log_2 Q(y)] = -\log_2 \frac{1}{256} = \log_2 256 = 8 \text{ bits} = 1 \text{ byte}$$

$$1 \text{ nat} = \frac{1}{\ln 2} \text{ bits} \approx 1.44 \text{ bits}$$

The Coding Interpretation of Entropy

We can interpret $H_2(Q)$ as the number of bits required on average to represent items drawn from distribution Q .

We want to use fewer bits for common items.

There exists a representation where, for all y , the number of bits used to represent y is no larger than $-\log_2 y + 1$ (Shannon's source coding theorem).

$$H(Q) = \frac{1}{\ln 2} H_2(Q) \approx 1.44 H_2(Q)$$

Cross Entropy

Let P and Q be two distribution on the same set.

$$H(P, Q) = E_{y \sim P} - \ln Q(y)$$

$$\Phi^* = \operatorname{argmin}_{\Phi} H(\text{Pop}, P_{\Phi})$$

$H(P, Q)$ also has a data compression interpretation.

$H(P, Q)$ can be interpreted as 1.44 times the number of bits used to code draws from P when using the imperfect code defined by Q .

Entropy, Cross Entropy and KL Divergence

Let P and Q be two distribution on the same set.

$$\text{Entropy :} \quad H(P) = E_{y \sim P} - \ln P(y)$$

$$\text{CrossEntropy :} \quad H(P, Q) = E_{y \sim P} - \ln Q(y)$$

$$\begin{aligned} \text{KL Divergence :} \quad KL(P, Q) &= H(P, Q) - H(P) \\ &= E_{y \sim P} \ln \frac{P(y)}{Q(y)} \end{aligned}$$

We have $H(P, Q) \geq H(P)$ or equivalently $KL(P, Q) \geq 0$.

The Universality Assumption

$$\Phi^* = \operatorname{argmin}_{\Phi} H(\text{Pop}, P_{\Phi}) = \operatorname{argmin}_{\Phi} H(\text{Pop}) + KL(\text{Pop}, P_{\Phi})$$

Universality assumption: P_{Φ} can represent any distribution and Φ can be fully optimized.

This is clearly false for deep networks. But it gives important insights like:

$$P_{\Phi^*} = \text{Pop}$$

This is the motivation for the fundamental equation.

Asymmetry of Cross Entropy

Consider

$$\Phi^* = \operatorname{argmin}_{\Phi} H(P, Q_{\Phi}) \quad (1)$$

$$\Phi^* = \operatorname{argmin}_{\Phi} H(Q_{\Phi}, P) \quad (2)$$

For (1) Q_{Φ} must cover all of the support of P .

For (2) Q_{Φ} concentrates all mass on the point maximizing P .

Asymmetry of KL Divergence

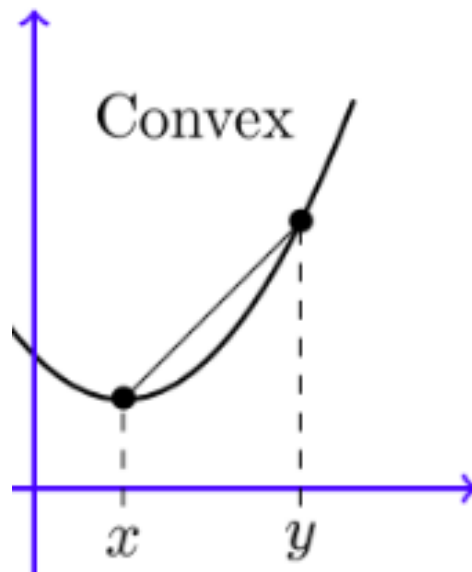
Consider

$$\begin{aligned}\Phi^* &= \operatorname{argmin}_{\Phi} KL(P, Q_{\Phi}) \\ &= \operatorname{argmin}_{\Phi} H(P, Q_{\Phi})\end{aligned}\tag{1}$$

$$\begin{aligned}\Phi^* &= \operatorname{argmin}_{\Phi} KL(Q_{\Phi}, P) \\ &= \operatorname{argmin}_{\Phi} H(Q_{\Phi}, P) - H(Q_{\Phi})\end{aligned}\tag{2}$$

If Q_{Φ} is not universally expressive we have that (1) still forces Q_{Φ} to cover all of P (or else the KL divergence is infinite) while (2) allows Q_{Φ} to be restricted to a single mode of P (a common outcome).

Proving $KL(P, Q) \geq 0$: Jensen's Inequality



For f convex (upward curving) we have

$$E[f(x)] \geq f(E[x])$$

Proving $KL(P, Q) \geq 0$

$$\begin{aligned} KL(P, Q) &= E_{y \sim P} - \log \frac{Q(y)}{P(y)} \\ &\geq -\log E_{y \sim P} \frac{Q(y)}{P(y)} \\ &= -\log \sum_y P(y) \frac{Q(y)}{P(y)} \\ &= -\log \sum_y Q(y) \\ &= 0 \end{aligned}$$

Summary

$$\Phi^* = \operatorname{argmin}_{\Phi} H(\text{Pop}, P_{\Phi}) \text{ unconditional}$$

$$\Phi^* = \operatorname{argmin}_{\Phi} E_{x \sim \text{Pop}} H(\text{Pop}(y|x), P_{\Phi}(y|x)) \text{ conditional}$$

$$\text{Entropy :} \quad H(P) = E_{y \sim P} - \ln P(y)$$

$$\text{CrossEntropy :} \quad H(P, Q) = E_{y \sim P} - \ln Q(y)$$

$$\text{KL Divergence :} \quad KL(P, Q) = H(P, Q) - H(P)$$

$$= E_{y \sim P} \ln \frac{P(y)}{Q(y)}$$

$$H(P, Q) \geq H(P), \quad KL(P, Q) \geq 0, \quad \operatorname{argmin}_Q H(P, Q) = P$$

Appendix: The Rearrangement Trick

$$H(P, Q) = H(P) + KL(P, Q)$$

$$KL(P, Q) = H(P, Q) - H(P)$$

The rearrangement trick applies to any expression of the form

$$E_{x \sim P} \ln \left(\prod_i A_i \right) = E_{x \sim P} \sum_i \ln A_i$$

END