The power and hardness of deep learning

Avraham Ruderman
(ANU & NICTA)

Outline

1. Why you might want to do deep learning?
2. How hard is deep leaning? (in theory and in practice)
3. Strategies for finding good parameters

Deep learning

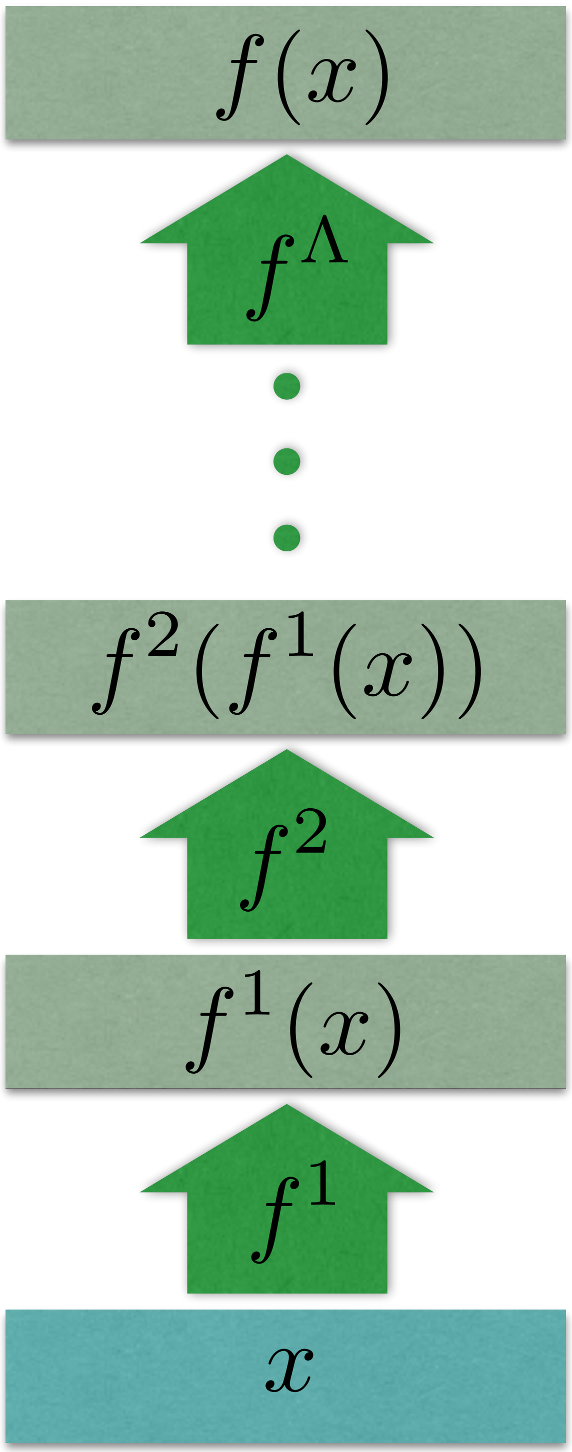
- I will call a predictor f deep if it is computed by a series of functions (layers)

$$f^1, f^2, \ldots, f^\Lambda$$

composed in sequence

$$f(x) = f^\Lambda \circ \ldots \circ f^2 \circ f^1(x)$$

- I will call a learning algorithm deep if it adjusts parameters in multiple layers of a deep predictor



Common components of deep models (1)

- Adaptive linear functions

$$f_W : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} : x \mapsto Wx$$

where W is a $d_2 \times d_1$ matrix

- Adaptive linear convolution functions

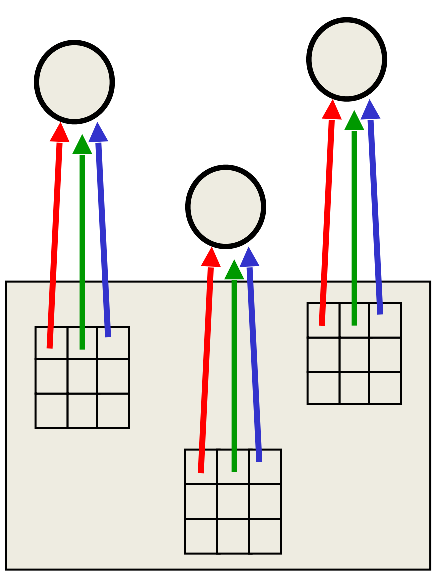


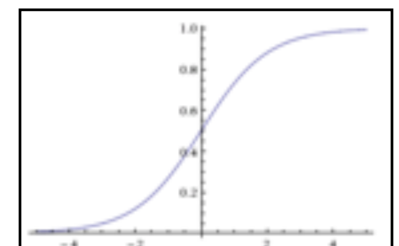
Image from:[Hinton2012]

Common components of deep models (2)

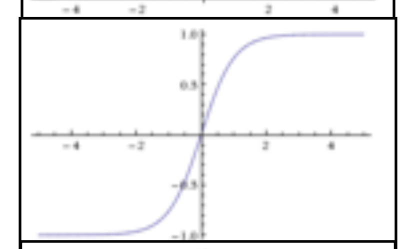
- Non-adaptive nonlinear functions usually defined component-wise:

$$f(x) : \mathbb{R}^d \to \mathbb{R}^d : x \mapsto (\sigma(x_1), \ldots, \sigma(x_d))$$

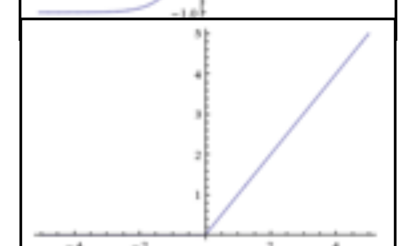
- Sigmoid: $\sigma : x \mapsto \dfrac{e^x}{1 + e^x}$



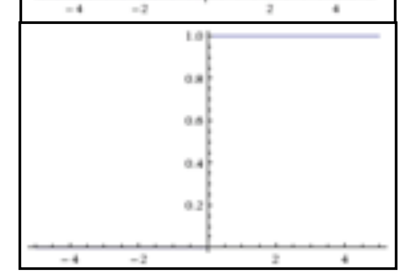
- Tanh: $\sigma : x \mapsto \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$



- ReLU: $\sigma : x \mapsto \max(0, x)$



- Threshold $\sigma : x \mapsto \mathbb{I}_{(x>0)}$



Why do deep learning? (1)

- Any function a deep predictor can approximate can be approximated by a predictor with only two adaptive layers.

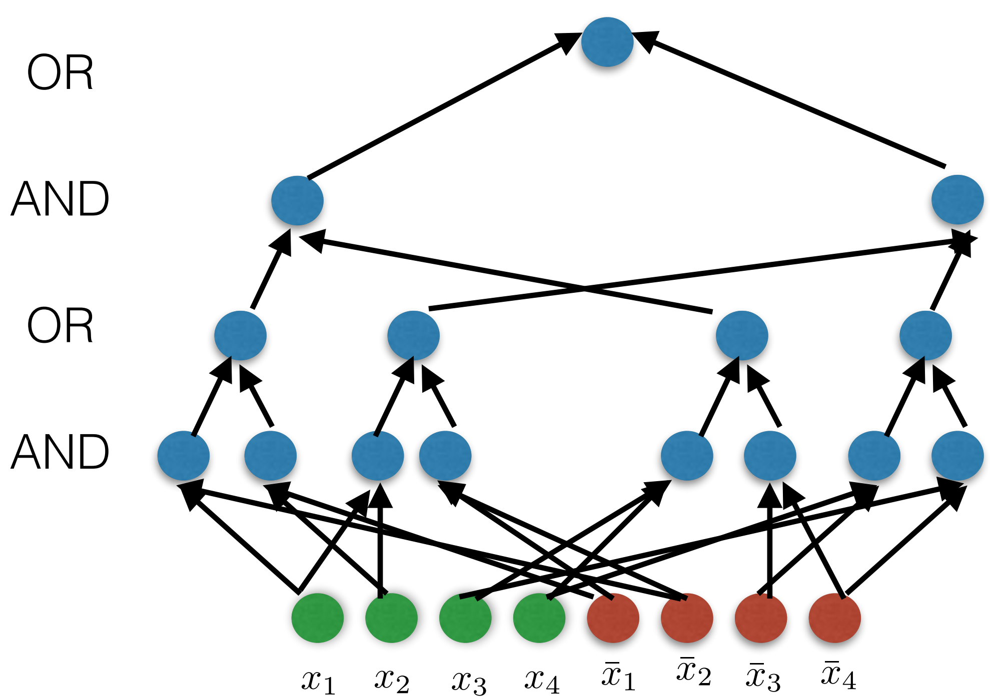
- Theorem [Cybenko1989]: For any continuous function
$$f : [0,1]^d \to \mathbb{R}$$
and any nonlinear layer f_2 computed point-wise by σ with
$$\lim_{x \to \infty} \sigma(x) = 1 \quad \text{and} \quad \lim_{x \to -\infty} \sigma(x) = 0$$
there exist linear layers f_1 and f_3 such that
$$|f(x) - f_3(f_2(f_1(x)))| < \epsilon \quad \forall x \in [0,1]^d$$

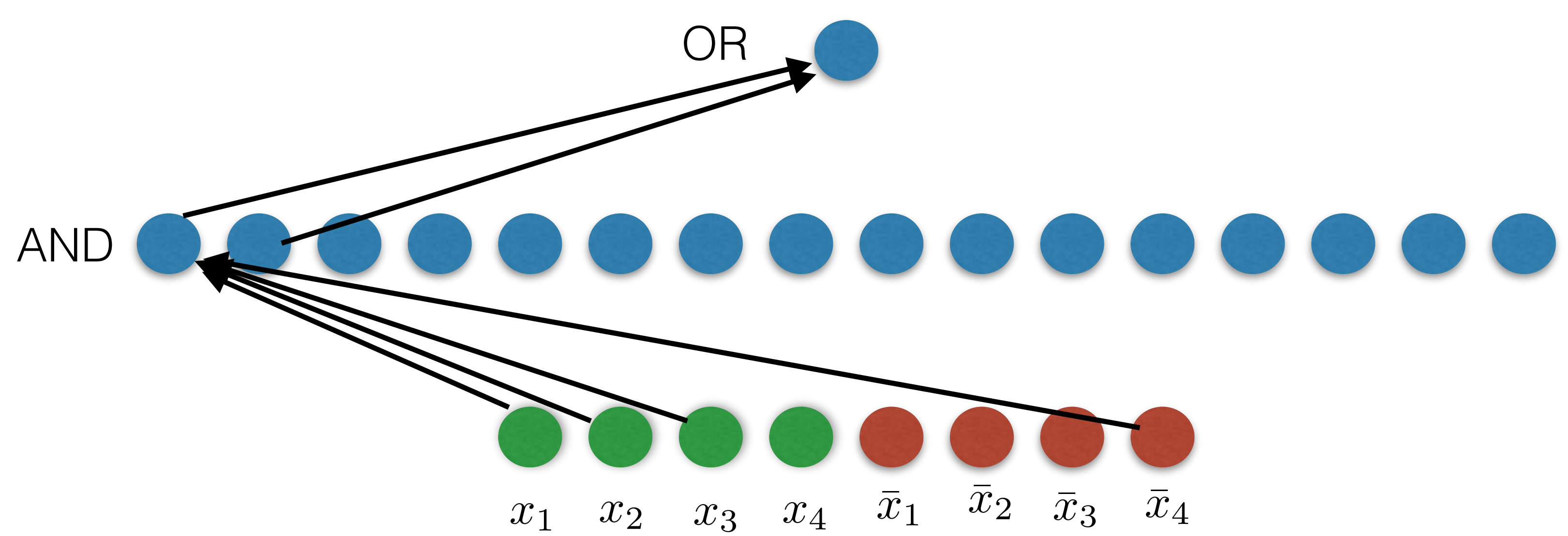
- In words, any continuous function can be uniformly approximated by a neural network with two adaptive linear layers separated by any of a fairly wide class of non-linearities.

Why do deep learning? (2)

- Theorem [FurstSaxeSipser1984]: A boolean circuit of depth d containing AND and OR gates with unbounded fan-in and fan-out 1 computing the parity function on n bits has size

$$\Omega\left(2^{n^{1/d}}\right)$$



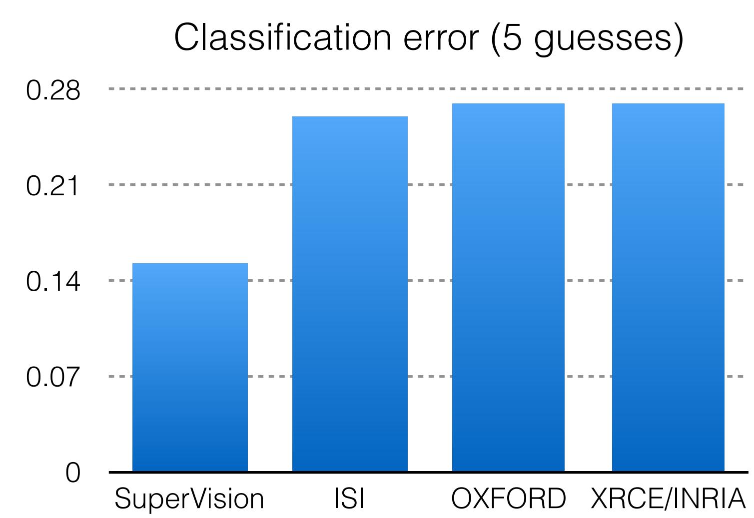


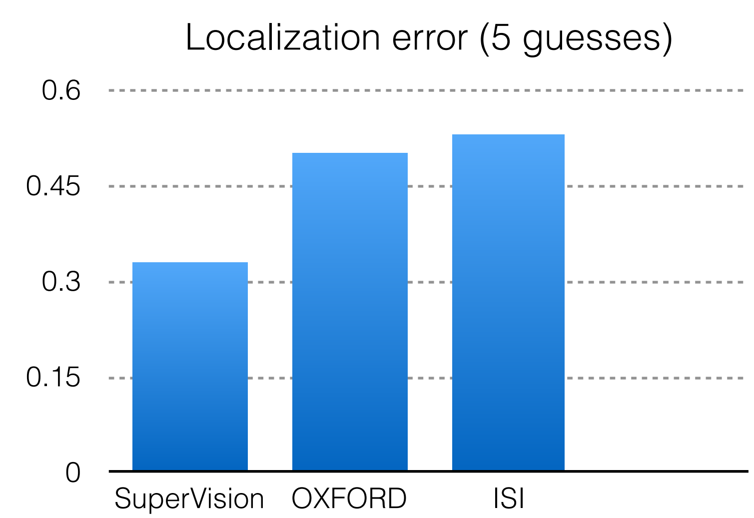
Why do deep learning? (3)

- Deep learning has had a fair bit of empirical success recently.

- Mostly on computer vision, speech recognition (and natural language processing) tasks.

    - Possibly most impressive were the results on the ImageNet Large Scale Visual Recognition Challenge 2012





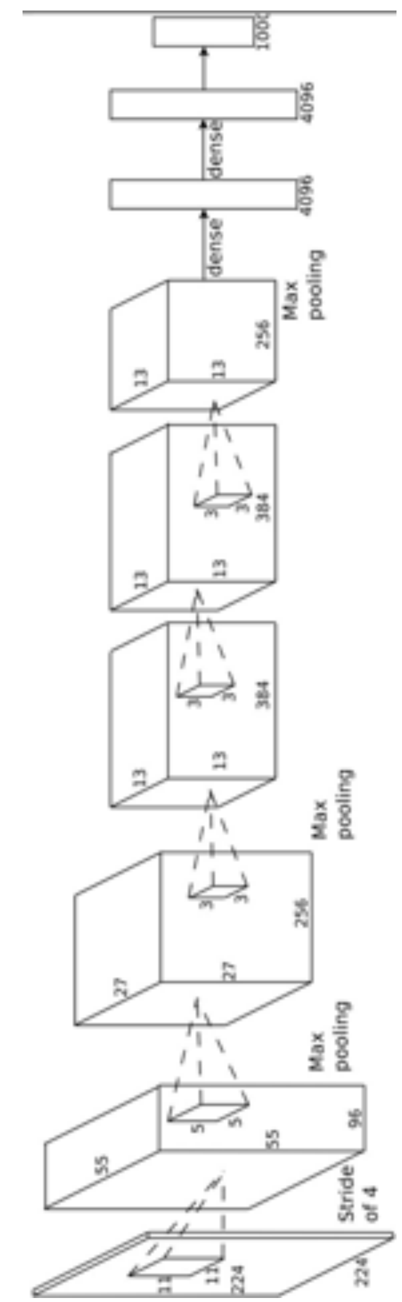
source:[ILCVR2012]

Why do deep learning? (4)

- Deep models correspond to the assumption that a good predictor can be expressed as a sequential application of simple functions.

- You can incorporate prior knowledge about the structure of a good predictor (feature or bug?)

- You get to automate some of the process of feature engineering.



source: [SuperVision2012]

How much labeled data do I need?

- Theorem [BartlettMaiorovMeir1998]: For a deep learner with Λ adaptive linear layers of width w alternating with nonadaptive ReLU layers, it is sufficient for your sample size to grow as

$$\Omega(\Lambda^3 w^2 \log w)$$

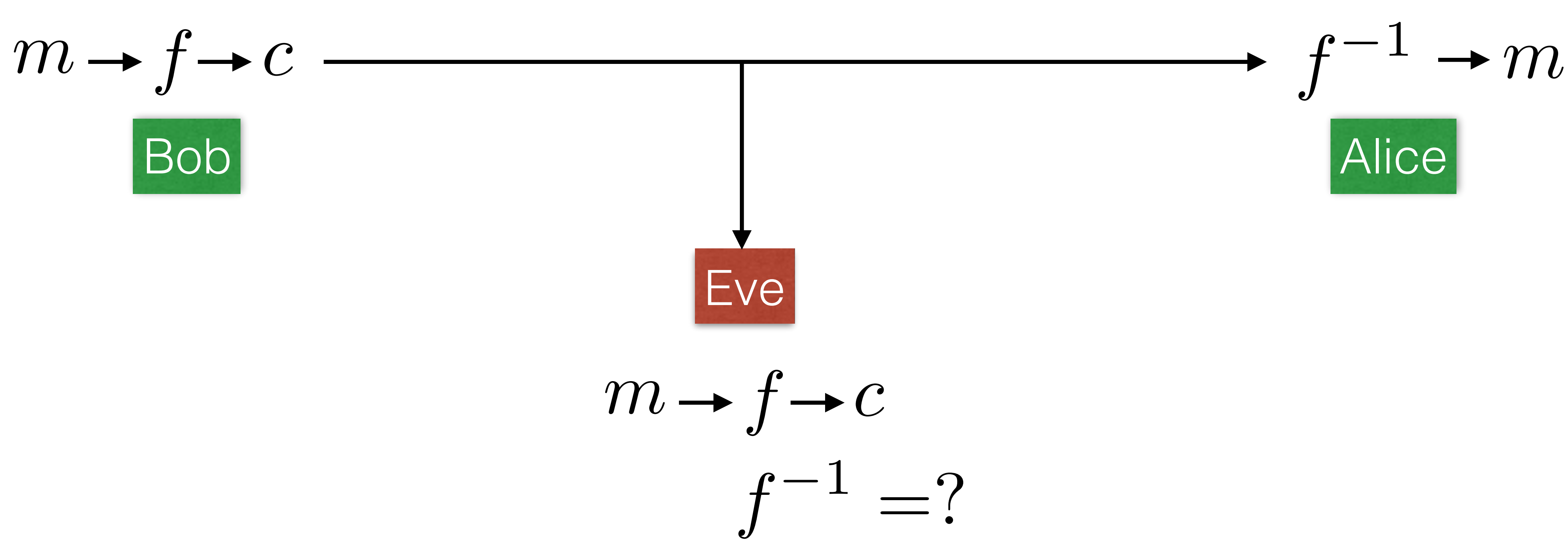
  so that the error you observe on you training set is likely to be very similar to the error you observe on your test set.

- This is in the worst case over data distributions

- This is without applying any regularisation

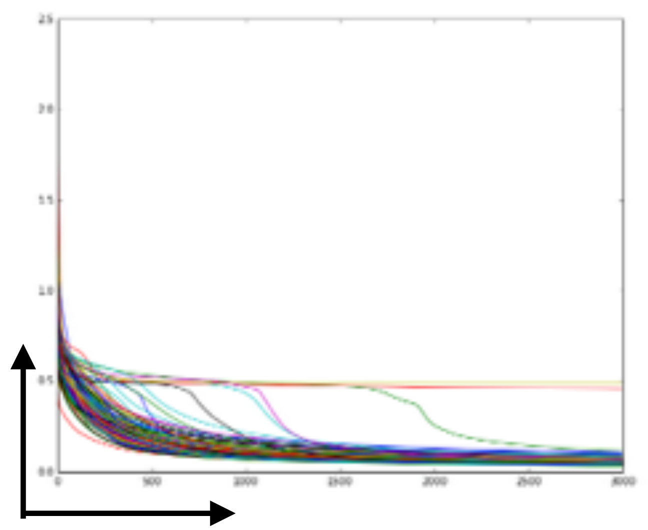
How long will it take? (in theory)

- Theorem [KearnsValiant1994]: If you could learn all constant depth threshold circuits of polynomial size in polynomial time, you could break the RSA crypto-system. The result holds even if you could only predict slightly better than random guessing.



Local optimisation methods for learning deep networks

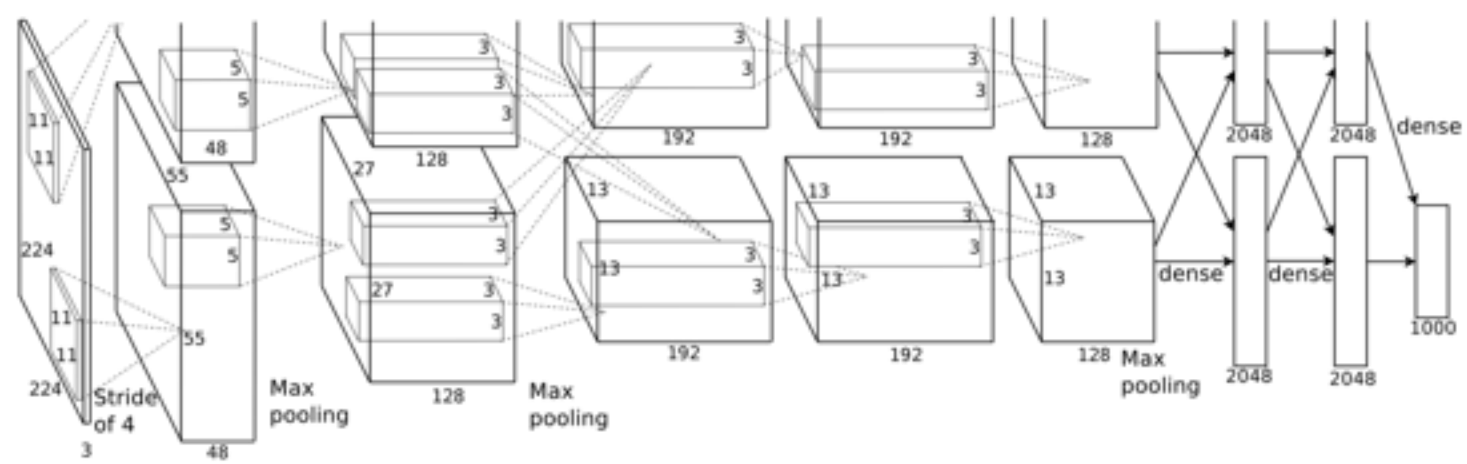
- The most common way to learn the parameters of a deep predictor is using backpropogation [RumelhartHintonWilliams1986].
- The backpropogation algorithm simply updates the parameters by performing gradient descent and uses the chain rule to find the gradients.
- Problems with non-convex optimisation
  - Highly dependent on initialisation
  - No clear stopping rules
  - Local optima do not seem to be the big problem



How long will it take? (in practice)

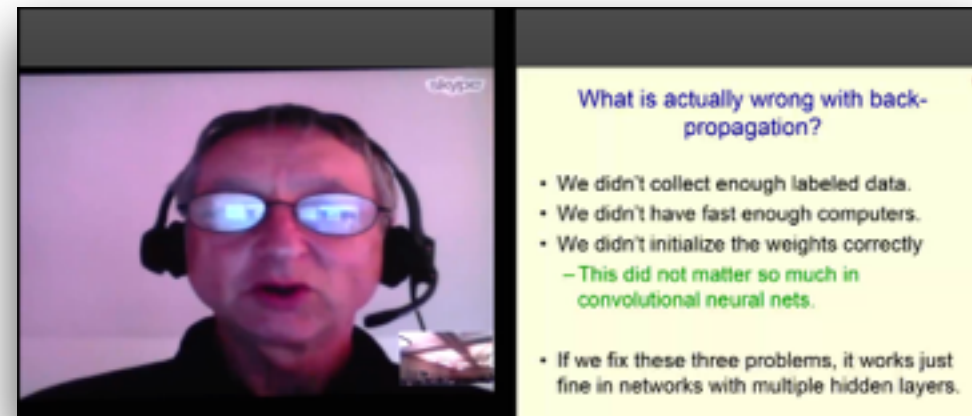
- Practical example [KrizhevskySutskeverHinton2012]:
  - ~1M datapoints
  - ~65K features
  - 60M parameter model
  - 8 adaptive layers
  - Took about 1 week on 2 GPUs

- Deep learning still requires significant effort in tweaking hyper-parameters.
  - Network architecture
  - Learning rates schedules
  - Regularisation per layer



Why didn't deep learning work until recently?

- According to Geoff Hinton [Hinton2013, SutskeverMartensDahlHinton2013]:
  - Datasets were too small
  - Computers were too slow
  - We didn't have good initialisation schemes



  - "Generative models were a digression" (RBM's, Auto-encoders etc. as initialisation)
- Algorithms of today are not very different from those of 20 years ago. Though the following do lead to slight improvements
  - Regularisation: Dropout
  - Optimisation: Using ReLU nonlinear layers. Using second order gradient information.

Restricting deep models to make their learning computationally feasible

- [LeeBartlettWilliamson1996]: Can learn bounded fan in networks in polynomial time. Bound is **exponential** in fan-in.

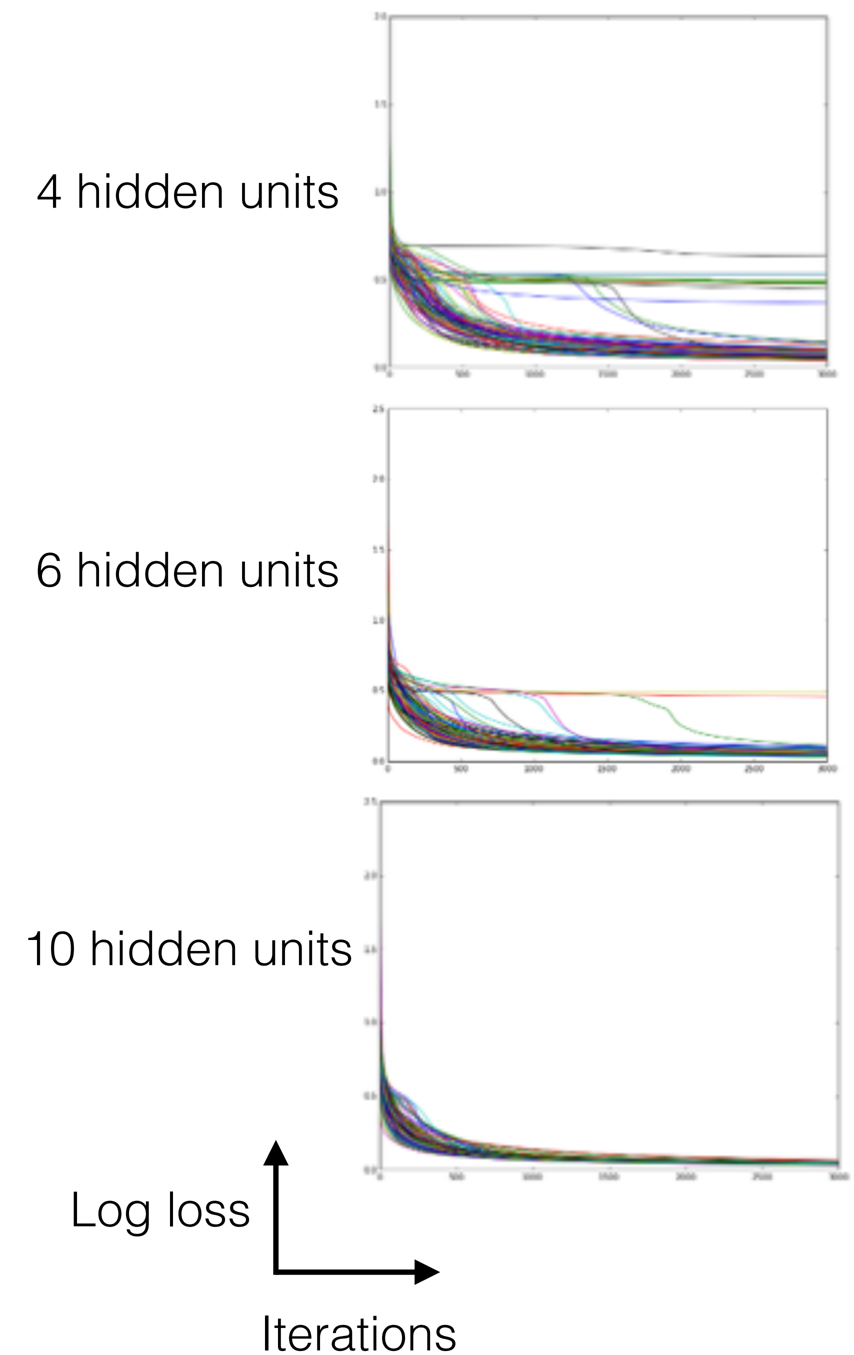
- [AslanChengZhangSchuurmans2013]: Solve a convex relaxation of the training objective for a two layer neural network. Resulting algorithm requires solution of SDPs and does not currently scale beyond a few 100s of data points.

- [LivniShalevShamir2013]: Restrict to nodes computing either linear functions or weighted products

$$(z_1, z_2) \mapsto w z_1 z_2$$

Adding parameters to make the optimisation easier

- Empirical observation: adding more parameters can make the optimisation easier.
- Q: Can we better understand how adding parameters makes optimisation easier?
- Downside: more parameters requires larger dataset to learn.
- Q: Can we better understand this tradeoff?



Future directions?

- Are there other restricted models where we can learn efficiently?
- Can we say anything more precise about when adding more parameters to the optimisation helps and how?
- Are there distributional assumptions that make deep learning easier?
- Are there other network components that can be used to achieve greater statistical or computational efficiency?

Where to get started

- Geoff Hinton's Coursera course:
  "Neural Networks for Machine Learning"

- http://www.deeplearning.net/tutorial/

  - This tutorial uses a Python library called Theano which allows fairly seamless use of GPUs for neural networks and other models.

  - You specify your model in Python and then Theano compiles the necessary functions into C code that can run on CPU or GPU.

References (1)

- SutskeverMartensDahlHinton2013: Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013.

- Hinton2013: http://techtalks.tv/talks/drednets/58115/ starting ~5:00. Retrieved 05/05/2014

- KearnsValiant1994: Kearns, Michael, and Leslie Valiant. "Cryptographic limitations on learning Boolean formulae and finite automata." *Journal of the ACM (JACM)* 41.1 (1994): 67-95.

- BlumRivest1992:Blum, Avrim L., and Ronald L. Rivest. "Training a 3-node neural network is NP-complete." Neural Networks 5.1 (1992): 117-127.

References (2)

- LeeBartlettWillimason1996:Lee, Wee Sun, Peter L. Bartlett, and Robert C. Williamson. "Efficient agnostic learning of neural networks with bounded fan-in." *Information Theory, IEEE Transactions on* 42.6 (1996): 2118-2132.

- AslanChengZhangSchuurmans2013:Aslan, Özlem, et al. "Convex Two-Layer Modeling." *Advances in Neural Information Processing Systems*. 2013.

- LivniShalevShamir2013:Livni, Roi, Shai Shalev-Shwartz, and Ohad Shamir. "A provably efficient algorithm for training deep networks." *arXiv preprint arXiv:1304.7045* (2013).

- KrizhevskySutskeverHinton2012: Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *NIPS*. Vol. 1. No. 2. 2012.

References (3)

- [ILCVR2012]: http://www.image-net.org/challenges/LSVRC/2012/results.html
retrieved 05/05/2014
- [BartlettMaiorovMeir1998]: Bartlett, Peter L., Vitaly Maiorov, and Ron Meir. "Almost linear VC-dimension bounds for piecewise polynomial networks." *Neural computation* 10.8 (1998): 2159-2173.
- [SuperVision2012]: http://image-net.org/challenges/LSVRC/2012/supervision.pdf
retrieved 08/05/2014
- [Hinton2012]: https://class.coursera.org/neuralnets-2012-001/lecture slides from lecture 5
retrieved 08/05/2014

Is deep learning computationally feasible? (Theory 2)

- Theorem [BlumRivest1992]: Consider a neural network with an adaptive linear layer followed by a layer containing two threshold units and then an AND or XOR gate. Deciding whether the network can classify a given dataset with zero error is NP-hard.

- For some hard examples, learning a more powerful class is possible.

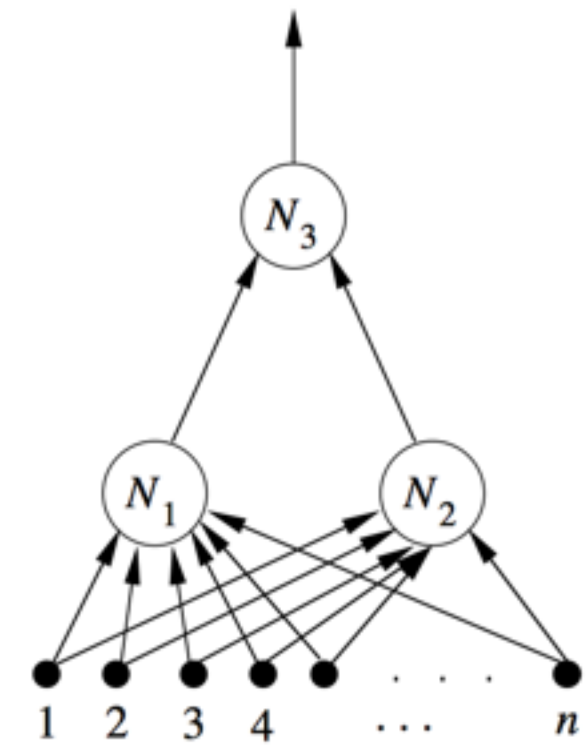


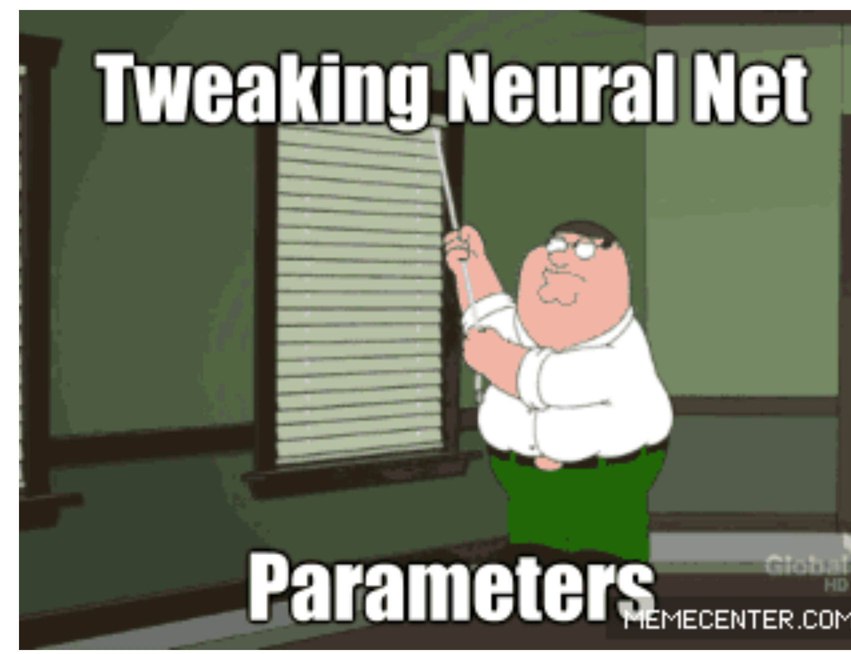
Figure 1: The 3-Node Network.

Is applying deep learning psychologically healthy?

- The number of hyper-parameters to tune typically grows linearly with the number of layers you have.

- The settings of these parameters can make a big difference.

- Non-convex optimisation means when things aren't working it can be very hard to figure out why.



from http://bbabenko.tumblr.com