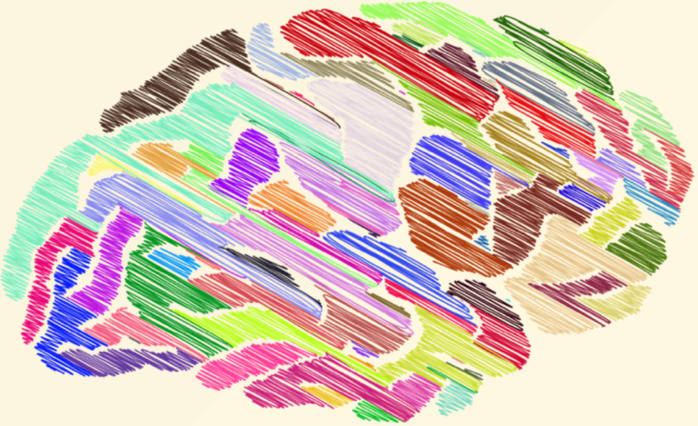
Neural Architecture Design



Jeremy Bernstein
bernstein@caltech.edu

Structure of this topic

Six lectures, covering:

1. Tools for understanding neural nets
2. Application: optimisation
3. Application: generalisation

Plus two homeworks.

Agenda for today

1. Class philosophy
2. Neural network basics
3. Motivating questions
4. Architecture design
5. Perturbation theory

Why theorise?

Why theorise?

Some reasons people do machine learning theory:

- They like math (aesthetes).
- *"You couldn't possibly use an algorithm without a theoretical guarantee!"*

Why theorise?

In this class, the main motivation will be:

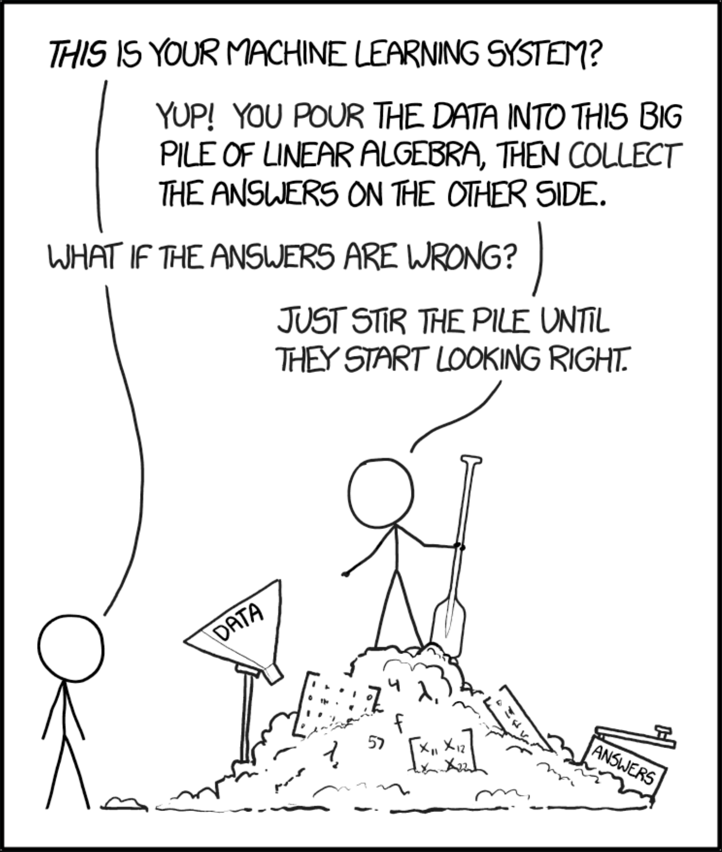
- Build a better understanding of what works, so as to both improve and build upon it.

e.g. combine w/ control

The stages of theory

1. Empirical exploration
2. Modelling "all models are wrong ... some are useful"
3. Derivation
4. Empirical validation

Pure exploration



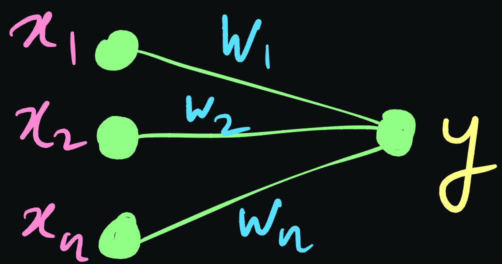
`xkcd.com/1838`

Pure derivation

kurzgesagt.org

Neural network basics

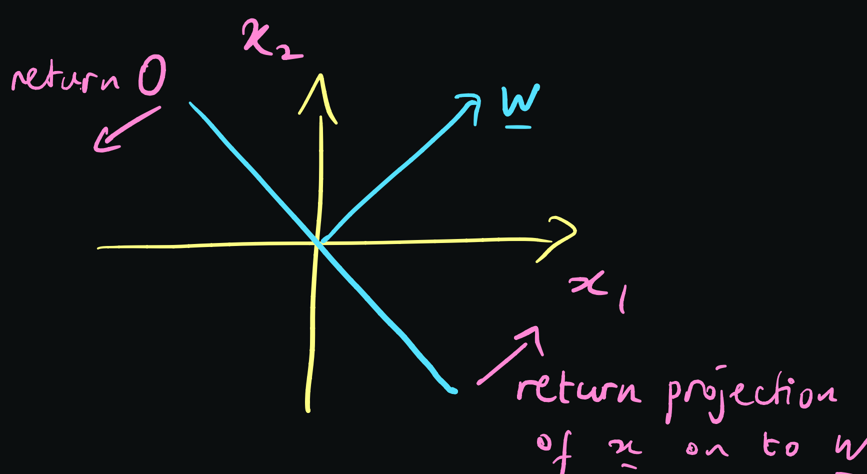
The artificial neuron



$$y = \phi\left(\sum_{i=1}^{n} w_i x_i\right) = \phi\left(\underline{w}^T \underline{x}\right)$$

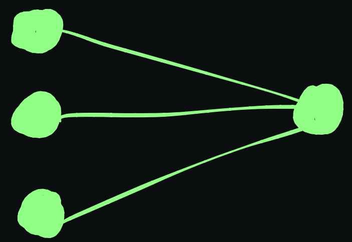
ϕ is the nonlinearity,
e.g. $\phi(x) = \max(0, x)$

"relu"

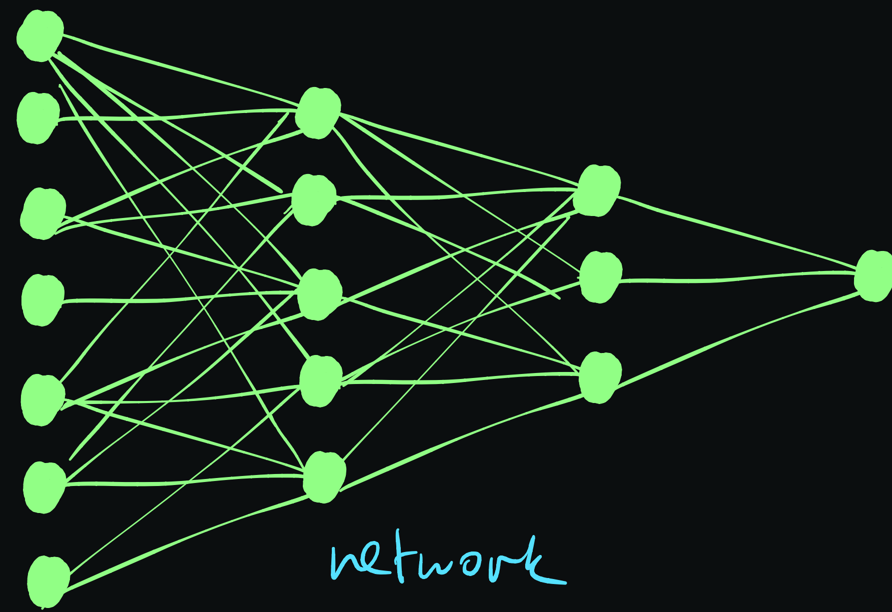


geometric interpretation of relu neuron

Composing neurons



neuron



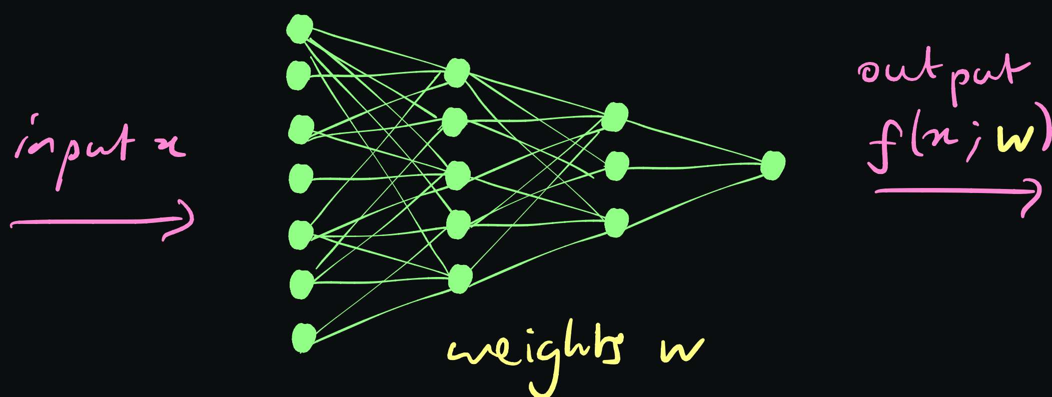
network

directed acyclic graph composed of neurons

General question: how do local properties of neurons translate to global properties of the network?

Backprop: a global view

Wish to train network to fit some targets.



Supervised learning: dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{N}$

Construct loss function $\mathcal{L}(w) = \sum_{i=1}^{N} \left(f(x^{(i)}; w) - y^{(i)}\right)^2$

Run gradient descent: $w \longrightarrow w - \eta \nabla_w \mathcal{L}(w)$

η is the "learning rate" — how small should it be?

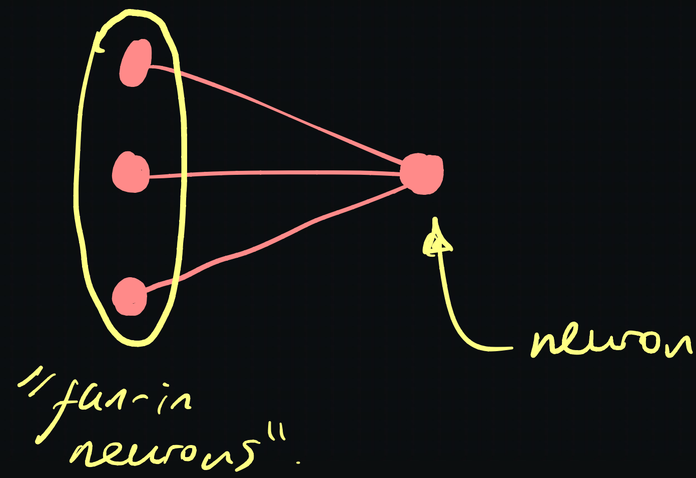
Backprop: a local view

ith activation at layer l+1

$$h^i_{l+1} = \mathrm{relu}\Big(\sum_j W^{ij}_{l+1} h^j_l\Big)$$

(ij)th weight at layer l+1

A neuron aggregates inputs over its "fan-in"

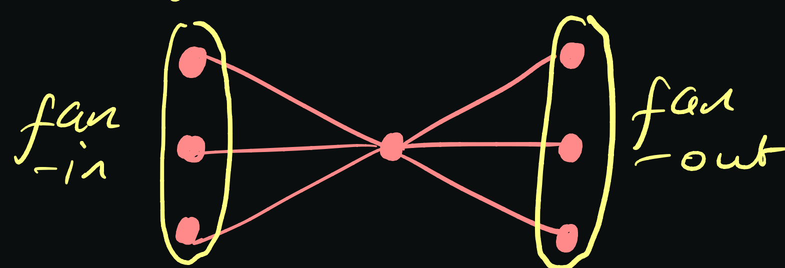


Backprop: a local view

<u>Backward pass</u> (2 steps)

- $\frac{\partial \mathcal{L}}{\partial h_l^j} = \sum_i \frac{\partial \mathcal{L}}{\partial h_{l+1}^i} \times \mathbb{I}[h_{l+1}^i > 0] \times W_{l+1}^{ij};$ ①
- $\frac{\partial \mathcal{L}}{\partial W_l^{ij}} = \frac{\partial \mathcal{L}}{\partial h_l^i} \times \mathbb{I}[h_l^i > 0] \times h_{l-1}^j.$ ②

① a neuron aggregates gradients over its "fan-out"



② the neuron uses this gradient to update its fan-in weights.

Motivating questions

Optimisation

- How do we design principled training algorithms for neural nets?
- How do we move beyond classic optimisation theory?

e.g convex opt.

currently, practioners tune not only the optimiser hyperparameters, but also the optimiser itself (e.g. Adam vs SGD)

Generalisation

- Why do neural nets generalise when

$$\#\text{parameters} \gg \#\text{data}?$$

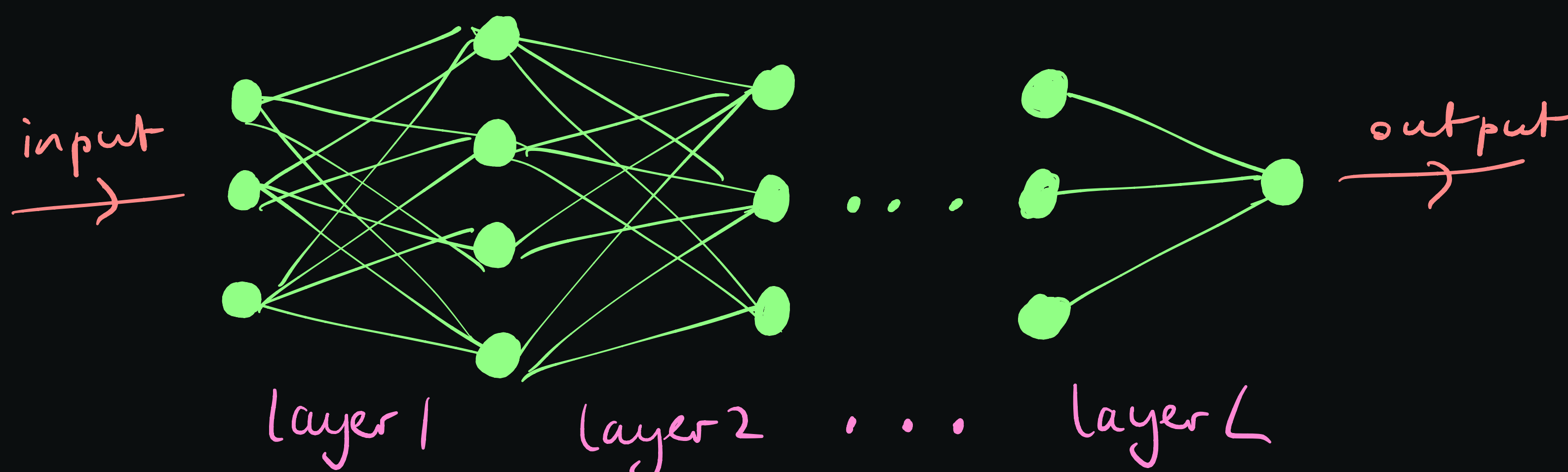
- Why do neural nets generalise when they have the capacity to fit any labelling of the training data?

this violates the central premise of Vapnik – Chervonenkis learning theory.

Neural architecture design

Multilayer perceptron (MLP)

- layered structure
- each layer is a matrix followed by nonlinearity
- assumes little about the structure of the input.



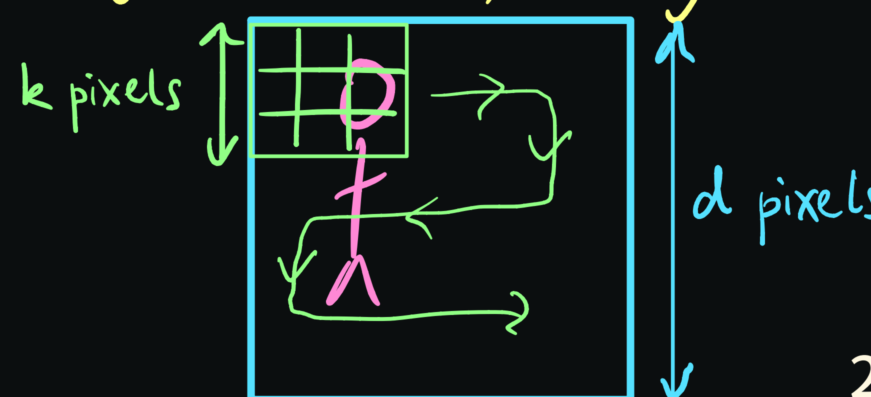
$$f(x; W) = (\phi \circ W_L) \circ \dots \circ (\phi \circ W_1)(x)$$

Convolutional neural network (CNN)

- assumes input is a 2D image
- input has translation invariance

"person" ... still "person"

A network layer exploits this structure by convolving a small filter with the image instead of doing a full matrix multiply.



Compare # parameters

CNN filter	$k \times k$
MLP neuron	$d \times d$

Architecture zoo

Different architectures account for data with different structure. For example:

Structure	Architecture
vector	MLP
image	CNN
sequence	transformer

Neural architecture search (NAS)

NAS is a computational approach to discovering new architectures.

It comes in two main flavours:

1. train lots of networks with slightly different architecture, e.g. NAS via reinforcement learning

[— can be viewed as an "evolutionary" outer loop where network training is the inner loop.]

2. try to learn the network weights and architecture at the same time, e.g. "DARTS"

What's missing?

<u>Neural Architecture Search</u>

① computationally expensive

② results of search biased by how the search space is defined —— would NAS discover transformers?

<u>Intuitive approach</u> "use CNNs for image data"

① not explicit about what the role of architecture is

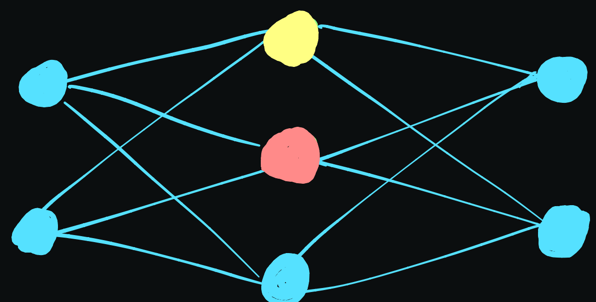
② doesn't answer concrete questions like:
"using architecture X to learn dataset Y will take Z datapoints"

We just looked at a global property of the architecture — network topology.

For now, let's turn to some local properties of neurons and the nonlinearity.

Local properties of architecture

A good rule-of-thumb in architecture design is to ensure that the activations are all on the same scale.



we don't want the activity of neuron A to dominate the activity of neuron B

Wiring constraints

Consider a "linear neuron" $\left(y = \sum_{i=1}^{n} w_i x_i\right)$ and impose two constraints on the weights:

$$(1) \quad \sum_i w_i = 0 \qquad \text{———— "balanced excitation \& inhibition"}$$

$$(2) \quad \sum_i w_i^2 = 1 \qquad \text{———— hyperspherical constraint}$$

Assuming that the inputs x_i are uncorrelated random variables with the same mean and variance 1, then:

$$\mathbb{E}\, y = 0 \quad \text{by (1)} \qquad \text{and} \qquad \mathbb{E}\, y^2 = 1 \quad \text{by (2)}.$$

So the output y has the same scale as the inputs x_i.

Nonlinearity design

Consider the "scaled relu" nonlinearity $\phi(x) = \alpha \cdot \max(0, x)$.
— what's the best α?

Suppose $x \sim \mathcal{N}(0,1)$. Then $\phi(x)$ is "rectified Gaussian"
with variance $\frac{\alpha^2}{2}\left(1 - \frac{1}{\pi}\right) \approx 0.34\,\alpha^2$.

For $\alpha = 1$, the standard relu nonlinearity tends to "squash"
its input by a factor of 0.34.

But by setting $\alpha = \sqrt{\frac{2}{1 - \frac{1}{\pi}}}$ we avoid this, and
obtain $\mathrm{Var}[\phi(x)] = \mathrm{Var}[x] = 1$.

We care about this because we train networks by perturbation!

Perturbation theory

General question: for a network output $f(x;w)$, how does $\Delta f = f(x;w+\Delta w) - f(x;w)$ depend on the size of the perturbation Δw?

Matrix perturbation theory

There are a lot of results about how a matrix A behaves under perturbation $A \mapsto A + \Delta A$. For example:

① perturbation expansions

e.g. $\lambda_i(A+\Delta A) \approx \lambda_i(A) + h(\Delta A) + O(\Delta A^2)$

i^{th} eigenvalue

some linear function

② perturbation bounds

e.g. $\|A+\Delta A\|_F \leq \|A\|_F + \|\Delta A\|_F$

Frobenius norm

triangle inequality

Deep perturbation theory

A neural network is just a product of matrices (and nonlinearities).

Consider a toy example for a network with weight vector $\underline{a} \in \mathbb{R}^d$.

$$f(x; \underline{a}) = \left(\prod_{i=1}^{d} a_i \right) x$$

"deep, linear, scalar network"

Perturbation result:

$$\frac{f(x; \underline{a} + \Delta \underline{a}) - f(x; \underline{a})}{f(x; \underline{a})} = \prod_{i=1}^{d} \left(1 + \frac{\Delta a_i}{a_i} \right) - 1 .$$

Will generalise this to "deep, linear, matrix network" in HW 3.

Summary

- we looked at network topology and said things like "CNNs seem to be well-suited to images". We will return to this issue in lecture 11 when we look at PAC-Bayesian generalisation theory.
- we looked at properties of neurons and saw how they effect the balance of network activity.
- we looked at perturbation theory of compositional functions. This will help in lecture 9 when we look at optimisation theory of neural nets.

Next lecture

We will develop a major tool of NN theory:

the neural network — Gaussian process correspondence

This will let us move from parameter space to function space so that we can study the typical kinds of function that an NN implements.