rec 07: ideas in neural architecture

Let's discuss neural networks design choices. By today's end, we'll be able to:

tailor a neural network to a task's inherent dependencies
exploit a task's symmetries to improve generalization.
explain the assumptions behind each design choice leading to RNNs

As always, please ask questions at any time, including by interrupting me!

Our rough schedule is:
19:30 width and depth
19:45 initialization, expressivity
20:00 'to build a tool, use it'
20:15 latent representations
20:30 symmetries
20:45 rnns: 'derivation', backprop practice
21:00

C. automatic featurization, continued

basic architecture: width and depth

LAST TIME: WIDTH —

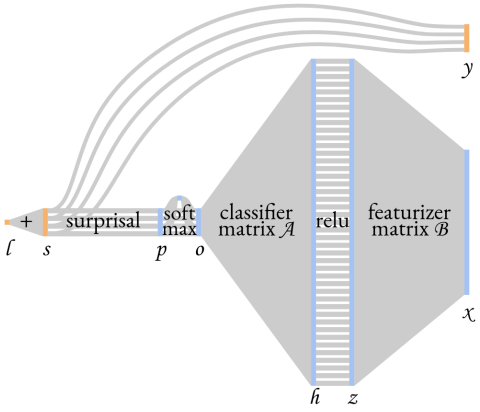

Figure 1: **Shallow neural nets.** Data flows right to left via gray transforms. We use the blue quantities to predict; the orange, to train. Thin vertical strips depict vectors; small squares, scalars. We train the net to maximize data likelihood per its softmax predictions:

$$\ell = \sum_k s_k \quad s_k = y_k \log(1/p_k)$$

$$p_k = \exp(o_k)/\sum_{\bar{k}} \exp(o_{\bar{k}})$$

The decision function o is a linear combination of features h nonlinearly transformed from x:

$$o_k = \sum_j A_{kj} h_j$$

Each "**hidden activation**" or "**learned feature**" h_j measures tresspass past a linear boundary determined by a vector B_j:

$$h_j = \text{relu}(z_j) = \max(0, z_j) \quad z_j = \sum_i B_{ji} x_i$$

TODAY: ALSO DEPTH —

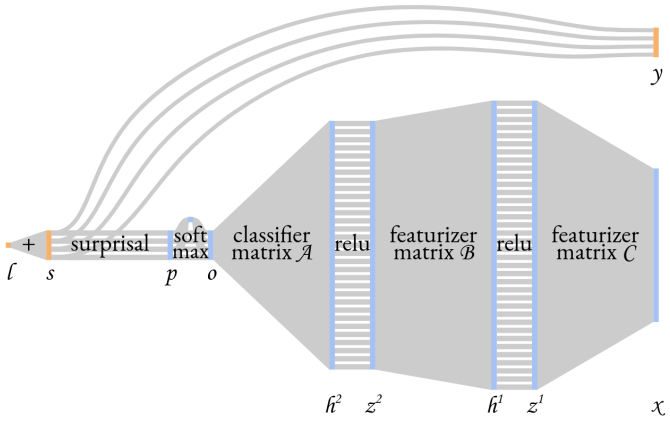

Figure 2: **A deeper neural net.** As before, data flows right to left via gray transforms. We train the net to maximize data likelihood per its softmax predictions and the decision function o is a linear combination of features h^2 nonlinearly transformed from x. However, that nonlinear transform is now less direct: we transform x to a first layer h^1 of hidden features and then to h^2. Explicitly:

$$h^2_j = \text{relu}(z^2_j) \quad z^2_j = \sum_i B_{ji} h^1_i$$

and

$$h^1_j = \text{relu}(z^2_j) \quad z^1_j = \sum_i C_{ji} x_i$$

We may regard x as h^0 and o as z^3 if we wish. We say this net has a **depth** of **three weight layers** or of **two hidden layers**.

The next three questions consider classifying raw dimension-784 inputs into 10 classes. We'll answer in terms of the dimensions (one number for the shallow net; two for the deeper net) of the hidden layers.

Exercise: *How many parameters does the shallow net have, ignoring bias terms?*
Exercise: *How many parameters does the deeper net have, ignoring bias terms?*
Exercise: *How about if we allow bias terms for each weight layer?*

INITIALIZATION — In the next three questions, we initialize all weights to zero. We're doing binary classification without weight regularization.

Exercise: *What is the training loss at initialization?*

Exercise: *What is the loss gradient at initialization?*

Exercise: *What is the testing loss after a thousand SGD updates?*

Due to the pathology° uncovered in the above exercises, we like to *randomly* initialize our weights. What distribution should we use? In what follows, we focus on the input-most layer defined on an input vector x by $h^1_j = \text{relu}\left(\sum_i C_{ji} x_i\right)$. We assume that $D^1 \times D^0$ many matrix elements C_{ji}s are chosen independently according to some centered distribution with expected absolute value s.

← The nonconvexity of learning — as evidenced by permutation symmetry of hidden features in the model — allows this pathology.

Exercise: *If each* $|x_i| \approx 1$ *then each* $|z^1_j| \approx$ *what?*

Exercise: *If each* $|x_i| \approx 1$ *then each* $|h^1_j| \approx$ *what?*

Exercise: *If each* $|x_i| \approx 1$ *and each* $|\partial \ell / \partial h_j| \approx 1$ *then each* $|\partial \ell / \partial x_i| \approx$ *what?*

So for calibrated forward propagation, we might initialize each C_{ji} to have scale roughly $\sqrt{2/D^0}$.° For calibrated backward propagation, we might initialize each C_{ji} to have scale roughly $\sqrt{2/D^1}$. We often use a compromise:°

← Don't pay much heed to the constant factor 2.

← This formula is named after **Glorot, Bengio, Xavier**, and probably others. Don't pay much heed to the constant factor 4.

$$|C_{ji}| \approx \sqrt{\frac{4}{D^1 + D^0}}$$

For example, we can initialize by sampling C_{ji} from a centered normal with variance $4/(D^1 + D^0)$. The tension between the forward and backward scales is least° when $D^1 \approx D^0$.

← This is a weak reason to favor gradual rather than abrupt changes in dimension across a neural network. It's a weak reason because one could also counter this tension by using different learning rates for each layer. In practice, adaptive methods such as ADAM automatically do the latter.

EXPRESSIVITY — Both **wider** and **deeper** neural nets (more hidden features per hidden layer; more hidden layers) have more supple decision boundaries than smaller neural nets. We can "spend" parameters on increased width or increased depth: both add complexity, much like increasing the regularization weakness $C = 1/\lambda$. Intuitively, increasing width is like enriching a child's vocabulary while increasing depth is like enriching a child's grammar. The question *Would the cow have helped its kin had not that act been what we loved?* consists of simple, one-syllable words but expresses a pretty sophisticated idea.

So let's consider a neural network $\mathcal{N}$ with one hidden layer and with layer dimensions $(1 \leftarrow 30 \leftarrow 100)$. This has roughly 3000 many parameters. By contrast, we could try a network $\mathcal{N}'$ with two hidden layers and with layer dimensions $(1 \leftarrow 25 \leftarrow 25 \leftarrow 100)$. This also has roughly 3100 many parameters. Which of these decision boundaries can $\mathcal{N}'$ approximate much better than $\mathcal{N}$?

Exercise: *...recognizing whether a given* 10×10 *handwritten ASCII character (dark ink on light paper) belongs to the set* `0123456789` *of standard arabic numerals.*

Exercise: *...recognizing whether a given* 10×10 *bit matrix has all rows the same.*

Exercise: *...recognizing whether a given* 10×10 *bit matrix has no two rows the same.*

Exercise: *...recognizing whether the input lies within the* 100*-dimensional unit ball.*

The above four exercises are written pretty vaguely. They're just to get us thinking about the geometry of depth. Exam questions will of course be dramatically more precisely posed.

The following unfairly difficult exercise invites us to consider how depth and width affect expressivity. Unlike the previous exercise, it fails to illustrate typical behaviors visible only in high dimensions.

Exercise: *(Unfair but educational). The following four collections of images show decision boundaries from randomly initializations of four architectures. The hidden dimensions are* $(4), (64), (4, 64), (64, 4)$. *Match architectures to collections.*

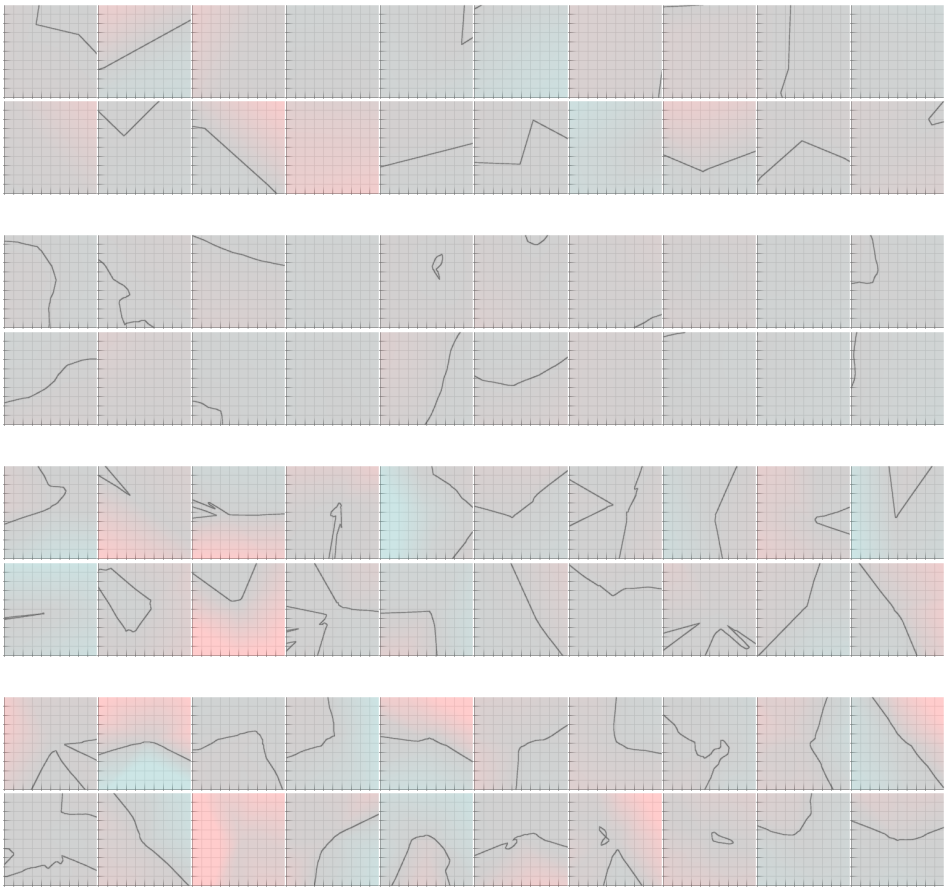

Figure 3: **Four collections of decision boundaries**. Each collection shows boundaries for twenty random initializations of one of four architecture. Each plot shows ten by ten unit squares with the origin at center. **Top rows**: decision boundaries are jagged polygons with edges long and angles moderate. The boundaries show almost no oscillation in their direction of curvature: counterclockwise turns are never quickly followed by clockwise turns. The displayed plane tends to be cut into one or two pieces. **Center top rows**: decision boundaries are smooth curves with edges short and angles small. The boundaries show little oscillation in their direction of curvature: counterclockwise turns are usually not quickly followed by clockwise turns. The displayed plane tends to be cut into one or two pieces. **Center bottom rows**: decision boundaries are jagged curves with edges short and angles sometimes small and often large. The boundaries show much oscillation in their direction of curvature: counterclockwise turns are often quickly followed by clockwise turns. The displayed plane tends to be cut into two or three (up to five) pieces. **Bottom rows**: decision boundaries are smoothed polygons with edges short and angles usually small and occasionally large. The boundaries show moderate oscillation in their direction of curvature: counterclockwise turns are sometimes quickly followed by clockwise turns. The displayed plane tends to be cut into two or three (up to four) pieces.

HYPERPARAMETERS AFFECT GENERALIZATION —

Exercise: *why is the generalization gap usually positive?*

Exercise: *why not just gradient descend on test loss?*

For the two questions below, we assume a fixed, smallish training set size and a fixed, moderate number of gradient descent steps.

Exercise: *sketch the training and testing accuracies as a function of hidden dimension.*

Exercise: *sketch the training and testing accuracies as a function of the learning rate.*

wishful thinking

TO BUILD A TOOL, USE IT — sam draws stuff

ARCHITECTURE, DEPTH, HIERARCHY — sam draws stuff

LATENT REPRESENTATIONS — sam draws stuff

EXPLOITING SYMMETRY — Let's help our machine not re-invent the wheel.

data augmentation
canonicalization
data abstraction
equivariant architecture
sam draws stuff

example: recurrent neural networks

LOCALITY AND SYMMETRY: 1D CNN — sam draws stuff

LATENTS AND DEPENDENCIES: RNN — sam draws stuff

MEMORY ACTIVATION FUNCTIONS — sam draws stuff

FORWARD PASS — sam draws stuff

BACKWARD PASS — sam draws stuff