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.
train an RNN to do basic sentiment analysis.
As always, please ask questions at any time, including by interrupting me!

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

C. automatic featurization, continued

basic architecture: width and depth

LAST TIME: WIDTH —

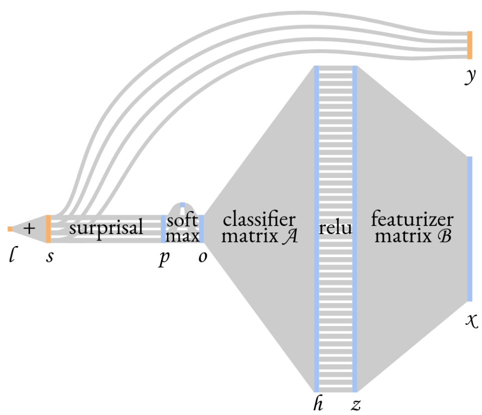

Figure 1: **Shallow neural nets.** Data flows right to left via gray transforms. We use the blue quantities to predict; the orange, to train. Thin vertical strips depict vectors; small squares, scalars. We train the net to maximize data likelihood per its softmax predictions:

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

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

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 —

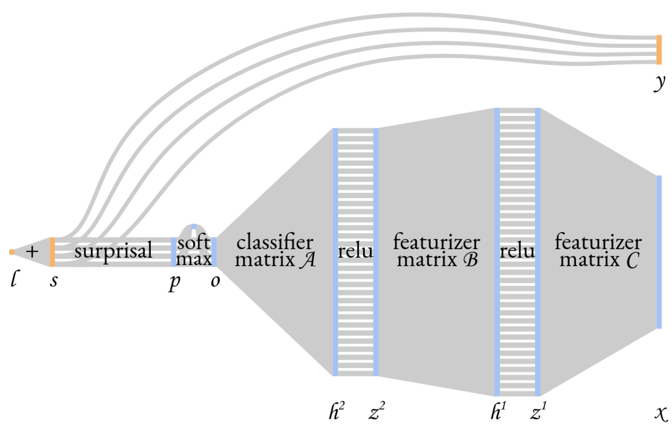

Figure 2: **A deeper neural net.** As before, data flows right to left via gray transforms. We train the net to maximize data likelihood per its softmax predictions and the decision function o is a linear combination of features h^2 nonlinearly transformed from x. However, that nonlinear transform is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

$$h_j^2 = \text{relu}(z_j^2) \quad z_j^2 = \sum_i B_{ji} h_i^1$$

and

$$h_j^1 = \text{relu}(z_j^2) \quad z_j^1 = \sum_i C_{ji} x_i$$

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 symmetries 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, named after **Glorot, Bengio, Xavier**, and probably others:

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

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.

EXPRESSIVITY —

Exercise:

Exercise:

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 —

ARCHITECTURE, DEPTH, HIERARCHY —

LATENT REPRESENTATIONS —

exploiting symmetry through...

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

...DATA AUGMENTATION —

...CANONICALIZATION —

...DATA ABSTRACTION —

...EQUIVARIANT ARCHITECTURE —

example: recurrent neural networks

LOCALITY AND SYMMETRY: 1D CNN —

LATENTS AND DEPENDENCIES: RNN —

FORWARD PASS —

BACKWARD PASS —