recitation 06 (optional 6.86x notes)

Let's start NEURAL NETWORKS! Hurrah! By today's end, we'll be able to:

- *implement* training and prediction code for a shallow neural classifier
- *compute* gradient updates for shallow and deep neural networks
- *diagnose* and remedy bad initialization widths and learning rates

Our rough schedule is:

- *19:30* what's a neural network?
- *19:45* which features are most helpful? — math problems
- *20:00* making features better — math problems
- *20:15* making features better — code problems
- *20:30* what happens during training? — visual example
- *20:45* how do design decisions affect error?
- *21:00*

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

C. automatic featurization

new features from old

ARCHITECTURE AND EXPRESSIVITY — Last week, we saw how approximation and generalization depend crucially on how we *featurize* our data. We made our models more expressive by nonlinearly transforming our 'raw' features into the features on which we actually do linear classification. Last week, we nonlinearly transformed according to hand-designed functions such as a quadratic. Today, we'll reduce the amount of manual labor by *fitting* a nonlinear 'featurizer function' to data! There are two conceptual components here: first, we've got to parameterize a family of possible featurizers (that's part of **architecture**) and second, we've got to figure out how to find a good featurizer from among this family.

First, architecture. We'll define a bunch of features h_j by $h_j = \text{relu}(\sum_i A_{ji} x_i)$. The A_{ji}s are parameters to figure out. This model is a **shallow neural net**.

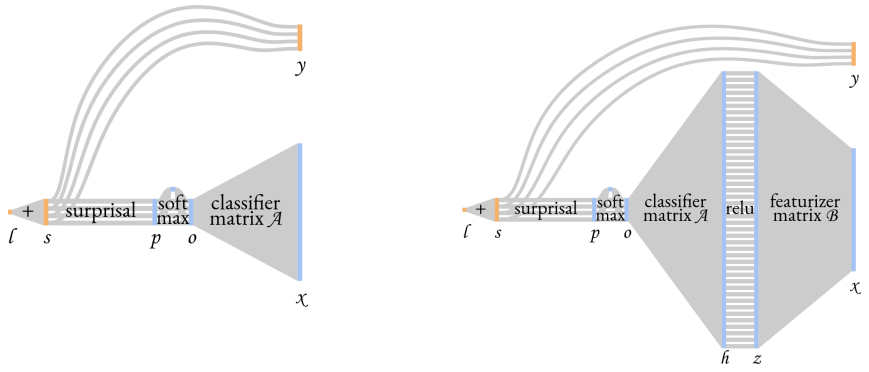


Figure 1: **Linear models (left) vs shallow neural nets (right).** Data flows right to left via gray transforms. The thin vertical strips depict vectors; the small squares, scalars. We use the blue quantities to predict. We additionally use orange to learn from data. Both models use maximum likelihood loss on 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 models both set the decision function o as a linear combination of "features" but the two differ in how they compute those features. The linear model (left) simply uses the raw input features x; the shallow neural net (right) uses features h nonlinearly transformed from x:

$$o_k = \sum_i A_{ki} x_i \quad \text{OR} \quad o_k = \sum_j A_{kj} h_j$$

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

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

We define the **relu** function by

$$\text{relu}(z) = \max(0, z)$$

Do you see what this looks like?

Exercise (PHILIP, MARIANA, MAWUKO): *if our featurizer matrix* B *has shape* 10×3*, and if there are* 4 *labels, then what shape is the classifier matrix* A*?*

Which models can exactly fit the smiley data in the margin? (Imagine a very very thin unpopulated boundary between red and blue; lines that appear straight-ish are supposed to be exactly straight.)

Exercise (Mawuko, Amadeo): *...a quadratic (6-feature) kernel svm?*

Exercise (Philip, James): *...a cubic (10-feature) kernel svm?*

Exercise (Mariana, Bereket): *...a shallow neural network with 4 learned features?*

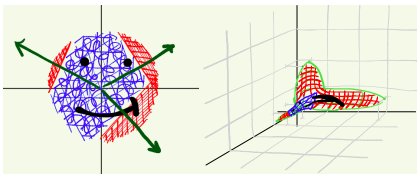


Figure 2: **Smiley Data. Left.** Our 2D raw input vectors with binary class labels in red and blue. The black and green marks are merely annotations. **Right.** The smiley data transformed according to a B matrix whose rows are given by the three green arrows. To aid visualization, we used a variant of relu, namely $f(z) = \max(\exp(-z), 1+z)$

TRAINING WITH RANDOM FIXED FEATURES — To further fortify our intuition for these new features, let's randomly set B and see how linear classification works on the resulting new features. We'll work with this artificial galaxy dataset for ease of visualization:

Exercise (James, Bereket, Amadeo): *which features look most useful?*

SENSITIVITY ANALYSIS —

Exercise (James, Bereket, Amadeo): *what's a formula for* $\partial \ell / \partial o_k$*?*

Exercise (Philip, Mariana, Mawuko): *what's a formula for* $\partial \ell / \partial h_j$*?*

fitting features to data

ORGANIZING DERIVATIVE CALCULATIONS FOR GRADIENT DESCENT —

The following four exercises depend on given weight matrices and input data.

Exercise (James, Bereket, Amadeo): *what code computes the predicted probability?*

Exercise (Philip, Mariana, Mawuko): *what code computes* $\partial \ell / \partial o_k$*?*

Exercise (Philip, Mariana, Mawuko): *what code computes* $\partial \ell / \partial z_j$*?*

Exercise (James, Bereket, Amadeo): *what code computes* $\partial \ell / \partial A_{kj}$ *and* $\partial \ell / \partial B_{ji}$*?*

HORIZONTAL VS VERTICAL LEARNING SIGNALS — In the next three questions we'll compare various machine learning models.

Exercise (Philip, James): *Can different linear models induce the same function from inputs to labels?*

Exercise (Mariana, Bereket): *Can different linear models induce the same function from inputs to label probabilities?*

Exercise (Mawuko, Amadeo): *Can different shallow neural nets induce the same function from inputs to label probabilities?*

To compare likes with likes, we'll fix the raw input featurization and use standard softmax throughout; we won't compare different kernels to each other. When we speak of "different models", we thus mean "models whose weight matrices (necessarily of the same shapes) differ". The one weight matrix of a linear model has shape

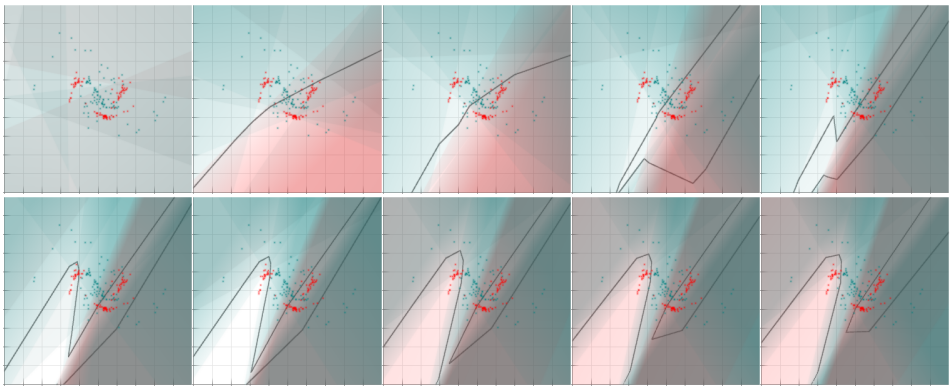
$$(\#\text{ classes} \times \#\text{ raw features})$$

The two weight matrices of a shallow neural network have shape

$$(\#\text{ classes} \times \#\text{ learned features}),$$
$$(\#\text{ learned features} \times \#\text{ raw features})$$

dynamics of shallow net learning

VISUALIZING LEARNING DYNAMICS —



INITIALIZATION AND LEARNING RATE — In the next three questions, we initialize $A = B = 0$.

Exercise (Mariana, Bereket): *What is the training loss at initialization?*

Exercise (Mawuko, Amadeo): *What is the loss gradient at initialization?*

Exercise (Philip, James): *What is the testing accuracy after a thousand SGD updates?*

HYPERPARAMETERS AFFECT GENERALIZATION —

Exercise (James, Bereket, Amadeo): *why is the generalization gap usually positive?*

Exercise (Philip, Mariana, Mawuko): *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 (Philip, Mariana, Mawuko): *what should the training and testing accuracies look like as a function of hidden dimension?*

Exercise (James, Bereket, Amadeo): *what should the training and testing accuracies look like as a function of the learning rate?*