UNIVERSITY OF ILLINOIS
Department of Electrical and Computer Engineering
ECE 417 Multimedia Signal Processing

Lecture 17 Sample Problems

Problem 17.1

Suppose you're given a training database of 200 examples. Each example includes a two-dimensional real-valued feature vector $\vec{x}_i$ and a two-dimensional one-hot label vector $\vec{\zeta}_i$. As it turns out, though, all examples from class $\vec{\zeta} = [1, 0]$ have the same $\vec{x}$, and all examples from class $\vec{\zeta} = [0, 1]$ have the same class:

$$\left(\vec{x}_i, \vec{\zeta}_i\right) = \begin{cases} \left(\begin{bmatrix} 2 \\ -2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) & 1 \le i \le 100 \\ \left(\begin{bmatrix} -2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) & 101 \le i \le 200 \end{cases}$$

You want to train a one-layer neural net using a softmax output:

$$y_{ki} = \frac{e^{a_{ki}}}{\sum_m e^{a_{mi}}}, \quad \vec{a}_i = U\vec{x}_i$$

Since both $\vec{y}$ and $\vec{x}$ are 2D vectors, U is a 2×2 matrix. Its coefficients are trained to minimize cross-entropy

$$u_{kj} \leftarrow u_{kj} - \eta \frac{\partial E}{\partial u_{kj}}, \quad E = -\frac{1}{200} \sum_{i=1}^{200} \sum_{k=1}^{2} \zeta_{ki} \ln y_{ki}$$

With initial values $u_{kj} = 0$. Find u_{kj} after one round of gradient-descent training, assuming $\eta = 1$. Notice that after one round of training, the training corpus is classified with 100% accuracy! Notice also that the second row of U is -1 times the first row—that will always be true for a two-class softmax. Why?

Problem 17.2

Suppose you're given a training database of just 4 training examples. Each example includes a two-dimensional real-valued feature vector $\vec{x}_i$ and a two-dimensional one-hot label vector $\vec{\zeta}_i$:

$$\left(\vec{x}_i, \vec{\zeta}_i\right) = \begin{cases} \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) & i = 1 \\ \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) & i = 2 \\ \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) & i = 3 \\ \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) & i = 4 \end{cases}$$

You want to train a two-layer neural net using a softmax output and logistic hidden units:

$$z_{\ell i} = \frac{e^{b_{\ell i}}}{\sum_m e^{b_{mi}}}, \quad \vec{b}_i = V\vec{y}_i$$

$$y_{ki} = \sigma(a_{ki}), \quad \vec{a}_i = U\vec{x}_i$$

Suppose that U and V are initialized as all-zero matrices. Use forward propagation to compute $\vec{y}_i$ and $\vec{z}_i$ for each training token, then use back-propagation to compute $\vec{\epsilon}_i$ and $\vec{\delta}_i$ for each training token, then use the outer products to find

$$V^{(1)} = V^{(0)} - \frac{1}{n}\sum_{i=1}^{n} \vec{\epsilon}_i \vec{y}_i^T, \quad U^{(1)} = U^{(0)} - \frac{1}{n}\sum_{i=1}^{n} \vec{\delta}_i \vec{x}_i^T$$

Notice that, because of the symmetry of this problem, starting from an all-zero initialization will result in a neural net that never trains. In order to train this neural net, you would have to break the symmetry by starting with small random initial values in U and V.