Practice Questions

CS4787 — Principles of Large-Scale Machine Learning Systems

Problem 1: Preconditioning and Adaptive Learning Rates.

Consider the linear regression problem defined by a dataset with examples $x \in \mathbb{R}^d$ and labels $y \in \mathbb{R}$, with model

$$h_w(x) = w^T x$$

and using the square loss

$$L(\hat{y}, y) = \frac{1}{2} \|\hat{y} - y\|^2 .$$

Suppose that $d = 2$ and the dataset contains the examples

$$x_1 = \begin{bmatrix} 1000 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \quad x_3 = \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \quad x_4 = \begin{bmatrix} -1000 \\ 0 \end{bmatrix}, \quad x_5 = \begin{bmatrix} 0 \\ -2 \end{bmatrix},$$

but you don't know the labels y_i *a priori*.

(a). Roughly, what is the largest step size α you could use for SGD on this problem without it diverging? (An order-of-magnitude answer from a back-of-the-envelope calculation is fine.)

(b). What is the condition number κ of this problem? What does this tell us about the convergence rate of gradient descent and SGD on this problem? Roughly how many steps do you expect they would take to converge? (Again, an order-of-magnitude answer from a back-of-the-envelope calculation is fine for this second question.)

(c). Choose a preconditioning matrix P such that preconditioned SGD

$$w_{t+1} = w_t - \alpha P \nabla f(w_t)$$

would perform better than baseline SGD on this task. Roughly how many steps do you expect your new preconditioned model would take to converge? (Again, an order-of-magnitude answer from a back-of-the-envelope calculation is fine for this second question.)

(d). How well do you expect AdaGrad and Adam would perform on this problem? Why?

(e). Do you expect that adding Polyak averaging to SGD could help accelerate learning for this problem? Why or why not?

Problem 2: Dimensionality Reduction and Sparsity.

(a). Describe the method of random projections. When is it useful? According to the J-L lemma, how many dimensions do I need for the space I am projecting into in order to ensure the existence of a projection that preserves pairwise distances for a dataset of size n up to an error tolerance of ϵ?

(b). When would I want to use an *autoencoder* instead of random projection or PCA? How does the overall compute time needed compare between dimensionality reduction with an autoencoder and dimensionality reduction with random projection?

(c). Consider the matrix

$$A = \begin{bmatrix} -7 & 0 & 3 & 0 & 0 & 4 & -2 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Express this matrix as a sparse matrix in **COO** format and **CSR** format. What are some of the advantages of each storage format?

Problem 3: Deep Learning.

(a). Write down the expression for (forward) inference using a deep feedforward fully connected neural network. (That is, if I have an already trained neural network, how do I use it to get a prediction given a training example x?)

(b). What is the computational cost of running inference on a deep neural network? Suppose that all the nonlinearities operate element-wise using the ReLU function

$$\mathsf{ReLU}(a) = \max(a, 0)$$

and the sizes of the layers are d_0 (where $x \in \mathbb{R}^{d_0}$), $d_1, d_2, \ldots, d_{\mathcal{L}} = 1$. How many numerical operations would computing this forward pass require? Which operation dominates: the matrix multiplies or the nonlinearities?

(c). Now write down the expression/algorithm for backpropagation on this same network, using ReLU activation functions. (That is, how do I compute the gradient of this neural network with respect to the weights for a training example x?)

(d). What is the computational cost of running backpropagation on a deep neural network? That is, how many numerical operations would computing backpropagation require? Which operation dominates: the matrix multiplies or the nonlinearities?