Deep Learning: Sample Questions

Yann LeCun

May 2014

1. (35 points) **Quick Basic Knowledge Questions**
   (a) (2 points) under what condition on the step size is stochastic gradient guaranteed to converge?
   (b) (2 points) what are the advantages and disadvantages of stochastic gradient descent versus second-order methods in batch mode
   (c) (2 points) why is it good to subtract the means of each input variable and normalize the variances to 1 when training a neural net?
   (d) (2 points) what is the purpose of the ISTA algorithm?
   (e) (5 points) explain the basic idea behind convolutional nets.
   (f) (5 points) explain the basic idea behind sparse coding and write down the energy function.

2. (20 points) **Multinomial Logistic Regression**: Dagobert is a data scientist whose pointy-haired boss read in the magazine "Bizness Weak" that "multinomial logistic regression is the best method to pro-actively capitalize on customer-focused, enterprise-wide data analytics". He tells Dagobert: "use this multimonial logic digression, or whatever it's called." In the following, the notation $W'X$ denotes the dot product between vectors W and X.
   (a) (10 points) Multinomial logistic regression computes a 4-dimensional output vector Q, whose components are estimates of the conditional probabilities of the categories given the input $Q_y \approx P(Y|X)$. The output is computed as $Q = \text{softmax}(A)$, where A is a 4-dim vector of weight sums whose components are $A_k = W_k'X,\ k = 1..4$, and each W_k is a 50-dim weight vector. The y-th component of the softmax module is geiven by

$$Q_y = \frac{\exp(A_y)}{\sum_{k=1}^{4} \exp(A_k)}$$

   . Give the back-prop formula for the softmax module in the form:

$$\frac{\partial L}{\partial A_i} = {}'\text{something goes here}' \frac{\partial L}{\partial Q_j}$$

   (b) (10 points) Multinomial logistic regression uses the log loss, which for a given sample (X^i, Y^i) is $L_{log} = -\log Q_{Y^i}$, where Q_{Y^i} depends on W and X^i as indicated above. In that case, the loss function for a single sample reduces to

$$L(W, X^i, Y^i) = -\log \left(\frac{\exp(W_{Y^i} X^i}{\sum_{k=1}^{4} \exp(W_k X^i)} \right)$$

   Give the stochastic gradient descent update formula for a each W_k. Hint: there are two cases: $k = Y^i$ and $k \neq Y^i$.

3. (20 points) **Metric Learning with NCA**: Dagobert wants to use a dimensionality reduction method that shows the separation between the categories. He wants to use Neighborhood Component Analysis (NCA), which is a so-called "metric learning" method that learns a transformation of the input vectors such that in the transformed space, vectors of the same class are closer to each other than vectors of different classes. In its simplest for, NCA uses a linear transform AX where A is the transformation matrix to be learned. For every sample X^i, we are given the set C^i of indices all other samples in the dataset that are of the same class as X^i. The loss function is:

$$\mathcal{L}(A) = \sum_i \sum_{j \in C^i} \frac{\exp(||AX^i - AX^j||^2}{\sum_{k \neq i} \exp(||AX^i - AX^k||^2)}$$

where the sum in the denominator runs over all examples other than X_i.

   (a) (10 points) compute $\frac{\partial \mathcal{L}(A)}{\partial A}$.

   (b) (10 points) write a stochastic gradient update for A.

4. (25 points) Given an $N \times K$ dictionary matrix W_d, sparse coding represents a particular N-dimensional input vector Y as the K-dimensional code vector Z^* that minimizes the energy function

$$E(Y, Z, W_d) = ||Y - W_d Z||^2 + \alpha \sum_{k=1}^{K} |Z_k|$$

$$Z^* = \text{argmin}_Z E(X, Z, W_d)$$

   (a) (5 points) What is the main use of sparse coding?

   (b) (5 points) Generally, we set K larger than N. Why?

   (c) (5 points) Name one optimization algorithm that computes Z^*

   (d) (5 points) Given a training set $\{X^1 \dots X^P\}$, we can learn W_d by minimizing the loss function $\mathcal{L}(W_d) = 1/P \sum_{i=1}^{P} \min_Z E(X^i, Z, W_d)$. The columns of W_d must be constrained within a sphere of a given radius. Explain why.

   (e) (5 points) Imagine that Y is a vector of dimension $N = 2$, and Z of dimensions $K = 3$. Imagine that the training set is such that for each training sample X^i, there exist a code vector Z^i with only 1 or 2 non-zero components such that $X^i = W_d Z^i$. Describe the subset of points in 3D space within which the training samples must lay for this to be true.

5. (15 points) We decided to enter the ImageWeb competition in which we are given 10 million labeled images, each containing one roughly-centered object, and labeled with a single category among 10,000 possible categories. We will use a large convolutional network for this.

   (a) (5 points) name and describe the four or five basic module types that must be assembled to construct a convolutional net.

   (b) (5 points) each input image is 200 by 200 pixels. Design a typical convolutional net architecture that could be applied to these images. Give the number of layers/stages, the sizes of the kernels, number of feature maps, pooling sizes, pooling strides, etc.

   (c) (5 points) what loss function do we typically use with neural nets for classification tasks?

6. (15 points) Someone gives you a large dataset with 10 million vectors of size 1000. One half of the features seems distributed according to a normal (Gaussian) distribution of various means

and standard deviations, while the other half seems to take only positive values and seems distributed according to a log-normal distribution, i.e. a distribution over x such that $\log x$ is Gaussian of various means and standard deviations.

(a) (5 points) why can it be a problem to have features with widely varying means and standard deviations?

(b) (5 points) how would you pre-process the normal-distributed features?

(c) (5 points) how would you pre-process the log-normal-distributed features?

7. (20 points) Neural Nets Suppose we have some neural network architecture, and we would like to add a module that computes the quadratic kernel of examples X and weights W_i

$$K(X, W_i) = (\langle X, W_i \rangle + 1)^2$$

(a) (20 points) Derive the backpropagation formulas for this module, being careful to backprop to both inputs and weights

8. (15 points) **Neural Nets**

Peter works for the online shopping site RainForest.com. He wants to build a 2-layer neural net that takes an n-dimensional vector X describing a user, a p-dimensional vector V describing a product sold by RainForest.com, and predicts the probablility that the user will buy the product. His network has q hidden units.

Instead of the usual tanh or ReLU, he uses the following non-linearity for the hidden units:

$$z_h = g(\sigma_h) = \frac{1}{\beta}\log(1 + e^{\beta\sigma_h})$$

where β is a positive constant and σ is the output of the previous module.

Instead of a conventional linear module for the first layer, Peter wants to use a quadratic module:

$$\sigma_h = \sum_{jj'} t_{hjj'} x_j v_{j'}$$

where $T_{hjj'}$ is a $q \times n \times p$ tensor.

(a) (5 points) write the formula to backpropagate gradients through this non-linearity in the form:

$$\frac{\partial L}{\partial \sigma_h} = \text{some function of} \frac{\partial L}{\partial z_h}$$

(b) (5 points) write the formula to compute the gradient with respect to the T tensor in the form:

$$\frac{\partial L}{\partial T_{hjj'}} = \text{some function of} \frac{\partial L}{\partial \sigma_h}$$

(c) (5 points) write the formula to compute the gradient with respect to the input in the form:

$$\frac{\partial L}{\partial X_j} = \text{some function of} \frac{\partial L}{\partial \sigma_h}$$

9. (30 points) **Convergence of linear regression.** We want to train a linear regressor $\bar{Y} = W'X + b$, where W is the weight vector, X the input vector (both of which are 2-dimensional), and b is a scalar bias. We use the mean squared error loss function $\mathcal{L}(W) = \frac{1}{P}\sum_{i=1}^{p} \frac{1}{2}||Y^i - (W'X^i + b)||^2$.

   (a) (10 points) We are given a first training set composed of the following following two (X, Y) pairs: $([-2 -1], -1), ([+2 +1], +1)$. We will train the system with gradient descent in batch mode (admittedly, a rather inefficient learning procedure): $W(k+1) = W(k) - \eta\frac{\partial L(W(k))}{\partial W}$.

      (i) the loss function is a second degree polynomial in W_1 and W_2 (the two components of W). Write down this polynomial.

      (ii) compute the numerical value of the second derivative of the loss with respect to W_1 and with respect to W_2.

      (iii) We want to set η so that W_1 converges in one iteration. What is this value of *eta*?

      (iv) For this value of η, the weight W_2 will take longer to converge. how can we massage the training vectors so that W_1 and W_2 both converge in one iteration?

   (b) (10 points) We are given a second training set composed of the following four (X, Y) pairs: $([0\ 2], 1)$, $([0\ 0], 1)$, $([2\ 0], 1)$, $([2\ 2], -1)$. The loss function can be written as $\mathcal{L}(W) = \frac{1}{2}W'HW + G'W + C$. where H is the Hessian matrix.

      (i) Compute H.

      (ii) The eigenvalues of H are 3 and 1. Why is this bad?

      (iii) How can we massage the input data so that the eigenvalues of the Hessian at both 1.