CS/CNS/EE 156 a — Caltech

Learning Systems — Fall 2000

Homework # 3

DUE TUESDAY, NOVEMBER 7, 2000, AT 2:30 PM

Collaboration in the sense of discussions is allowed, but you should write the final solutions alone and understand them fully. Do not read class notes or homework solutions from previous years at any time. Other books and notes can be consulted, but not copied from. You should justify your answers, at least briefly. Definitions and notation follow the lectures.

The handouts and data for the homeworks can be found at:

http://work.caltech.edu/cs156/00/homeworks.htm

1. Bias and Variance

Consider the case where the target function $f : [-\pi, \pi] \rightarrow [-1, 1]$ is given by $f(x) = sin(x)$ and the input probability distribution is uniform on $[-\pi, \pi]$. Assume that the training set has only two examples (picked independently), and that the learning algorithm picks the hypothesis that minimizes the mean squared error on the examples. For each of the following learning models, find **(a)** The best hypothesis that approximates f in the mean-squared-error sense (f is assumed to be known for this part), **(b)** The expected value (with respect to the training set) of the hypothesis that the learning algorithm produces, and **(c)** the expected out-of-sample error and its bias and variance components.

(i) The learning model consists of all functions of the form $g(x) = ax + b$ (if you need to deal with the infinitesimal-probability case of two identical examples, choose the hypothesis tangential to f).

(ii) The learning model consists of all functions of the form $g(x) = b$.

To help you solve the problem a table of integrals is provided at the following web address:

http://www.work.caltech.edu/cs156/00/hw/hw3/integrals/

2. Gradient and Hessian

Consider the nonlinear function (from homework #1) $E(u, v) = (ue^v - 2ve^{-u})^2$. We start at the point $u = 1$ and $v = 1$, and take a step of length 0.1 in the u, v space with a view to minimizing E.

(i) Compute the Hessian (2×2 matrix) at the starting point. Recall the values of the gradient ($\frac{\partial E}{\partial u}$ and $\frac{\partial E}{\partial v}$) from homework # 1.

(ii) If we descend along the gradient, compute the value of

$$\Delta E = E(u + \Delta u, v + \Delta v) - E(u, v)$$

using the second-order approximation (keeping terms up to 2nd order in Taylor series), and compare it to the first-order approximation and to the exact value of ΔE from homework # 1.

(iii) Numerically, find Δu and Δv (total length 0.1) that minimize ΔE, and compute the resulting ΔE.

(iv) Repeat (iii), using the second-order approximation.

(v) Repeat (iv), neglecting the off-diagonal elements of the Hessian.

3. Momentum and Adaptive Learning Rate

For this problem you will write a program to optimize the following error function.

$$E(\mathbf{w}) = (\sum_i e^{a_i w_i} + \sum_i e^{-b_i w_i}) \times (\sum_i e^{c_i w_i} + \sum_i e^{-d_i w_i})$$

Use the parameter values specified in

http://www.work.caltech.edu/cs156/00/hw3/parameters.txt

For each part, plot optimization curves $E(\mathbf{w}_t)$ vs. t for 250 time steps.

(i) Write functions to evaluate E and its gradient, and use them to minimize E by gradient descent. Start with $\mathbf{w} = 0$ and learning rate $\eta = 10^{-5}$.

(ii) Modify your code to use an adaptive learning rate. At each step:
if $\Delta E < 0$, set $\eta_{t+1} = \alpha \eta_t$
if $\Delta E \geq 0$, undo the weight update and set $\eta_{t+1} = \beta \eta_t$
Try using $\alpha = 1.1, 1.5, 2$ and $\beta = 0.2, 0.5, 0.8$, and pick the best combination. Compare with the results of part (i).

(iii) Modify your code to use momentum. $\Delta \mathbf{w}_t = -\eta \nabla E + \mu \Delta \mathbf{w}_{t-1}$ Try $\mu = 0.2, 0.5, 0.8, 0.99$, and pick the best value. Compare with parts (i) and (ii).

4. Line Search and Conjugate Gradient

Implement a line search (e.g., the binary search described in the handout, with $\epsilon = 10^{-5}$). Consider the error function $E(\mathbf{w})$ from problem 3.

(i) Using your line search, optimize E by minimizing along the line in the direction of the gradient (steepest descent). Compare the results with those of problem 3.

(ii) Using your line search, implement the conjugate gradient optimization. Reset $\mathbf{d}_t = -\mathbf{g}_t$ every $\mathcal{T}$ steps. Try $\mathcal{T} = 2, 10, 100$, and pick the best value. Compare the optimization to the results of problem 3.