CSE 417T: Homework 2 Solution Sketches

September 22, 2022

Note: These are not intended to be comprehensive, just to help you see what the answers should be.

Implementation E_{in} and the classification error for the non-normalized data are reported below:

Algorithm	E_{in}	Classification Error	
		Training	Testing
GD 10000	.5847	.3092	.3172
GD 100000	.4937	.2237	.2069
GD 1000000	.4354	.1513	.1310

Main takeaways: the generalization performance is excellent, in the sense that testing error is very close to training error, so the model is definitely not overfitting. The performance improves significantly with lower E_{in}, which makes sense given that the model generalizes well. Therefore, the benefits of optimizing the model for the training set logistic error measure are good.

For the normalization, note that we need to normalize the training and test sets using the mean/variance of the training set. It achieves the same error (E_{in}) much faster with appropriately chosen η.

η	E_{in}	Binary training error	Binary test error	# iterations
0.01	0.4074	0.1711	0.1103	23222
0.1	0.4074	0.1711	0.1103	2319
1	0.4074	0.1711	0.1103	229
4	0.4074	0.1711	0.1103	53
7	0.4074	0.1711	0.1103	46
7.5	0.4074	0.1711	0.1103	182
7.6	0.4074	0.1711	0.1103	453
7.7	0.4083	0.1645	0.1172	1000000

Problem 2.22 First, compute $E_{\text{out}}(g^{\mathcal{D}}) = \mathbb{E}_{\mathbf{x},y}[(g^{\mathcal{D}}(\mathbf{x}) - y)^2]$. Replacing y with $f(\mathbf{x}) + \epsilon$ and expanding, we get $\mathbb{E}_{\mathbf{x},\epsilon}(g^{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))^2 - 2(g^{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))\epsilon + \epsilon^2$. Since ϵ is independent of $\mathbf{x}$ the second term disappears, and we get $\mathbb{E}_{\mathbf{x}}(g^{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))^2 + \mathbb{E}_{\epsilon}[\epsilon^2]$. The second term is σ^2 and when taking the expectation over the first one we get back the bias + variance.

Problem 2.24

(a) When you fit the two data points to a line, you get a slope of $x_1 + x_2$ and an intercept of $-x_1x_2$. Since the expectations of both of these terms are 0, you get $\bar{g}(x) = 0$.

(b) Generate many datasets, compute a, b for each of them, and essentially use the average of a large number of samples to simulate expectation of the quantities of interests. Compute $\bar{g}(x) = \bar{a}x + \bar{b}$. Now,

$$E_{\text{out}}(a, b) = \frac{1}{2}\int_{-1}^{1} dx(ax + b - x^2)^2 = \frac{1}{5} + \frac{a^2 - 2b}{3} + b^2$$

The bias can be computed as $E_{\text{out}}(\bar{a}, \bar{b})$. The variance is the variance of $ax + b$, which is given by $x^2\text{Var}(a) + 2x\text{Cov}(a, b) + \text{Var}(b)$. Integrating over x from -1 to 1, the second term disappears, and we are left with $\frac{1}{3}\text{Var}(a) + \text{Var}(b)$ which we can use for the calculation.

(c) Run simulations and report the results, which should be similar to the analytical ones below.

(d) Analytically, $\bar{a} = \bar{b} = 0$. Now in order to use the formulae above, we still need to compute $\overline{a^2} = \mathbb{E}[(x_1 + x_2)^2] = 2/3$ and $\overline{b^2} = \mathbb{E}[x_1^2 x_2^2] = 1/9$. Plugging into the formula for E_{out} above, we get $8/15$, with bias and variance (also computed as above) of $1/5$ and $1/3$ respectively.

Problem 3.4

a Consider two cases. If $y_n \mathbf{w}^T \mathbf{x}_n < 1$ then $E_n(\mathbf{w}) = (1 - y_n \mathbf{w}^T \mathbf{x}_n)^2$ which is differentiable. When $y_n \mathbf{w}^T \mathbf{x}_n > 1$, $E_n(\mathbf{w}) = 0$, which is differentiable. Therefore, the only case where a problem could occur would be when $y_n \mathbf{w}^T \mathbf{x}_n < 1$. From the left, the gradient is $-2(1 - y_n \mathbf{w}^T \mathbf{x}_n) y_n \mathbf{x}_n$ which approaches 0, and from the right the gradient is 0, so the gradient is continuous, hence $E_n(\mathbf{w})$ is continuously differentiable.

b If $y_n \mathbf{w}^T \mathbf{x}_n > 0$ this is trivial. If not, $(1 - y_n \mathbf{w}^T \mathbf{x}_n)^2 > 1$ and we're done. The inequality for the classification error now follows easily.

c If we perform stochastic gradient descent with learning rate $\eta/2$ (it's fine to just derive it with η, this version just ends up giving you back Adaline), then the update for a random data point $(\mathbf{x}_n, y_n)$ is given by $\mathbf{w} = \mathbf{w} - \frac{\eta}{2} \nabla E_n(\mathbf{w})$. Using the gradients derived in part (a), we get the updates:

$$\mathbf{w} = \begin{cases} \mathbf{w} + \eta(1 - y_n \mathbf{w}^T \mathbf{x}_n) y_n \mathbf{x}_n & \text{if } y_n \mathbf{w}^T \mathbf{x}_n < 1 \\ \mathbf{w} & y_n \text{if } \mathbf{w}^T \mathbf{x}_n \geq 1 \end{cases}$$

Problem 3.19 For part (a), as long as no two training examples are identical, this transform will always lead to a linearly separable dataset. The VC dimension is then infinity, so there's no generalization guarantee at all. For part (b), this feature transform is basically encoding similarity to the existing training examples, and as the number of training examples grows the complexity of the encoding grows with them, so again the VC dimension is infinite and you cannot guarantee generalization. By contrast, for part (c), the vector is large (101x101 + 1), so it won't work great for small N, but at least for very large N one could eventually get some generalization.