Optimization of ML Assignment

Eric Muthomi Gitonga

March 2025

1 Gradient Descent

1. **The data situation is called complete separation, i.e., the classes can be perfectly classified with a linear classifier. Show that in this situation if θ perfectly classifies the data then**

   $R_{emp}(\tilde{\theta}) > R_{emp}(\alpha\tilde{\theta})$ with $\alpha > 1$.

   Solution

   Empirical risk is described as the minimum average error of the loss function over a training set. It helps in approximating the true risk, which is the expected loss over the entire dataset and as such, minimize loss on the training data.

   $R_{emp}(\theta) = \sum_{i=1}^{n} log(1 + e^{-y_i\theta^T x_i})$

   Assuming complete separation, there exists $\tilde{\theta}$ such that:

   $$(\tilde{\theta}^T x_i) < 0 \quad if \quad y^i = 0$$
   $$(\tilde{\theta}^T x_i) > 0 \quad if \quad y^i = 1$$

   This means that all samples are correctly classified, and the loss per sample is:

   $$\log(1 + e^{-y_i\tilde{\theta}^T x_i})$$

   Upon scaling$\tilde{\theta}$ by $\alpha > 1$, i.e., $\theta' = \alpha\tilde{\theta}$. The resulting equation becomes:

   $$\log(1 + e^{-y_i\alpha\tilde{\theta}^T x_i})$$

   Since $\alpha > 1$, the term $-y_i\alpha\tilde{\theta}^T x_i$ is more negative than $-y_i\tilde{\theta}^T x_i$:

   $$e^{-y_i\alpha\tilde{\theta}^T x_i} < e^{-y_i\tilde{\theta}^T x_i}$$

   which implies:

$$\log(1 + e^{-y_i \alpha \tilde{\theta}^T x_i}) < \log(1 + e^{-y_i \tilde{\theta}^T x_i})$$

Summing over all n samples,

$$R_{emp}(\alpha\tilde{\theta}) < R_{emp}(\tilde{\theta})$$

2. **Visualize R_{emp} in [-1, 4] $\times$ [-1, 4].**

   Implemented in jupyter notebook by creating a heatmap of the logistic loss function.

$$R_{emp}(\theta) = \sum_{i=1}^{n} log(1 + e^{-y_i(\theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2})})$$

   The script defines a grid of (θ_1, θ_2) values within the range and computes the empirical risk R_{emp} for each pair of θ_1, θ_2

3. **Find the gradient of R_{emp} for arbitrary θ.**

   Solution

   Given the logistic regression:

$$R_{emp}(\theta) = \sum_{i=1}^{n} log(1 + e^{-y_i(\theta^T x_i)})$$

   Where x_i is the feature vectors and y_i as the parameter vectors

   Compute the derivative of the logistic loss for a single sample:

$$\frac{d}{d\theta} \log(1 + e^{-y_i(\theta^T x_i)})$$

   Using the chain rule:

   $f'(g(x))g'(x)$

$$\frac{1}{1 + e^{-y_i(\theta^T x_i)}} \cdot (-y_i e^{-y_i(\theta^T x_i)}) \cdot (-x_i)$$

   Simplifying:

$$\frac{y_i x_i e^{-y_i(\theta^T x_i)}}{1 + e^{-y_i(\theta^T x_i)}}$$

   The gradient with respect to θ is:

$$\nabla_\theta R_{emp}(\theta) = \sum_{i=1}^{n} \frac{-y_i x_i}{1 + e^{y_i(\theta^T x_i)}}$$

4. **Solve the logistic regression via gradient descent. Use a step size $\alpha = 0.01$, starting point $\theta^{[0]} = (0,0)^T$ and train for 500 steps. Repeat this with $\alpha = 0.02$. Explain your observation.**

   Given the gradient with respect to θ as:

$$\nabla_\theta R_{emp}(\theta) = \sum_{i=1}^{n} \frac{-y_i x_i}{1 + e^{y_i(\theta^T x_i)}}$$

   Step updates are made to θ using gradient descent such that:

$$\theta^+ = \theta^t - \alpha \nabla_\theta R_{emp}(\theta)$$

   where α is the step sizes 0.01 and 0.02

   For $\alpha = 0.01$, The gradient descent converges smoothly to an optimal θ, signifying that the trajectory was stable and gradually approached the solution.

   For $\alpha = 0.02$, the convergence is observed to be faster but on the other hand, more oscillations are present. A steady increase in the step size would ultimately lead to divergence.

   This greatly underscores the importance of choosing the optimal learning rate to ensure convergence which is essential for efficient training of models.

5. **Repeat 4) but add an L2 penalization term (with $\lambda = 1$) to the objective. What do you observe now?**

   By integrating the L2 regularization, the logistic regression is resulting equation becomes:

$$R_{emp}(\theta) = \sum_{i=1}^{n} log(1 + e^{-y_i(\theta^T x_i)}) + \frac{\lambda}{2}\|\theta\|^2$$

   where $\|\theta\|^2 = \theta_1^2 + \theta_2^2$ is the L2 norm of vector θ.

   The resulting gradient is:

$$\nabla_\theta R_{emp}(\theta) = \sum_{i=1}^{n} \frac{-y_i x_i}{1 + e^{y_i(\theta^T x_i)}} + \lambda\theta$$

   The gradient descent is:

$$\theta^{t+1} = \theta^t - \alpha(\sum_{i=1}^{n} \frac{-y_i x_i}{1 + e^{y_i(\theta^T x_i)}} + \lambda\theta)$$

   Discussion

With L2 regularization, the term $\frac{\lambda}{2}\|\theta\|^2$ penalizes larger magnitude values and forces the model to favour smaller values. This directly aids in preventing overfitting, evidenced by the smoother trajectories of θ_1 and θ_2.

6. **Repeat (5) but with backtracking. Set $\gamma = 0.9$ and $\tau = 0.5$.**

   After removing L2 regularization, the gradient updates tend to be noisier. Backtracking ensures that there is stability, by reducing the step sizes when needed by a factor of τ.

2 Stochastic Gradient Descent

Consider the ordinary linear least squares problem (without intercept) where we want to minimize $E_{x,y}[\nabla(\theta^T x - y)^2]$ with x $\sim N(0, \sum_x)$ and $y \mid x \sim N(\theta^T x, \sigma^2)$

1. Show that $E_{x,y}[\nabla_\theta[(\theta^T x - y)^2]] = \nabla_\theta E_{x,y}[(\theta^T x - y)^2]$

   Solution

   Taking reference to the mean squared error:

$$L(\theta) = E_{x,y}[(\theta^T x - y)^2]$$

   We minimise an average of functions $\min_{\mathbf{x}} \frac{1}{m}\sum_{i=1}^m f_i(x)$ such that:

$$\nabla \sum_{i=1}^{m} f_i(x) = \sum_{i=1}^{m} \nabla f_i(x)$$

   Expectation over x and y:

$$E_{x,y}[\nabla_\theta[(\theta^T x - y)^2]] = E_{x,y}[2xx^T - 2xy]$$

   $= E_x E_{y|x}[2x^T\theta - 2xy]$
   $= E_x[2xx^T\theta - 2xx^T\theta^*]$
   If we generalize for any θ, we keep:
   $= 2\Sigma_x(\theta - \theta^*)$

$$\nabla_\theta E_{x,y}[(\theta^T x - y)^2] = \nabla_\theta E_{x,y}[\theta^T xx^T\theta - 2\theta^T xy + y^2]$$

   $= \nabla_\theta E_x[\theta^T xx^T\theta] - \nabla_\theta E_{x,y}[2\theta^T xy] + \nabla_\theta E_y[y^2]$
   $= 2\Sigma_x\theta - 2\nabla_\theta E_x E_{y|x}[\theta^T xy]$
   $= 2\Sigma_x\theta - 2\nabla_\theta E_x E_{y|x}[\theta^T xx^T\theta^*]$
   $= 2\Sigma_x\theta - 2\nabla_\theta(\theta^T\Sigma_x\theta^*)$
   Using expectation properties:
   $= 2\Sigma_x\theta - 2\Sigma_x\theta^*$
   $= 2\Sigma_x(\theta - \theta^*)$

2. **How can we interpret the above in terms of SGD.**

   The SGD updates aim to approximate the true gradient of the expected loss function which is $\nabla_\theta L(\theta) = 2\Sigma_x(\theta - \theta^*)$.

   Given SGD updates using a single sample or mini batch, the resulting step is: $\theta_{t+1} = \theta_t - \alpha \cdot 2x_t(\theta_t^T x_t - y_t)$.

   The SGD will behave like an iterative process given $= \Sigma_x(\theta - \theta^*)$.

3. **Consider the univariate setting with $\Sigma_x = 1/4$, $\sigma = 1/10$, $\theta^* = 1/2$ and a data set of size 10, 000. Write an R script which plots the "confusion", i.e., the variance of the gradients, for $\theta \, \epsilon \, 0, 0.05, 0.1, ...0.95, 1.0$. For each θ, plot 200 gradient samples. Perform two such simulation studies with random batches of size 100 and 1, 000**

   Solution

   The R script generates a dataset of 10,000 samples, coputing 200 gradient samples for each θ in the range specified.

   Given the gradient $\nabla_\theta L(\theta) = 2x(\theta x - y)$, the gradient computation is estimated using random batches of sizes 100 and 1.000 and the gradient variance visualized across the different θ values.

4. **What do you observe in 3?**

   The 100 batch sizes showed higher variance throughout the graph, since they provided noisier gradient estimates. This is comparable to the 1000 batch sizes which have a lower variance, pointing to it better approximating the full dataset gradient. The variance might increase as θ deviates from θ^*, highlighting instability in gradient estimation.

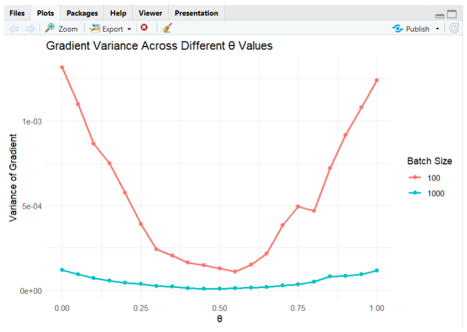

Figure 1: Gradient Variance

5. **Write an Python script or Jupyter Notebook which solves the setting in (3) with SGD with random batch sizes of 1 and $\alpha = 0.3$. Start with $\theta = 0$ and perform 20 iterations. Repeat this process 200 times. Compare with GD.**

Stochastic Gradient Descent (SGD) shows noisy, random paths indicated by the gray paths but on average converges towards θ^* shwon by the blue path. This indicates that SGD suffers from high variance despite being less computationally expensive.

In comparison, Gradient Descent (GD) red plot is smoother and more deterministic. However, since we are doing 200 iterations over 10,000 datapoints, the GD is more computationally expensive.

Justification

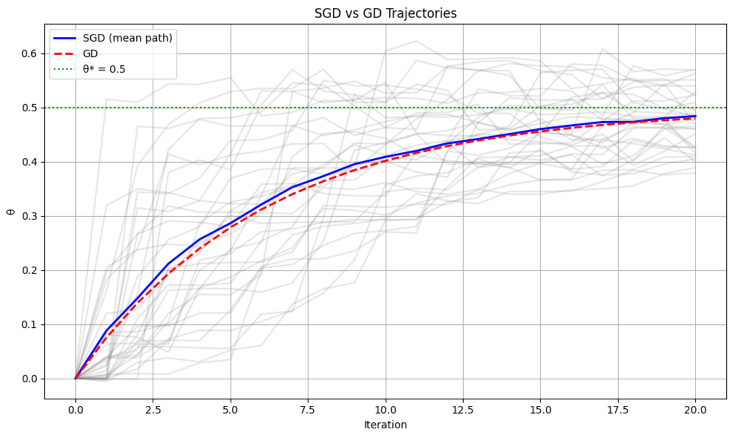

Figure 2: SGD vs GD

The high variance of update direction from SGD is because it uses random gradient estimates from a single data point, resulting in randomness in sampling. Over many iterations (20 in our case), the expected gradient approximates the true gradient, shown by the average path converging towards (θ^*).

GD uses the entire dataset to update its direction, resulting in a stable path towards the minimum. However, each step is computationally more expensive than the previous, where (O(n)) per iteration in comparison to (O(1)) per iteration for SGD.