IN4320 Machine Learning Exercise

March 17, 2016

The deadline for this exercise has not changed.

1 Exercises Stochastic Gradient Descent

Batch gradient descent iteratively updates the estimate of w at time t using the gradient ∇_w of some loss function Q. The loss function is computed over N data samples x_i and their ground truth labels y_i. At step $t+1$ it becomes

$$w_{t+1} = w_t - \lambda \frac{1}{N} \sum_{i=1}^{N} \nabla_w Q(x_i, y_i; w_t). \tag{1}$$

Stochastic gradient descent randomly picks a single example x_i to update the gradient

$$w_{t+1} = w_t - \lambda_t \nabla_w Q(x_i, y_i; w_t). \tag{2}$$

Stochastic gradient descent with mini-batches of size K randomly picks K examples to update the gradient

$$w_{t+1} = w_t - \lambda_t \frac{1}{K} \sum_{i=1}^{K} \nabla_w Q(x_i, y_i; w_t). \tag{3}$$

Note that an **iteration** denotes an update of w. Whereas an **epoch** refers to how many times the full dataset has been 'seen'.

1.1 Output

The goal is that you can convince me you have gained insight, so **always analyze your results**. It helps to show the output in the form of plots and . For example, show on the y-axis the desired output (for example the loss or the accuracy,

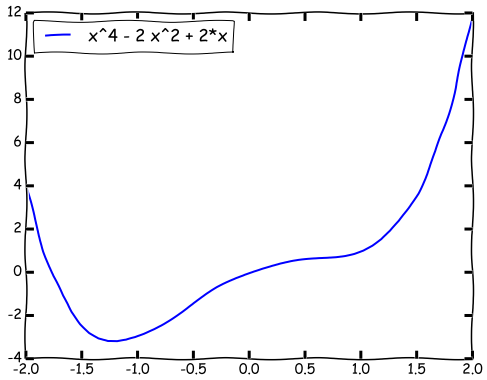

Figure 1: The polynomial $f(x) = x^4 - 2x^2 + 2x$.

computation-time), and on the x-axis the nr of samples seen or the epochs (=number of times the whole dataset has been used) For examples, see the sgd slide numbers 21, 22, 27. Code should be added in an external zip or in the appendix.

1.2 Exercises

(a) Lets start in 2D and say the polynomial function $f(x) = x^4 - 2x^2 + 2x$ in figure 1 is a loss function. The goal is to find a $w = \text{argmin}_w f(w)$ (ie: find a w for which $f(w)$ is the minimum of f). Since in this case we can compute the derivative analytically at every point, we have to simulate sampling from the gradient. First write a function `getGrad() =` $f'(x) + \mathcal{N}(0, 2)$, where $\mathcal{N}(0, 2)$ is a normal distribution: a Gaussian with $\mu = 0$ and $\sigma = 2$. For N datapoints, you would call the function `getGrad()` N times.

Implement the three GD version above: 1: Batch gradient Descent, 2: Stochastic gradient descent, 3: Stochastic gradient descent with mini-batches. Apply all 3 variants on this loss function to find its minimum. Pick a learning rate λ for each variant, and pick a suitable mini-batch size. Show epoch-loss plots with all 3 GD versions and analyze the result.

(b) Again, for the polynomial in figure 1, show the position of the first few (how many?) iterations (every update of the gradient, so not epoch) of the three methods on the function. Take care that you initialize each method at the same random location. Show 3 copies of this function plot, and indicate the point where the method goes to. Analyze the results.

(c) You can see the mini-batch variant as a generalization of the other two variants.

With $K = N$ it becomes batch gradient descent, with $K = 1$ it becomes stochastic gradient descent. For the polynomial function of figure 1 , measure the computation time for a suitable selection of K. Make sure to include $K = 1$ and $K = N$. Put the computation time on the x-axis, and the loss on the y-axis. Analyze the results. *Hint: to avoid bias in estimating computation time, it is a good idea to pre-compute all noise estimates.*

(d) Investigate the effect of the mini-batch size on the convergence. Show suitable plots and analyze the results.

(e) Investigate the effect of the learning-rate on the convergence for the 3 variants (Batch, Stochastic, mini-batch). Show suitable plots and analyze the results.

(f) Implement a learning-rate decay variant for the 3 variants. Show suitable plots and analyze the results.

(g) Now for the `optdigitsubset` from exercise 1 (Regularization & Sparsity, L2 regularization). Investigate the effect of the mini-batch size on the convergence. Show suitable plots and analyze the results. Also show the error on the test set after convergence.

(h) For the `optdigitsubset` from exercise 1 (Regularization & Sparsity, L2 regularization). For the 3 variants, investigate the effect of the learning-rate on the convergence. Show suitable plots and analyze the results. Also show the error on the test set after convergence.

(i) For the `optdigitsubset` from exercise 1 (Regularization & Sparsity, L2 regularization). For the 3 variants, investigate the effect of the regularization parameter on the convergence. Also investigate why do not we leave out the regularization term? Show suitable plots and analyze the results. Also show the error on the test set after convergence.

(j) **(bonus)** Implement momentum. Create your own, interesting, polynomial function. Investigate how well the 3 variants can handle with local minima.