11.4. Stochastic Gradient Descent

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

The objective function is usually the **average** of the **loss functions** over n training examples.

$$f(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{x})$$

- $f_i(\boldsymbol{x})$ loss function of the training data instance with index of i, and parameter vector of $\boldsymbol{x}$
- The gradient of the objective function at $\boldsymbol{x}$ is computed as

$$\nabla f(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}).$$

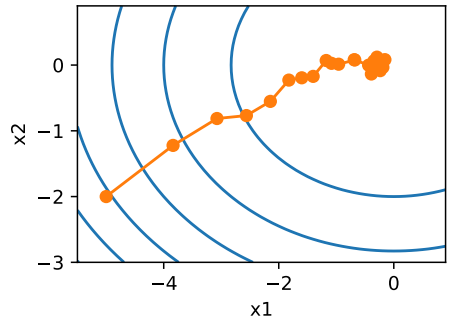
- The cost of gradient descent is $\mathcal{O}(n)$ per iteration.

- **Stochastic gradient descent** (SGD) reduces the cost at each iteration by uniformly sampling an index i and updating

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta \nabla f_i(\boldsymbol{x})$$

- The computing cost for each iteration is reduced from $\mathcal{O}(n)$ to $\mathcal{O}(1)$
- The stochastic gradient $\nabla f_i(\boldsymbol{x})$ is an unbiased estimate of gradient $\nabla f(\boldsymbol{x})$.

$$\mathbb{E}_i \nabla f_i(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}) = \nabla f(\boldsymbol{x}).$$



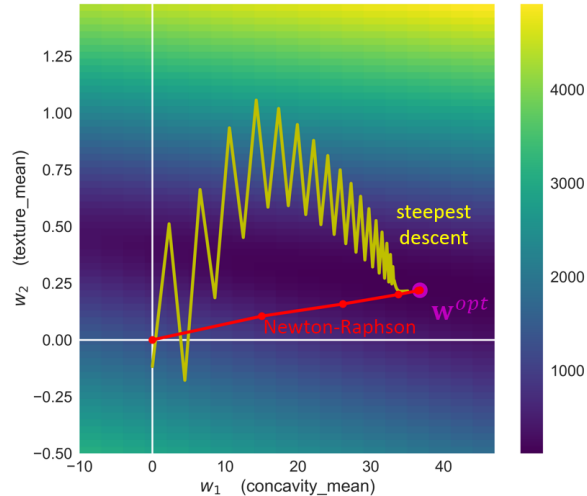
Logistic Regression

Logistic regression is a convex optimization problem that composes as a sum over training samples.

$$L(\mathbf{w}) = \underbrace{-\sum_{n \in c_1} \ln(\pi(\mathbf{x}_n \mathbf{w})) - \sum_{n' \in c_2} \ln(1 - \pi(\mathbf{x}_{n'} \mathbf{w}))}_{loss} + \underbrace{\lambda \cdot 0.5 \cdot \sum_{d=1}^{D} {w_d}^2}_{regularizer}$$

Gradient:

$$\nabla L\left(\mathbf{w}^t\right) = \begin{bmatrix} \frac{\partial L}{\partial w_1^t} \\ \vdots \\ \frac{\partial L}{\partial w_D^t} \end{bmatrix} = \underbrace{\mathbf{X}^T \left(\pi\left(\mathbf{X}\mathbf{w}^t\right) - I\left(\mathbf{y} == c_1\right)\right)}_{\nabla \text{loss}\left(\mathbf{w}^t\right)} + \underbrace{\lambda \cdot \mathbf{w}^t}_{\nabla \text{regularizer}\left(\mathbf{w}^t\right)}$$



- If we use a more suitable learning rate and update the independent variable in the opposite direction of the gradient, the value of the objective function might be reduced.

  Gradient descent repeats this update process until a solution that meets the requirements is obtained.

- Problems occur when the learning rate is too small or too large.

  A suitable learning rate is usually found only after multiple experiments.

- When there are more examples in the training data set, it costs more to compute each iteration for gradient descent, so SGD is preferred in these cases.

Mini-batch stochastic gradient descent performs random uniform sampling for each iteration to form a mini-batch and then use this mini-batch to estimate the gradient.
Set objective function $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$.

$$\boldsymbol{g}_t \leftarrow \nabla f_{\mathcal{B}_t}(\boldsymbol{x}_{t-1}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}_t} \nabla f_i(\boldsymbol{x}_{t-1})$$

In each subsequent time step $t > 0$, mini-batch SGD uses random uniform sampling to get a mini-batch $\mathcal{B}_t$ made of example indices from the training data set.
Given the learning rate η_t (positive), the update on the parameters is:

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \eta_t \boldsymbol{g}_t.$$

- Sampling of the mini-batch with replacement allows duplicate examples in the same mini-batch.
  - produces an unbiased estimate of the gradient $\nabla f(\boldsymbol{x}_{t-1})$
- Sampling without replacement does not allow duplicates.
  - has a sequential bias on small data sets.
    Consider the random variables $X_1, \ldots, X_N$ representing N draws without replacement

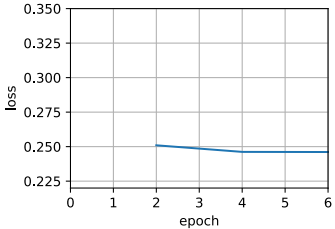
$$E[X_2|X_1 = x_j] = \frac{1}{N-1}\sum_{i \neq j} x_i = \frac{N}{N-1}\mu - \frac{1}{N-1}x_j$$

  - is most commonly used.

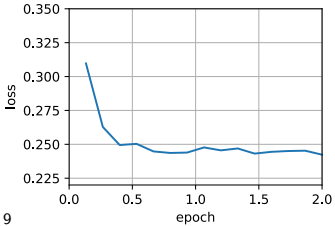
- The gradient has constant variance from random sampling.
  - In practice, the learning rate of the (mini-batch) SGD can self-decay during the iteration, such as $\eta_t = \eta t^{\alpha}$ (usually $\alpha = -1$ or -0.5), $\eta_t = \eta \alpha^t$ (e.g $\alpha = 0.95$), or learning rate decay once per iteration or after several iterations.
  - The variance of the learning rate and the (mini-batch) SGD will decrease.
- The cost for computing each iteration is $\mathcal{O}(|\mathcal{B}|)$.
  - When the batch size is small, fewer examples are used in each iteration, which will result in parallel processing and reduce the RAM usage efficiency.
  - When the batch size increases, each mini-batch gradient may contain more redundant information.
  - For larger batch sizes we typically need to increase the number of epochs.

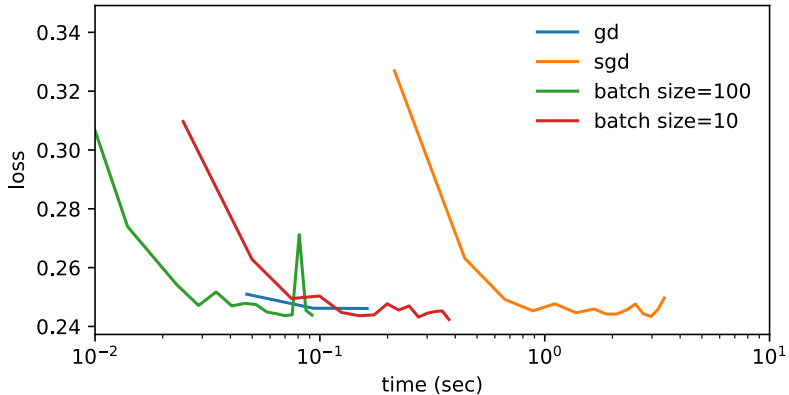
Runtime Comparison

- batch size 1500 ($= N$)



- batch size 10, learning rate 0.05





Summary

- Mini-batch stochastic gradient uses random uniform sampling to get a mini-batch training example for gradient computation.
- In practice, learning rates of the (mini-batch) SGD can self-decay during iteration.
- In general, the time consumption per epoch for mini-batch stochastic gradient is between what takes for gradient descent and SGD to complete the same epoch.