Stochastic Gradient Descent (SGD)

SGD is:

- an iterative optimization method, stochastic approximation of gradient descent
- the fundamental idea behind many machine learning algorithms such as linear support vector machines, logistic regression, graphical models
- the de facto standard training algorithm for neural networks

Given an objective function $Q_i(\boldsymbol{w})$ which computes some sort of error for sample i, e.g. $Q_i(\boldsymbol{w}) = (\hat{y}_i - y_i)^2$

Classical SGD:

Initialize parameter vector $\boldsymbol{w}$ and set learning rate η
while termination criterion not satisfied **do**
  randomly shuffle samples in training set
  for $i \leftarrow 1, n$ **do**
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla Q_i(\boldsymbol{w})$
  end for
end while

SGD Extensions

Extension 1: Batch updates

Issue: Each iteration of classical SGD evaluates the gradient at a single point x_i only

$\Rightarrow$ often results in noisy estimates of the true gradient

$\Rightarrow$ computationally expensive (blocking vectorization)

Idea: Combine multiple gradient estimates of a so-called "mini-batch", then do a single update

$\Rightarrow$ smoother convergence thanks to better estimates

$\Rightarrow$ speed-up owing to vectorization

Extension 2: Implicit updates (ISGD)

Issues:

- Classical SGD sensitive to choice of η and may easily diverge
- Better convergence with some decay schedule of η over iterations, $\eta = \eta(\text{iter})$

Idea: Evaluate stochastic gradient at the next iteration rather than the current one:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \nabla Q_i(\boldsymbol{w}_{k+1})$$

To see the effect, assuming an ordinary least squares objective

$$Q_i(\boldsymbol{w}) = \frac{1}{2}(\hat{y}_i - y_i)^2$$

in the usual samples $\{(x_i, y_i)\}_{i=1}^N$, compare the resulting update rules:

$$\text{Classical SGD:} \quad \boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta(y_i - \boldsymbol{x}_i^\top \boldsymbol{w}_k)\boldsymbol{x}_i$$

$$\text{Implicit SGD (closed-form!):} \quad \boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \frac{\eta}{1 + \eta \left\|\boldsymbol{x}_i\right\|^2}\left(y_i - \boldsymbol{x}_i^\top \boldsymbol{w}_k\right)\boldsymbol{x}_i$$

Note the normalizing effect on η, resulting in numerically stable procedure for virtually all η

Extension 3: Momentum

Issue: Classical SGD frequently shows oscillations, hampering convergence

Idea: Introduce some "memory" $\Delta \boldsymbol{w}_k$ of the current direction, i.e., do not take "hard turns" at every step:

$$\Delta \boldsymbol{w}_{k+1} = -\eta \nabla Q_i(\boldsymbol{w}_k) + \alpha \Delta \boldsymbol{w}_k$$

resulting in the update rule

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \Delta \boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \nabla Q_i(\boldsymbol{w}_k) + \alpha \Delta \boldsymbol{w}_k$$

with $\alpha \in [0, 1)$ an exponential decay factor (hyperparameter)