Batch Gradient Descent

\[ L_T(w) = \frac{1}{N} \sum_{n=1}^{N} \ell_n(w) \]

- \( \nabla L_T(w) = \frac{1}{N} \sum_{n=1}^{N} \nabla \ell_n(w) \).

- Taking a macro-step \(-\alpha \nabla L_T(w_t)\) is the same as taking the \(N\) micro-steps \(-\frac{\alpha}{N} \nabla \ell_1(w_t), \ldots, -\frac{\alpha}{N} \nabla \ell_N(w_t)\).

- First compute all the \(N\) steps at \(w_t\), then take all the steps.

- Thus, standard gradient descent is a *batch* method:
  Compute the gradient at \(w_t\) using the entire batch of data, then move.

- Even with no line search, computing \(N\) micro-steps is still expensive.
Stochastic Descent

- Taking a macro-step $-\alpha \nabla L_T(w_t)$ is the same as taking the $N$ micro-steps $-\frac{\alpha}{N} \nabla \ell_1(w_t), \ldots, -\frac{\alpha}{N} \nabla \ell_N(w_t)$
- First compute all the $N$ steps at $w_t$, then take all the steps
- Can we use this effort more effectively?
- Key observation: $-\nabla \ell_n(w)$ is a poor estimate of $-\nabla L_T(w)$, but an estimate all the same: Micro-steps are correct on average!
- After each micro-step, we are on average in a better place
- How about computing a new micro-gradient after every micro-step?
- Now each micro-step gradient is evaluated at a point that is on average better (lower risk) than in the batch method
Stochastic Gradient Descent

Batch versus Stochastic Gradient Descent

- \( s_n(w) = -\frac{\alpha}{N} \nabla \ell_n(w) \)

- **Batch:**
  - Compute \( s_1(w_t), \ldots, s_N(w_t) \)
  - Move by \( s_1(w_t) \), then \( s_2(w_t) \), \ldots then \( s_N(w_t) \)
    (or equivalently move once by \( s_1(w_t) + \ldots + s_N(w_t) \))

- **Stochastic (SGD):**
  - Compute \( s_1(w_t) \), then move by \( s_1(w_t) \) from \( w_t \) to \( w_{t}^{(1)} \)
  - Compute \( s_2(w_{t}^{(1)}) \), then move by \( s_2(w_{t}^{(1)}) \) from \( w_{t}^{(1)} \) to \( w_{t}^{(2)} \)
  - \( \vdots \)
  - Compute \( s_N(w_{t}^{(N-1)}) \), then move by \( s_N(w_{t}^{(N-1)}) \) from \( w_{t}^{(N-1)} \)
  - to \( w_{t}^{(N)} = w_{t+1} \)

- In SGD, each micro-step is taken from a better (lower risk) place on average
Why “Stochastic?”

- Progress occurs only on average
- Many micro-steps are bad, but they are good on average
- Progress is a random walk
Reducing Variance: Mini-Batches

- Each data sample is a poor estimate of $T$: High-variance micro-steps
- Each micro-step take full advantage of the estimate, by moving right away: Low-bias micro-steps
- High variance may hurt more than low bias helps
- Can we lower variance at the expense of bias?
- Average $B$ samples at a time: Take \textit{mini-steps}
- With bigger $B$,
  - Higher bias
  - Lower variance
- The $B$ samples are a \textit{mini-batch}
Mini-Batches

- Scramble $T$ at random
- Divide $T$ into $J$ mini-batches $T_j$ of size $B$
- $w^{(0)} = w$
- For $j = 1, \ldots, J$:
  - Batch gradient:
    \[ g_j = \nabla L_{T_j}(w^{(j-1)}) = \frac{1}{B} \sum_{n=(j-1)B+1}^{jB} \nabla \ell_n(w^{(j-1)}) \]
  - Move: $w^{(j)} = w^{(j-1)} - \alpha g_j$
- This for loop amounts to one macro-step
- Each execution of the entire loop uses the training data once
- Each execution of the entire loop is an epoch
- Repeat over several epochs until a stopping criterion is met
Momentum

- Sometimes $\mathbf{w}^{(j)}$ meanders around in shallow valleys
  - No $\alpha$ adjustment here
  
- $\alpha$ is too small, direction is still promising
- Add momentum
  
$$
\begin{align*}
\mathbf{v}_0 &= 0 \\
\mathbf{v}^{(j+1)} &= \mu^{(j)} \mathbf{v}^{(j)} - \alpha \nabla L_T(\mathbf{w}^{(j)}) \\
\mathbf{w}^{(j+1)} &= \mathbf{w}^{(j)} + \mathbf{v}^{(j+1)}
\end{align*}
$$

$$(0 \leq \mu^{(j)} < 1)$$
Regularization

- The capacity of deep networks is very high: It is often possible to achieve near-zero training loss
- “Memorize the training set” ⇒ overfitting
- All training methods use some type of regularization
- Regularization can be seen as *inductive bias*: Bias the training algorithm to find weights with certain properties
- Simplest method: *weight decay*, add a term $\lambda \|W\|^2$ to the risk function: Keep the weights small (Tikhonov)
- Many proposals have been made
- Not yet clear which method works best, a few proposals follow
Early Termination

- Terminating training well before the $L_T$ is minimized is somewhat similar to “implicit” weight decay
- Progress at each iteration is limited, so stopping early keeps us close to $w_0$, which is a set of small random weights
- Therefore, the norm of $w_t$ is restrained, albeit in terms of how long the learner takes to get there rather than in absolute terms
- A more informed approach to early termination stops when a validation risk (or, even better, error rate) stops declining
- This (with validation check) is arguably the most widely used regularization method.
Dropout

- *Dropout* inspired by ensemble methods:
  Regularize by averaging multiple predictors
- Key difficulty: It is too expensive to train an ensemble of deep neural networks
- Efficient (crude!) approximation:
  - Before processing a new mini-batch, flip a coin with $\mathbb{P}[\text{heads}] = p$ (typically $p = 1/2$) for each neuron
  - Turn off the neurons for which the coin comes up tails
  - Restore all neurons at the end of the mini-batch
  - When training is done, multiply all weights by $p$
- This is very loosely akin to training a different network for every mini-batch
- Multiplication by $p$ takes the “average” of all networks
- There are flaws in the reasoning, but the method works
Regularization

(a) Standard Neural Net  (b) After applying dropout.
Data Augmentation

- Data augmentation is not a regularization method, but combats overfitting
- Make new training data out of thin air
- Given data sample \((x, y)\), create perturbed copies \(x_1, \ldots, x_k\) of \(x\) (these have the same label!)
- Add samples \((x_1, y), \ldots, (x_k, y)\) to training set \(T\)
- With images this is easy. The \(x_i\)s are cropped, rotated, stretched, re-colored, \(\ldots\) versions of \(x\)
- One training sample generates \(k\) new ones
- \(T\) grows by a factor of \(k + 1\)
- Very effective, used almost universally
- Need to use realistic perturbations