Review
Our running example for the loss minimization problem

\[
\arg\min_w \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} (y_n - w \cdot x_n)^2 + \frac{1}{2} \lambda \|w\|^2
\]

- How do we run GD/SGD?

- how do we set the step size? \( \lambda \)? the “mini-batch” size?

We will help with guidance/understanding/theory. Ultimately, we just have try to tune these ourselves to get experience. HW3 will help...
review: GD for the square loss

Data: step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$

Result: parameter $w$

initialize: $w^{(0)} = 0$;

for $k \in \{1, \ldots, K\}$ do

\[ w^{(k)} = w^{(k-1)} + \eta^{(k)} \left( \frac{1}{N} \sum_n (y_n - w^{(k-1)} \cdot x_n) x_n \right) \]

end

return $w^{(K)}$;

Algorithm 1: SGD

- the term in red is a costly to compute!
- Even by using matrix multiplications (and not explicitly doing the sum), it is often too slow.
Gradient Descent: review

- how do we set the stepsize?
  - Remember: we diverge if the step size is too big!
  - you just set it a little lower (like 1/2) less than when things start to diverge.

- do we decay it?
  No: GD will converge just fine without decaying the learning rate.

- Is GD a good algorithm?
  GD is often too slow:
  - computing the gradient of the objective function involves a sum over

- Today: SGD/let’s sample the gradient!
Today
SGD for the square loss

**Data**: step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$

**Result**: parameter $w$

initialize: $w^{(0)} = 0$;

**for** $k \in \{1, \ldots, K\}$ **do**

| Sample $n \sim \text{Uniform}\{1, \ldots, N\}$; 
| $w^{(k)} = w^{(k-1)} + \eta^{(k)}(y_n - w^{(k-1)} \cdot x_n) \cdot x_n$; |

end

return $w^{(K)}$;

**Algorithm 2**: SGD

- the term in red is a “sampled” gradient.
“mini-batch” SGD for the square loss

Data: step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$

Result: parameter $w$

initialize: $w^{(0)} = 0$

for $k \in \{1, \ldots, K\}$ do

Sample $m$ examples of $(x, y)$ (uniformly at random) from the training set and let $\mathcal{M}$ be the set of these $m$ points;

$w^{(k)} = w^{(k-1)} + \eta^{(k)} \frac{1}{m} \sum_{(x, y) \in \mathcal{M}} (y - w^{(k-1)} \cdot x) \cdot x$;

end

turn $w^{(K)}$;

Algorithm 3: SGD

- the term in red is a lower variance, “sampled” gradient.
- how do we choose $m$?
  larger $m$ means lower variance but more computation.
- Matrix algebra can make computing the term in red very fast!
  This is critical to get big performance bumps.

$$L(w) = \frac{1}{n} \sum_{n} (y - w \cdot x)^2$$
SGD: How do we set the step sizes?

- Theory: If you turn down the step sizes at (some prescribed decaying method) then SGD will converge to the right answer. The “classical” theory doesn’t provide enough practical guidance.

- Practice:
  - starting stepsize: start it “large”:
    - if it is “too large”, then either you diverge (or nothing improves). set it a little less (like 1/4) less than this point.
  - When do we decay it?
    - When your training error stops decreasing “enough”.

- HW: you'll need to tune it a little. (a slow approach: sometimes you can just start it somewhat smaller than the “divergent” value and you will find something reasonable.)
SGD: How do we set the mini-batch size $m$?

- **Theory:** there are diminishing returns to increasing $m$.
  - As you grow $m$, your “improvements” tend to diminish.
  - mini-batch size $m$ “small”: you can turn it up and you will find that you are making more progress per update.
  - mini-batch size $m$ “large”: you can turn it up and you will make roughly the same amount of progress (so your code will become slower).
- **Practice:** there are diminishing returns to increasing $m$.
- **How do we set it?**
  Easy: just keep cranking it up and eventually you’ll see that your code doesn’t get any faster.
“regularized” SGD for the square loss, $m = 1$

Data: step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$

Result: parameter $w$

Initialize: $w^{(0)} = 0$;

For $k \in \{1, \ldots, K\}$ do

- Sample $n \sim \text{Uniform}\{1, \ldots, N\}$;
- $w^{(k)} = w^{(k-1)} + \eta^{(k)} \left( (y_n - w^{(k-1)} \cdot x_n) x_n - \lambda w \right)$;

End

Return $w^{(K)}$;

Algorithm 4: SGD

- Regularization has been added: How do we set it?
- We can combine this with mini-batching
Regularization: How do we set it?

- Theory: really just says that $\lambda$ controls your “model complexity”.
  - we DO know that “early stopping” for GD/SGD is (basically) doing L2 regularization for us
  - i.e. if we don’t run for too long, then $\|w\|^2$ won’t become too big.

- Practice:
  - Exact methods (like matrix inverse/least squares): always need to regularize or something horrible happens....
  - GD/SGD: sometimes (often ?) it works just fine ignoring regularization
  - Other times we have to tune it on some dev set.
    Fortunately, it is pretty robust to tune, by trying out different “orders of magnitude” guesses.