Gradient and Stochastic Gradient Descent

David S. Rosenberg

Bloomberg ML EDU

September 29, 2017

Gradient Descent

Setting

Objective function $f : \mathbf{R}^d \to \mathbf{R}$ is *differentiable.*
Want to find

$$x^* = \arg\min_{x \in \mathbf{R}^d} f(x)$$

The Gradient

- Let $f : \mathbf{R}^d \to \mathbf{R}$ be differentiable at $x_0 \in \mathbf{R}^d$.
- The **gradient** of f at the point x_0, denoted $\nabla_x f(x_0)$, is the direction to move in for the **fastest increase** in $f(x)$, when starting from x_0.

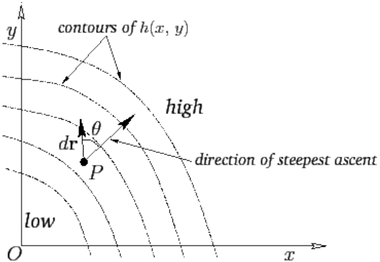


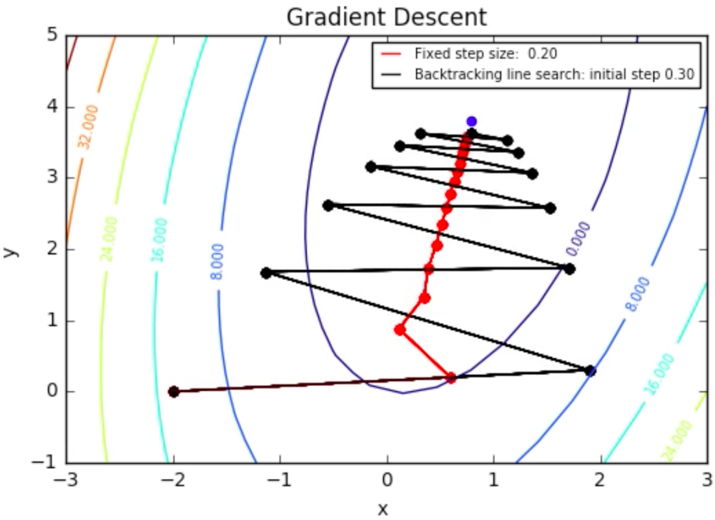
Figure A.111 from Newtonian Dynamics, by Richard Fitzpatrick.

Gradient Descent

Gradient Descent

- Initialize $x = 0$
- repeat
  - $x \leftarrow x - \underbrace{\eta}_{\text{step size}} \nabla f(x)$
- until stopping criterion satisfied

Gradient Descent Path



Gradient Descent: Step Size

- A fixed step size will work, eventually, as long as it's small enough (roughly - details to come)
  - Too fast, may diverge
  - In practice, try several fixed step sizes
- Intuition on when to take big steps and when to take small steps?
  - Demo.

Convergence Theorem for Fixed Step Size

Theorem

*Suppose $f : \mathbf{R}^d \to \mathbf{R}$ is convex and differentiable, and ∇f is **Lipschitz continuous** with constant $L > 0$, i.e.*

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$$

*for any $x, y \in \mathbf{R}^d$. Then gradient descent with fixed step size $t \leqslant 1/L$ **converges**. In particular,*

$$f(x^{(k)}) - f(x^*) \leqslant \frac{\|x^{(0)} - x^*\|^2}{2tk}.$$

Step Size: Practical Note

- Although a $1/L$ step-size guarantees convergence,
  - it may be **much slower** than necessary.
- May be worth trying larger step sizes as well.
- But math tells us, no need for anything smaller.

Gradient Descent: Questions to Ponder

- "Empirically $\eta = 0.1$ often works well" (says an ML textbook)
- How can one rate work well for most functions?
- Suppose $\eta = 0.1$ works well for $f(x)$, what about $g(x) = f(10x)$?
  - Do we want bigger steps or smaller steps?
  - How does the magnitude of the gradient compare between $g(x)$ and $f(x)$?
  - How does the Lipschitz constant compare between $g(x)$ and $f(x)$?

Backtracking Line Search

- If we step in negative gradient direction, $\|\nabla f(x)\|$ gives us rate of decrease.
  - at least for infinitesimally small step size.
- Find step size that gives at least some fixed fraction of instantaneous rate of decrease.
- We'll discuss **backtracking line search,** based on this idea, in the Lab.

Gradient Descent: When to Stop?

- Wait until $\|\nabla f(x)\|_2 \leqslant \varepsilon$, for some ε of your choosing.
  - (Recall $\nabla f(x) = 0$ at minimum.)
- For learning setting,
  - evalute performance on validation data as you go
  - stop when not improving, or getting worse

Gradient Descent for Empirical Risk (And Other Averages)

Setup

- Input space $\mathcal{X} = \mathbf{R}^d$
- Output space $\mathcal{Y} = \mathbf{R}$
- Action space $\mathcal{Y} = \mathbf{R}$
- Loss: $\ell(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$
- **Hypothesis space:** $\mathcal{F} = \left\{ f : \mathbf{R}^d \to \mathbf{R} \mid f(x) = w^T x,\, w \in \mathbf{R}^d \right\}$

- Given data set $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$,
  - Let's find the ERM $\hat{f} \in \mathcal{F}$.

Objective Function: Empirical Risk

The function we want to minimize is the empirical risk:

$$\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n}\left(w^T x_i - y_i\right)^2,$$

where $w \in \mathbf{R}^d$ parameterizes the hypothesis space $\mathcal{F}$.

- Now let's think more generally...

Gradient Descent for Empirical Risk and Averages

- Suppose we have a hypothesis space of functions $\mathcal{F} = \left\{ f_w : \mathcal{X} \to \mathcal{A} \mid w \in \mathbf{R}^d \right\}$
  - Parameterized by $w \in \mathbf{R}^d$.
- ERM is to find w minimizing

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i)$$

- Suppose $\ell(f_w(x_i), y_i)$ is differentiable as a function of w.
- Then we can do gradient descent on $\hat{R}_n(w)$...

Gradient Descent: How does it scale with n?

- At every iteration, we compute the gradient at current w:

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell(f_w(x_i), y_i)$$

- We have to touch all n training points to take a single step. [$O(n)$]
- Will this scale to "big data"?
- Can we make progress without looking at all the data?

"Noisy" Gradient Descent

- We know gradient descent works.
- But the gradient may be slow to compute.
- What if we just use an estimate of the gradient?
- Turns out that can work fine.
- **Intuition**:
  - Gradient descent is an interative procedure anyway.
  - At every step, we have a chance to recover from previous missteps.

Minibatch Gradient

- The **full gradient** is

$$\nabla \hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla_w \ell(f_w(x_i), y_i)$$

- It's an average over the **full batch** of data $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Let's take a subsample of size N:

$$(x_{m_1}, y_{m_1}), \ldots, (x_{m_N}, y_{m_N})$$

- The **minibatch gradient is**

$$\nabla \hat{R}_N(w) = \frac{1}{N}\sum_{i=1}^{N} \nabla_w \ell(f_w(x_{m_i}), y_{m_i})$$

- What can we say about the minibatch gradient?

- What's the expected value of the **minibatch gradient**?

$$
\begin{aligned}
\mathbb{E}\left[\nabla \hat{R}_N(w)\right] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\nabla_w \ell(f_w(x_{m_i}), y_{m_i})\right] \\
&= \mathbb{E}\left[\nabla_w \ell(f_w(x_{m_1}), y_{m_1})\right] \\
&= \sum_{i=1}^{n}\mathbb{P}(m_1 = i)\,\nabla_w \ell(f_w(x_i), y_i) \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla_w \ell(f_w(x_{m_i}), y_{m_i}) \\
&= \nabla \hat{R}_n(w)
\end{aligned}
$$

Minibatch Gradient Properties

- Minibatch gradient is an **unbiased estimator** for the [full] batch gradient:

$$\mathbb{E}\left[\nabla \hat{R}_N(w)\right] = \nabla \hat{R}_n(w)$$

- The bigger the minibatch, the better the estimate.

Minibatch Gradient – In Practice

- Tradeoffs of minibatch size:
  - Bigger $N \implies$ Better estimate of gradient, but slower (more data to touch)
  - Smaller $N \implies$ Worse estimate of gradient, but can be quite fast
- Even $N = 1$ works, it's called **stochastic gradient descent** (SGD).

Terminology Review

- **Gradient descent** or **"batch" gradient descent**
  - Use full data set of size n to determine step direction
- **Minibatch gradient descent**
  - Use a random subset of size N to determine step direction
  - Yoshua Bengio says[1]:
    - N is typically between 1 and few hundred
    - $N = 32$ is a good default value
    - With $N \geqslant 10$ we get computational speedup (per datum touched)
- **Stochastic gradient descent**
  - Minibatch with $m = 1$.
  - Use a single randomly chosen point to determine step direction.

[1] See Yoshua Bengio's "Practical recommendations for gradient-based training of deep architectures" `http://arxiv.org/abs/1206.5533`.

Minibatch Gradient Descent

Minibatch Gradient Descent (minibatch size N)

- initialize $w = 0$
- repeat
  - randomly choose N points $\{(x_i, y_i)\}_{i=1}^{N} \subset \mathcal{D}_n$
  - $w \leftarrow w - \eta \left[\frac{1}{N} \sum_{i=1}^{N} \nabla_w \ell(f_w(x_i), y_i) \right]$

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent

- initialize $w = 0$
- repeat
  - randomly choose training point $(x_i, y_i) \in \mathcal{D}_n$
  - $w \leftarrow w - \eta \underbrace{\nabla_w \ell(f_w(x_i), y_i)}_{\text{Grad(Loss on i'th example)}}$

Step Size

- For SGD, fixed step size can work well in practice, but no theorem.
- For convergence guarantee, use decreasing step sizes (dampens noise in step direction).
- Let η_t be the step size at the t'th step.

Robbins-Monro Conditions

Many classical convergence results depend on the following two conditions:

$$\sum_{t=1}^{\infty} \eta_t^2 < \infty \qquad \sum_{t=1}^{\infty} \eta_t = \infty$$

- As fast as $\eta_t = O\left(\frac{1}{t}\right)$ would satisfy this... but should be faster than $O\left(\frac{1}{\sqrt{t}}\right)$.
- A useful reference for practical techniques: Leon Bottou's "Tricks": `http://research.microsoft.com/pubs/192769/tricks-2012.pdf`