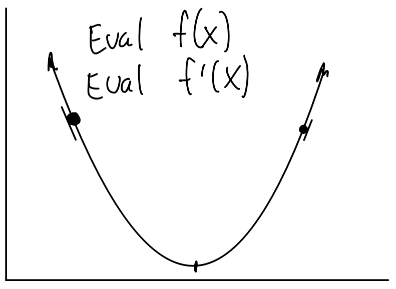
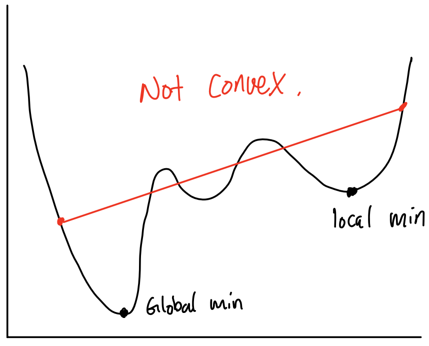
Gradient Descent:



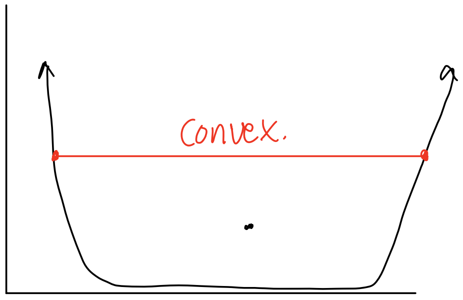
Goal: Find the minimal of this function.

If $f'(x) < 0$, move a bit to right
If $f'(x) > 0$, move a bit to left.
If $f'(x) = 0$, stop at output x.



Convexity: A function is convex if the chord connecting any 2 points of the graph lies above the function.

Equivalent definition: A function is convex if $f(ax_1 + (1-a)x_2) \leq a f(x_1) + (1-a) f(x_2)$



let's say x^* is the global min, and we are currently at x,

$$f(ax + (1-a)x^*) \leq a f(x) + (1-a) f(x^*)$$

Another Example:

$f(x) = w^T x + b$ (linear function in d dimensions).

currently at point x. We want to know what directions should we move in to minimize f?

By direction we mean unit vector, u.

$f(x+u) = w^T x + \underbrace{w^T u}_{} + b$

choice of u affects this middle term.

correct choice of $u = \frac{-w}{\|w\|}$

If we move in direction $\frac{-w}{\|w\|}$ then f decreases by $\|w\|_2$. Note: $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

Thus far, our idea has been to look at tangent lines and this idea works for, say, linear functions and simple convex functions.

Even if we want to minimize more complicated functions, assume they are "locally" linear.

f at point x: Taylor Expansion,

$$f(x+\epsilon) = \underbrace{f(x) + \epsilon f'(x)}_{\text{linear function of } \epsilon} + \underbrace{\frac{\epsilon^2}{2} f''(x) + \dots \frac{\epsilon^3}{3!} f'''(x) + \dots +}_{\text{when } \epsilon \text{ is small, these terms are negligible.}}$$

Taylor's Theorem also holds in d dimensions

Instead of taking derivatives (univariate case) for higher dimensions we must look at gradients.

The gradient of f at point x:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_d}(x) \right)$$ ← d dimensional vector.

$f = w^T x + b \Rightarrow \nabla f = w \qquad \frac{\partial f}{\partial x_i} = w_i$

<u>Another Example:</u>

$f(x) = x^T A x - b^T x$ (n - dimension)

$$f(x) = \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j}_{\star} - \sum_{i=1}^{n} b_i x_i$$

$$\frac{\partial f}{\partial x_k} = \underbrace{\sum_{j=1}^{n} a_{kj} x_j}_{\substack{k^{th} \text{ row of } A \\ \text{inner with } x}} + \underbrace{\sum_{i=1}^{n} a_{ik} x_i}_{\substack{k^{th} \text{ column of } A \\ \text{inner with } x}} - b_k$$

Case 1: $i = k$ (when $i = k$ and $j = k$ in $\star$ $a_{kk} x_k^2$)

$$\frac{\partial (a_{kk} x_k^2)}{\partial x_k} = 2 a_{kk} x_k$$

Case 2: $i \neq k$

we won't have $a_{kk} x_k$ term because $i \neq k$.

Answer: $\boxed{Ax + A^T x - b}$ ← gradient at point x.

If A is symmetric, $2Ax - b$.

Gradient of $f(x) = x^T A x - b^T x$.

Define Gradient Descent:

Initially we'll choose w randomly (want to minimize $f(w)$)

If $\underbrace{\|\nabla f(w)\|}_{\text{gradient wrt } \vec{w}}{}_2 < \epsilon$, stop and output w.

2 norm

otherwise $w_{new} = w_{old} - \eta \nabla f(w)$

step size parameter, usually relatively small.

Coordinate-wise:

$$w_j^{new} = w_j^{old} - \eta \frac{\partial f}{\partial w_j}(w)$$

vectors

moving in the opposite direction of the gradient, to make smaller.

Apply it to linear Regression:

$h(x) = w^T x + b$ (searching for this function).

(we have a training set of size m).

$$MSE(w) = \frac{1}{m} \sum_{j=1}^{m} \underbrace{(w^T x^i + b - y^i)^2}_{g_j}$$

$$\frac{\partial g_j}{\partial w_i} = 2 \cdot (w^T x^j + b - y^j) x_i^j \quad \text{(coordinate version)}$$

$$\nabla g_j(w) = 2 (w^T x^j + b - y^j) x^j \quad \text{(complete version)}.$$

$$\boxed{\nabla MSE(w) = \frac{2}{m} \cdot \sum_{j=1}^{m} (w^T x^j + b - y^j) \cdot x^j}$$

wrt $\vec{w}$

convex function

sum over m, then divide by m → average.

Runtime : $O(m \cdot n)$.
can be massively parallelized. | $w_{new} = w_{old} - \eta \nabla MSE(w)$.

Derivation:

let $v(w) = X \cdot w - y$

let $f(v) = |v(w)|_2^2 \Rightarrow$

$$\frac{\partial(|X \cdot w - y|_2^2)}{\partial w} = \frac{\partial f(v)}{\partial v} \frac{\partial v}{\partial w} \quad \text{(chain rule)}.$$

Note: $f(v) = v_1^2 + v_2^2 + \ldots + v_m^2$,

1) $\frac{\partial f}{\partial v_1} = 2v_1 \Rightarrow \frac{\partial f(v)}{\partial v} = 2v^T$. (tranpose from numerator layout notation)

2) $\frac{\partial v}{\partial w}$:

Let $X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}$, x_i = column vector

$\Rightarrow v = w_1 x_1 + \cdots w_n x_n - y$ and $\frac{\partial v}{\partial w_i} = x_i$

$\Rightarrow \frac{\partial v}{\partial w} = [x_1, x_2, \ldots x_n] = X$

Putting it together: $\frac{\partial f}{\partial w} = \frac{\partial f}{\partial v}\frac{\partial v}{\partial w}$

$= 2v^T X = 2(X \cdot w - y)^T X$. Set $= 0$

$\Rightarrow (X \cdot w - y)^T X = [0, \ldots 0]$

$$= X^T(X\cdot w - y) = [0, \dots, 0]^T \qquad \text{Note: } (AB)^T = B^T A^T$$

$$\Rightarrow X^T X \cdot w = X^T y$$

$$\Rightarrow \boxed{w = (X^T X)^{-1} X^T y}$$

Stochastic Gradient Descent:

average

- Previously in the linear regression example, we summed over all points in the training set.
- Choose index j at random, compute the gradient wrt this point only:

$$W_{new} = W_{old} - 2 \cdot \eta \left(w^T x^j + b - y^j\right) x^j.$$

Why does it work? Each point chosen with equal probability.

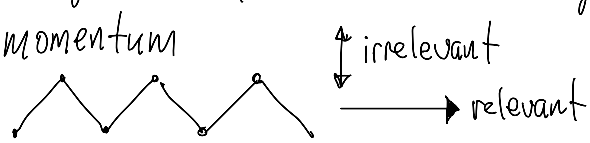
$$E\left[W_{new}\right] = W_{old} - 2\eta \frac{1}{m} \sum_{j=1}^{m} \left(w^T x^j + b - y^j\right) x^j$$

Batch Gradient Descent: In btwn the 2 types.

- Use "batches" to interpolate btwn gradient descent and pure SGD.
- Reduces the variance.

How to choose η (step size):

- more art than science; use cross validation to pick η.
- Many techniques for adaptively choosing η
- momentum



the positives and negatives in the y direction averages out to ~ 0, so you end up moving in x.

momentum has a velocity variable V.

$V_0 = 0$

$\underbrace{V_i = \alpha V_{i-1} - \eta g_i}$ gradient computed at the i^{th} step

This takes a weighted exponential moving average of $-\eta g_i$'s.

$W_{new} = W_{old} + V_i$

Accelerated Gradient Descent: Beyond scope.

Read ch 14 of ML book.