Data Analysis: Statistical Modeling and Computation in Applications

Gradient Descent

Outline

- Convex functions
- Gradient descent: main scheme
- Direction
- Step size
- Convergence
- Stochastic Gradient Descent

Recall: Ordinary least Squares Estimator (OLS)

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \underbrace{\mathbf{x}_i \boldsymbol{w}}_{\hat{y}_i})^2$$

- Today: general method for finding minima

Notation: to avoid confusion with "typical" notation in optimization vs statistics, for today's lecture, we replaces $\boldsymbol{\beta}$ by $\boldsymbol{w}$: $\boldsymbol{w}$ are the parameters to optimize.

Recall: Ordinary least Squares Estimator (OLS)

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \mathbf{x}_i \boldsymbol{w})^2$$

- Today: general method for finding minima
- Recall:
  setting derivative to zero gives *normal equations* and closed form $\hat{\boldsymbol{w}}$

Notation: to avoid confusion with "typical" notation in optimization vs statistics, for today's lecture, we replaces $\boldsymbol{\beta}$ by $\boldsymbol{w}$: $\boldsymbol{w}$ are the parameters to optimize.

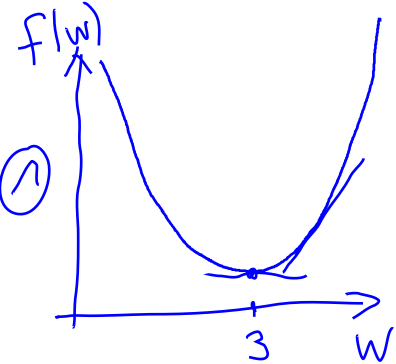
Recall: Ordinary least Squares Estimator (OLS)

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \mathbf{x}_i \boldsymbol{w})^2$$

- Today: general method for finding minima
- Recall:
  setting derivative to zero gives *normal equations* and closed form $\hat{\boldsymbol{w}}$
- *When does setting the derivative to zero give the minimum?*

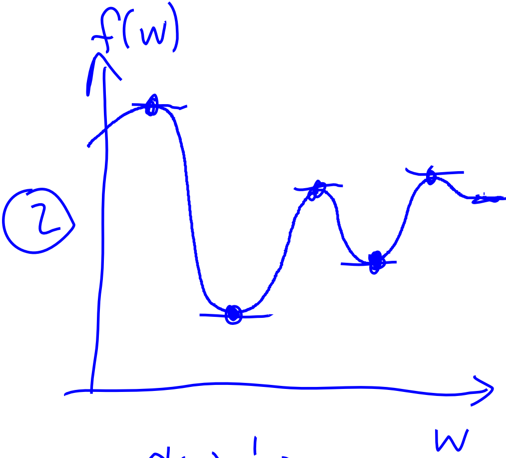
Notation: to avoid confusion with "typical" notation in optimization vs statistics, for today's lecture, we replaces $\boldsymbol{\beta}$ by $\boldsymbol{w}$: $\boldsymbol{w}$ are the parameters to optimize.

Intuition



$f(w) = (3-w)^2$ ✓

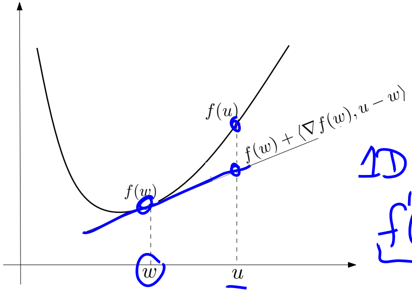
convex



$f'(w) \stackrel{!}{=} 0$ ✗

non-convex

Convexity: “bowl-shapedness”

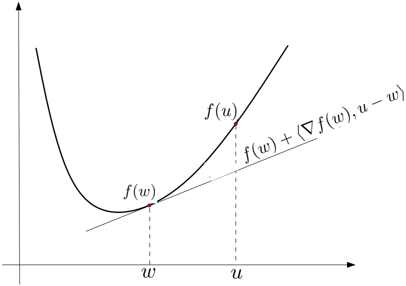


1D: $f(w) + \underbrace{f'(w)} \cdot (u - w)$

Function f is *convex* if at each point, the gradient gives a linear lower bound, i.e., for all u, w:

$$f(u) \geq \underbrace{f(w) + \langle \nabla f(w), u - w \rangle}.$$

Convexity: "bowl-shapedness"

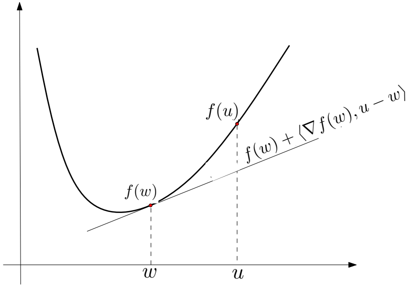


Function f is *convex* if at each point, the gradient gives a linear lower bound, i.e., for all u, w:

$$f(u) \geq f(w) + \underbrace{\langle \nabla f(w), u - w \rangle}_{=0}.$$

- if $\nabla f(w) = 0$ (local property), then w is a *global minimum* (global): $f(u) \geq f(w) + \langle 0, u - w \rangle$ for all u. Not for non-convex functions!

Convexity: “bowl-shapedness”

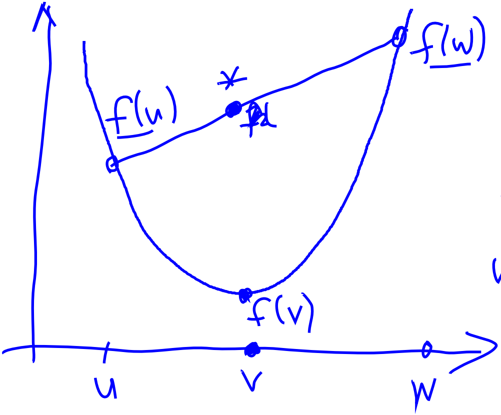


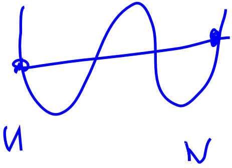
Function f is *convex* if at each point, the gradient gives a linear lower bound, i.e., for all u, w:

$$f(u) \geq f(w) + \langle \nabla f(w), u - w \rangle.$$

- if $\nabla f(w) = 0$ (local property), then w is a *global minimum* (global): $f(u) \geq f(w) + \langle 0, u - w \rangle$ for all u. Not for non-convex functions!
- Hence, for convex f, all we need to find is a point with $\nabla f(w) = 0$.

Convexity: “chord across bowl”





- *“chord across bowl”:* f is convex if for all u, w and $0 \leq \lambda \leq 1$:

$$f\left(\underbrace{\lambda w + (1-\lambda)u}_{v}\right) \leq \underbrace{\lambda f(w) + (1-\lambda) f(u)}_{*}$$

Convexity: 3 criteria

1. "linear lower bound"
2. "chord across bowl"

Convexity: 3 criteria

1. "linear lower bound"
2. "chord across bowl"
3. if f is twice differentiable: f is convex if for all w
   $\nabla^2 f(w)$ is positive semidefinite (in 1D: $f''(w) \geq 0$).

A is psd if $v^T A v \geq 0$

$\forall v$

Examples of convex functions

scalar	**vector**	
$f(w) = bw$	$f(w) = b^\top w$	for constant b

Examples of convex functions

scalar	**vector**	
$f(w) = bw$	$f(w) = b^\top w$	for constant b
$f(w) = aw^2$	$f(w) = w^\top \mathbf{A} w$	for constant $a \geq 0$, $\mathbf{A}$ positive semidefinite

Examples of convex functions

scalar	**vector**	
$f(w) = bw$	$f(w) = b^\top w$	for constant b
$f(w) = aw^2$	$f(w) = w^\top \mathbf{A} w$	for constant $a \geq 0$, $\mathbf{A}$ positive semidefinite
$f(w) = \exp(bw)$	$f(w) = \exp(b^\top w)$	for constant b

Examples of convex functions

scalar	**vector**	
$f(w) = bw$	$f(w) = b^\top w$	for constant b
$f(w) = aw^2$	$f(w) = w^\top \mathbf{A} w$	for constant $a \geq 0$, $\mathbf{A}$ positive semidefinite
$f(w) = \exp(bw)$	$f(w) = \exp(b^\top w)$	for constant b

If $f(w)$ and $g(w)$ are convex functions, then $f(w) + g(w)$ is convex.
$\Rightarrow$ least squares loss is convex!

Outline

- Convex functions
- Gradient descent: main scheme
- Direction
- Step size
- Convergence
- Stochastic Gradient Descent

Optimization

- sometimes, solving $\nabla f(w) = 0$ is difficult directly:
  - nonlinear equations
  - matrix inversion expensive for large matrices ...

Optimization

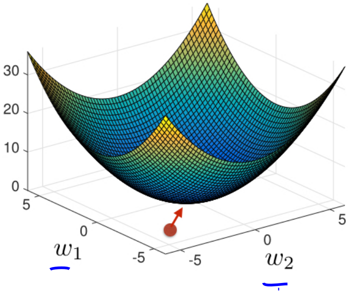
- sometimes, solving $\nabla f(w) = 0$ is difficult directly:
  - nonlinear equations
  - matrix inversion expensive for large matrices ...

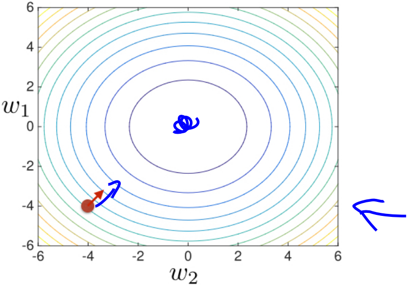
- **Optimization**: essential ingredient of modern data science.
  find
  $$\min_{w \in \mathbb{R}^p} f(w)$$
  Examples: minimizing (regularized) loss/error, maximizing likelihood, ...

Optimization

- sometimes, solving $\nabla f(w) = 0$ is difficult directly:
  - nonlinear equations
  - matrix inversion expensive for large matrices ...
- **Optimization**: essential ingredient of modern data science.
  find
  $$\min_{w \in \mathbb{R}^p} f(w)$$
  Examples: minimizing (regularized) loss/error, maximizing likelihood, ...
- typically iteratively

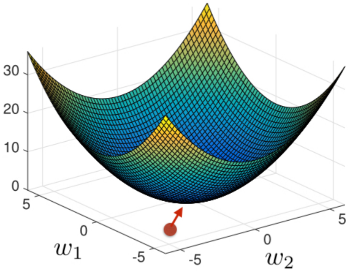
$$\min_{w \in \mathbb{R}^p} f(w)$$

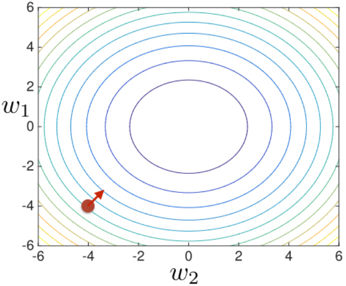




Optimization: Gradient Descent

$$\min_{w \in \mathbb{R}^p} f(w)$$



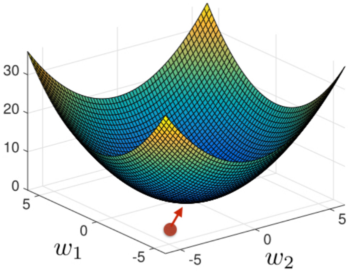


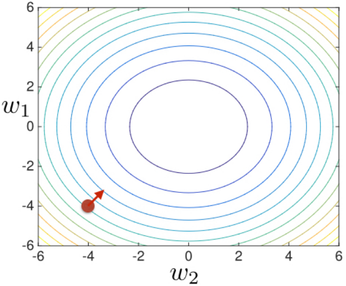
Basic idea: greedy local search

- Start with an arbitrary guess w^0.
- In each iteration, move in direction of “progress” (determined *locally*)

Optimization: Gradient Descent

$$\min_{w \in \mathbb{R}^p} f(w)$$





Basic idea: greedy local search

- Start with an arbitrary guess w^0.
- In each iteration, move in direction of "progress" (determined *locally*)

- if done right, finds point w with $\nabla f(w) \approx 0$.
  convex f: global minimum.
  non-convex f: local minimum, local maximum or saddle point

General scheme

start with some w^0
for $t = 0, 1, 2, \ldots$

$$w^{t+1} \leftarrow \underline{w^t} + \underbrace{\alpha_t}_{\text{step size}} \underbrace{d^t}_{\text{direction}}$$

$$w_i^{t+1} \leftarrow w_i^t + \alpha d_i^t$$

General scheme

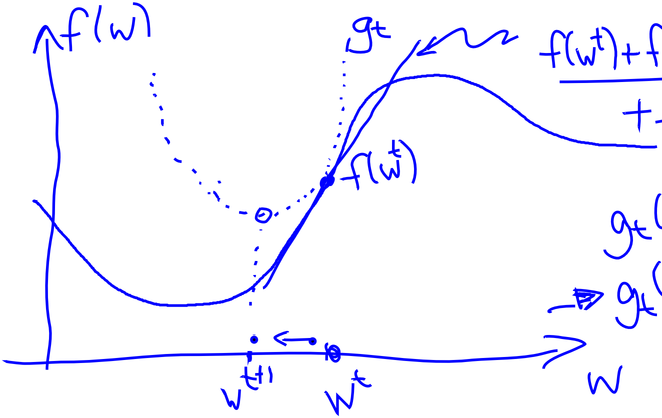
start with some w^0
for $t = 0, 1, 2, \ldots$

$$w^{t+1} \leftarrow w^t + \underbrace{\alpha_t}_{\text{step size}} \underbrace{d^t}_{\text{direction}}$$

Questions:

1. which direction d^t?
2. step size α_t?
3. how many iterations?

Solving $\nabla f(w) = 0$ is difficult in general, but easy for "nice" quadratic functions: Approximate f locally with quadratic function g_t, minimize g_t.



$$f(w^t) + f'(w^t)(u - w_t) + \frac{L}{2}(u - w_t)^2$$

$$g_t(w^t) = f(w^t)$$

$$\Rightarrow g_t(u) \geq f(u) \quad \forall u$$

Deriving GD: Quadratic Upper bounds

- **Want:** (1) $g_t(w) \geq f(w)\ \forall w$ and (2) $g(w^t) = f(w^t)$.

Deriving GD: Quadratic Upper bounds

- **Want:** (1) $g_t(w) \geq f(w)\ \forall w$ and (2) $g(w^t) = f(w^t)$.
- Use *Taylor expansion* to get $g_t(w)$: in 1D

Deriving GD: Quadratic Upper bounds

- **Want:** (1) $g_t(w) \geq f(w)\ \forall w$ and (2) $g(w^t) = f(w^t)$.
- Use *Taylor expansion* to get $g_t(w)$: in 1D

$$f(u) = \underbrace{f(w^t) + \underbrace{f'(w^t)}(u - w^t)} + \underbrace{\frac{1}{2}\underbrace{f''(w^t)}(u - w^t)^2} + \ldots$$

Deriving GD: Quadratic Upper bounds

- **Want:** (1) $g_t(w) \geq f(w)\ \forall w$ and (2) $g(w^t) = f(w^t)$.
- Use *Taylor expansion* to get $g_t(w)$: in 1D

$$f(u) = f(w^t) + f'(w^t)(u - w^t) + \frac{1}{2}f''(w^t)(u - w^t)^2 + \dots$$

$$\geq \quad g_t(u) = f(w^t) + f'(w^t)(u - w^t) + \frac{L}{2}(u - w^t)^2$$

- Choose L large enough to satisfy (1).

Deriving GD: Quadratic Upper bounds

- **Want:** (1) $g_t(w) \geq f(w)\ \forall w$ and (2) $g(w^t) = f(w^t)$.
- Use *Taylor expansion* to get $g_t(w)$: in 1D

$$f(u) = f(w^t) + f'(w^t)(u - w^t) + \frac{1}{2} f''(w^t)(u - w^t)^2 + \ldots$$

$$g_t(u) = f(w^t) + f'(w^t)(u - w^t) + \frac{L}{2}(u - w^t)^2$$

- Choose L large enough to satisfy (1).
- Setting $g_t'(u) = 0$ gives minimizer of g:

$$u_t = w^t - \frac{1}{L} f'(w^t) = w^{t+1}$$

$$u_t = w^t - \frac{1}{L} \nabla_w f(w^t) = w^{t+1}$$

Deriving GD: Quadratic Upper bounds

- **Want:** (1) $g_t(w) \geq f(w)\ \forall w$ and (2) $g(w^t) = f(w^t)$.
- Use *Taylor expansion* to get $g_t(w)$: in 1D

$$f(u) = f(w^t) + f'(w^t)(u - w^t) + \frac{1}{2} f''(w^t)(u - w^t)^2 + \dots$$

$$g_t(u) = f(w^t) + f'(w^t)(u - w^t) + \frac{L}{2}(u - w^t)^2$$

- Choose L large enough to satisfy (1).
- Setting $g_t'(u) = 0$ gives minimizer of g:

$$u_t = w^t - \frac{1}{L} f'(w^t)$$

Set $w^{t+1} = u_t = w^t - \frac{1}{L} f'(w^t)$, i.e., $d_t = -f'(w^t)$, $\alpha = 1/L$.

Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.

Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.
- Gradient for squared loss:

$$\nabla_w \Big(\sum_{i=1}^{N} (y_i - x_i \cdot w)^2 \Big) = \sum_{i=1}^{N} \nabla_w (y_i - x_i \cdot w)^2$$

$y_i - \hat{y}_i$

Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.
- Gradient for squared loss:

$$\nabla_w \Big(\sum_{i=1}^{N} (y_i - x_i \cdot w)^2 \Big) = \sum_{i=1}^{N} \nabla_w (y_i - x_i \cdot w)^2$$

$$= -2 \sum_{i=1}^{N} \underbrace{(y_i - x_i \cdot w)}_{y_i - \hat{y}_i} \underline{x_i^\top}$$

Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.
- Gradient for squared loss:

$$\nabla_w \Big(\sum_{i=1}^{N} (y_i - x_i \cdot w)^2 \Big) = \sum_{i=1}^{N} \nabla_w (y_i - x_i \cdot w)^2$$

$$= -2 \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$$

so $w^{t+1} = w^t + 2\alpha_t \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$

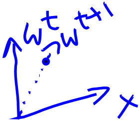
Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.
- Gradient for squared loss:

$$\nabla_w \Big(\sum_{i=1}^{N} (y_i - x_i \cdot w)^2 \Big) = \sum_{i=1}^{N} \nabla_w (y_i - x_i \cdot w)^2$$

$$= -2 \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$$

so $w^{t+1} = w^t + 2\alpha_t \sum_{i=1}^{N} (y_i - \overset{y_i}{x_i \cdot w}) x_i^\top$



- w^{t+1} will be a combination of x_i's

Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.
- Gradient for squared loss:

$$\nabla_w \Big(\sum_{i=1}^{N} (y_i - x_i \cdot w)^2 \Big) = \sum_{i=1}^{N} \nabla_w (y_i - x_i \cdot w)^2$$
$$= -2 \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$$

so $w^{t+1} = w^t + 2\alpha_t \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$

- w^{t+1} will be a combination of x_i's
- positive residual ($y_i > \hat{y}_i$):
  add fraction of x_i to w^t, increases dot product $x_i \cdot w$

Example: least squares regression

- We obtain iteration: $w^{t+1} \leftarrow w^t - \alpha_t \nabla f(w^t)$.
- Gradient for squared loss:

$$\nabla_w \Big(\sum_{i=1}^{N} (y_i - x_i \cdot w)^2 \Big) = \sum_{i=1}^{N} \nabla_w (y_i - x_i \cdot w)^2$$
$$= -2 \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$$

so $w^{t+1} = w^t + 2\alpha_t \sum_{i=1}^{N} (y_i - x_i \cdot w) x_i^\top$

- w^{t+1} will be a combination of x_i's
- positive residual ($y_i > \hat{y}_i$):
  add fraction of x_i to w^t, increases dot product $x_i \cdot w$
- negative residual: subtract fraction of x_i, decreases dot product

General scheme

start with some w^0
for $t = 0, 1, 2, \ldots$

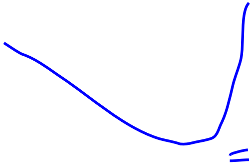
$$w^{t+1} \leftarrow w^t + \underbrace{\alpha_t}_{\text{step size}} \underbrace{d^t}_{\text{direction}}$$

Questions:

1. which direction d^t?
2. step size α_t?
3. how many iterations?

More on step sizes

- Step size determined by quadratic upper bound / increase in slope
- If increase is different in different directions: use smallest step size. slower progress overall.



More on step sizes

- Step size determined by quadratic upper bound / increase in slope
- If increase is different in different directions: use smallest step size. slower progress overall.

- If we don't know L: Tuning...

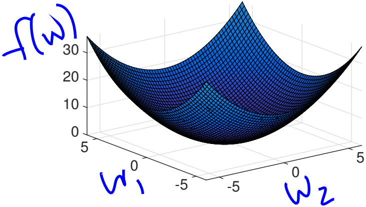
More on step sizes

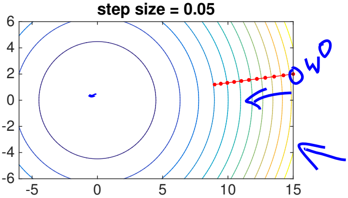
- Step size determined by quadratic upper bound / increase in slope
- If increase is different in different directions: use smallest step size. slower progress overall.

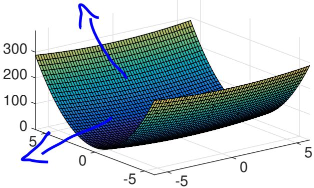
- If we don't know L: Tuning...

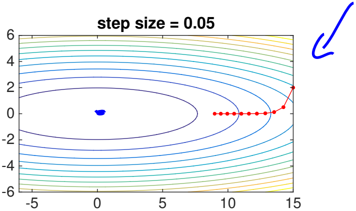
  too small: slow progess.
  too large: erratic or no convergence.

Step sizes & progress



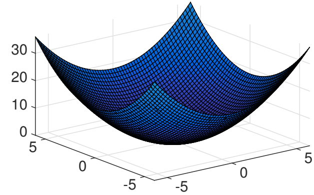


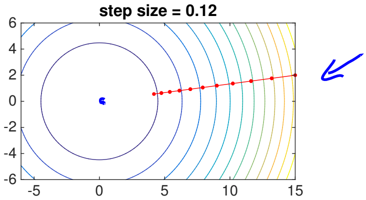


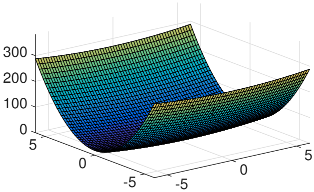


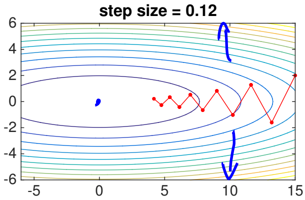
more demo: `http://fa.bianp.net/teaching/2018/eecs227at/gradient_descent.html`

Step sizes & progress

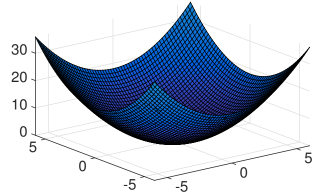


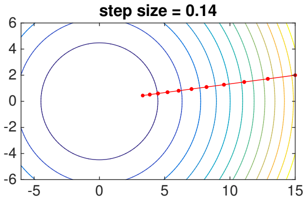


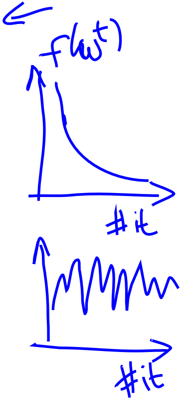


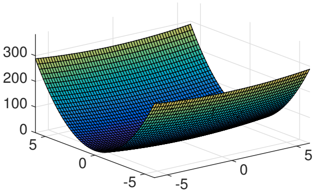


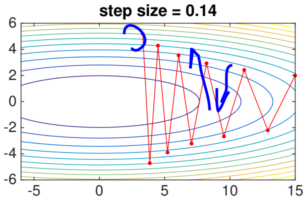
Step sizes & progress



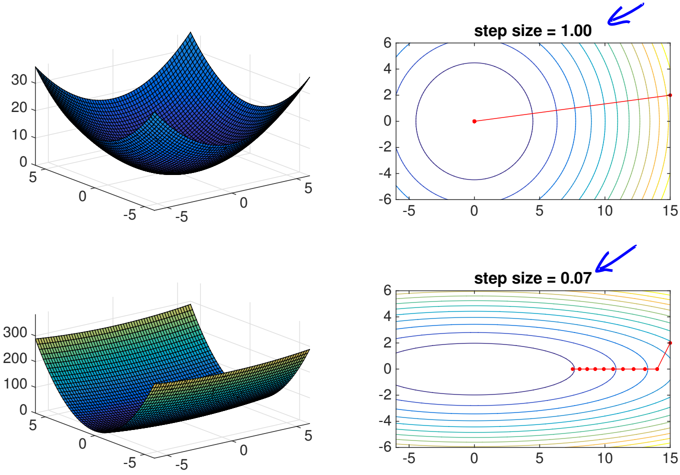








Step sizes: using $\alpha = 1/L$



More on step sizes

- Step size determined by quadratic upper bound / increase in slope
- If increase is different in different directions: use smallest step size. slower progress overall.

- If we don't know L: Tuning, or **Backtracking**:

More on step sizes

- Step size determined by quadratic upper bound / increase in slope
- If increase is different in different directions: use smallest step size. slower progress overall.

- If we don't know L: Tuning, or **Backtracking**:

$$\boxed{\text{With step size } \alpha_t = 1/L: \quad f(w^{t+1}) \le \underbrace{f(w^t) - \frac{1}{2L}\|f(w^t)\|^2}_{g_t(w^{t+1})}.}$$

More on step sizes

- Step size determined by quadratic upper bound / increase in slope
- If increase is different in different directions: use smallest step size. slower progress overall.

- If we don't know L: Tuning, or **Backtracking**:

With step size $\alpha_t = 1/L$: $\quad f(w^{t+1}) \le f(w^t) - \frac{1}{2L}\|f(w^t)\|^2$.

in each step t: select optimistic α_t, check if

$\|a\|^2 = \sum_{j=1}^{n} a_j^2$

$$f(w_t - \alpha_t \nabla f(w_t)) \le f(w_t) - \frac{\alpha_t}{2}\|\nabla f(w_t)\|^2$$

If yes: use α_t. If no: $\alpha_t = \alpha_t/2$ and check again.

How many iterations?

- **practice:** e.g. until $\underbrace{\|\nabla f(w^t)\|}$ is "small enough", or only small change in loss

$f(w^t) - f(w^{t+1})$

[1] means f is also lower bounded by a quadratic, with constant m (instead of L). See e.g. Boyd & Vandenberghe book.

How many iterations?

- **practice:** e.g. until $\|\nabla f(w)\|$ is "small enough", or only small change in loss

- **theory:** $f(w^t) - f(w^*) \leq \ldots$ (gap to optimum)

[1] means f is also lower bounded by a quadratic, with constant m (instead of L). See e.g. Boyd & Vandenberghe book.

How many iterations?

- **practice:** e.g. until $\|\nabla f(w)\|$ is "small enough", or only small change in loss

- **theory:** $f(w^t) - f(w^*) \leq \ldots$ (gap to optimum)
- *convex functions with bounded L (gradients)*:

$$f(w^t) - f(w^*) \leq \frac{L\|w^0 - w^*\|^2}{2t}$$

(need $O(1/\epsilon)$ iterations for $f(w^t) - f(w^*) \leq \epsilon$)

[1] means f is also lower bounded by a quadratic, with constant m (instead of L). See e.g. Boyd & Vandenberghe book.

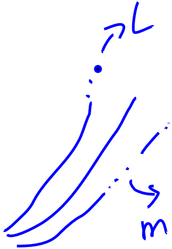
How many iterations?

- **practice:** e.g. until $\|\nabla f(w)\|$ is “small enough”, or only small change in loss

- **theory:** $f(w^t) - f(w^*) \leq \ldots$ (gap to optimum)
- *convex functions with bounded L (gradients)*:

$$f(w^t) - f(w^*) \leq \frac{L\|w^0 - w^*\|^2}{2t}$$

(need $O(1/\epsilon)$ iterations for $f(w^t) - f(w^*) \leq \epsilon$)



- *m-strongly convex functions*[1]:

$$f(w^t) - f(w^*) \leq \underbrace{(1 - \tfrac{m}{L})}\,^{t}(f(w^0) - f(w^*))$$

(need $O(\log(1/\epsilon))$ iterations for $f(w^t) - f(w^*) \leq \epsilon$)

[1] means f is also lower bounded by a quadratic, with constant m (instead of L). See e.g. Boyd & Vandenberghe book.

Outline

- Convex functions
- Gradient descent: main scheme
- Direction
- Step size
- Convergence
- Stochastic Gradient Descent

What if N is large?

Stochastic Gradient descent

- Setup: f is a sum: $f(w) = \sum_{i=1}^{N} f_i(w)$
- Gradient: $\nabla_w f(w) = \sum_{i=1}^{N} \nabla_w f_i(w)$

What if N is large?

Stochastic Gradient descent

- Setup: f is a sum: $f(w) = \sum_{i=1}^{N} f_i(w)$
- Gradient: $\nabla_w f(w) = \sum_{i=1}^{N} \nabla_w f_i(w)$
- Idea: estimate sum via few (one) term(s)!

What if N is large?

Stochastic Gradient descent

- Setup: f is a sum: $f(w) = \sum_{i=1}^{N} f_i(w)$
- Gradient: $\nabla_w f(w) = \sum_{i=1}^{N} \nabla_w f_i(w)$
- Idea: estimate sum via few (one) term(s)!

start with some w^0
for $t = 0, 1, 2, \ldots$
draw (data point) i uniformly at random, $1 \leq i \leq N$

$$w^{t+1} \leftarrow w^t - \alpha_t \nabla f_i(w^t)$$

What if N is large?

Stochastic Gradient descent

- Setup: f is a sum: $f(w) = \sum_{i=1}^{N} f_i(w)$
- Gradient: $\nabla_w f(w) = \sum_{i=1}^{N} \nabla_w f_i(w)$
- Idea: estimate sum via few (one) term(s)!

> start with some w^0
> for $t = 0, 1, 2, \ldots$
> draw (data point) i uniformly at random, $1 \leq i \leq N$
>
> $$w^{t+1} \leftarrow w^t - \alpha_t \nabla f_i(w^t)$$

- with "right" step size, also converges to minimum. Step size must shrink ($\approx \frac{1}{t+1}$)

What if N is large?

Stochastic Gradient descent

- Setup: f is a sum: $f(w) = \sum_{i=1}^{N} f_i(w)$
- Gradient: $\nabla_w f(w) = \sum_{i=1}^{N} \nabla_w f_i(w)$
- Idea: estimate sum via few (one) term(s)!

start with some w^0
for $t = 0, 1, 2, \ldots$
draw (data point) i uniformly at random, $1 \leq i \leq N$

$$w^{t+1} \leftarrow w^t - \alpha_t \nabla f_i(w^t)$$

- with "right" step size, also converges to minimum. Step size must shrink ($\approx \frac{1}{t+1}$)
- more erratic, but standard for large data

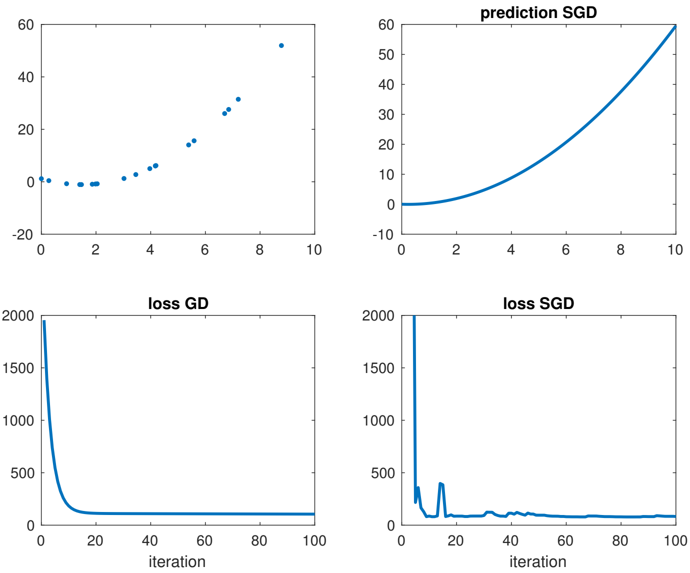
Demo

- Fit a function to observations from $y = \beta_1 x + \beta_2 x^2$ (upper left: observations) via least squares regression with gradient descent and stochastic gradient descent.

Demo

- Fit a function to observations from $y = \beta_1 x + \beta_2 x^2$ (upper left: observations) via least squares regression with gradient descent and stochastic gradient descent.
- Bottom row: Loss function (sum of squared residuals) values for gradient descent (GD, left) and stochastic gradient descent (SGD,right). Note that SGD only uses *one* data point per iteration, whereas GD uses all data points in each iteration.

Demo: plots



60
40
20
0
-20
0
2
4
6
8
10
prediction SGD
60
50
40
30
20
10
0
-10
0
2
4
6
8
10
loss GD
2000
1500
1000
500
0
0
20
40
60
80
100
iteration
loss SGD
2000
1500
1000
500
0
0
20
40
60
80
100
iteration

Summary: Gradient descent

- (Stochastic) Gradient Descent is a pillar of modern machine learning
- iterative method to find a point with zero gradient
- small steps in direction of negative gradient
- for convex functions, finds the global minimum

References

Convex Optimization

- S. Boyd & L. Vandenberghe. *Convex Optimization*. Available online: `http://stanford.edu/~boyd/cvxbook/`
  Parts of Chapters 3.1, (3.2 for additional optional reading); parts of Chapter 9.1 and 9.3

For some background in linear algebra

- very short: Appendix in Boyd & Vandenberghe. Or: Many statistics books have a chapter on it, e.g., D. Freedman. *Statistical Models – Theory and Practice*.
- a bit longer:
  T.A. Garrity. *All the Mathematics you missed: But need to know for Graduate School*. Cambridge University Press.

Appendix: some linear algebra concepts

- **rank** of matrix: max. number of linearly independent columns (rows)

- $\mathbf{X}^\top \mathbf{X}$ is invertible if it has full rank (no zero eigenvalues), i.e., if $N \geq p$ and all columns of $\mathbf{X}$ are linearly independent.

Appendix: some linear algebra concepts

- **rank** of matrix: max. number of linearly independent columns (rows)
- **eigenvalue**: an eigenvector $\mathbf{u}$ of a matrix $\mathbf{A}$ and the corresponding eigenvalue λ satisfy $\mathbf{Au} = \lambda\mathbf{u}$.

$$\lambda_{\max} = \max_{\mathbf{u}\in\mathbb{R}^p, \|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{Au} \qquad \lambda_{\min} = \min_{\mathbf{u}\in\mathbb{R}^p, \|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{Au}$$

- $\mathbf{X}^\top\mathbf{X}$ is invertible if it has full rank (no zero eigenvalues), i.e., if $N \geq p$ and all columns of $\mathbf{X}$ are linearly independent.

Appendix: some linear algebra concepts

- **rank** of matrix: max. number of linearly independent columns (rows)
- **eigenvalue**: an eigenvector $\mathbf{u}$ of a matrix $\mathbf{A}$ and the corresponding eigenvalue λ satisfy $\mathbf{Au} = \lambda\mathbf{u}$.

$$\lambda_{\max} = \max_{\mathbf{u} \in \mathbb{R}^p, \|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{A} \mathbf{u} \qquad \lambda_{\min} = \min_{\mathbf{u} \in \mathbb{R}^p, \|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{A} \mathbf{u}$$

- symmetric $\mathbf{A}$ is *positive semidefinite* (psd) if

$$\mathbf{v}^\top \mathbf{A} \mathbf{v} \geq 0 \text{ for all } \mathbf{v} \in \mathbb{R}^p$$

  Equivalent: $\mathbf{A}$ is symmetric and all eigenvalues are nonnegative.

- $\mathbf{X}^\top \mathbf{X}$ is invertible if it has full rank (no zero eigenvalues), i.e., if $N \geq p$ and all columns of $\mathbf{X}$ are linearly independent.

Appendix: Additional note: What is L?

- We want (scalar version):

$$f(u) \leq f(w) + f'(w)(u-w) + \frac{L}{2}(u-w)^2$$

Taylor expansion:

$$f(u) = f(w) + f'(w)(u-w) + \tfrac{1}{2}f''(z)(u-w)^2$$

so $L = \max_z f''(z)$.

Appendix: Additional note: What is L?

- We want (scalar version):

$$f(u) \leq f(w) + f'(w)(u-w) + \frac{L}{2}(u-w)^2$$

Taylor expansion:

$$f(u) = f(w) + f'(w)(u-w) + \tfrac{1}{2}f''(z)(u-w)^2$$

so $L = \max_z f''(z)$.

- for vectors, this condition becomes:

$$v^\top[\nabla^2 f(z)]v \;\leq\; Lv^\top v \qquad \text{for } v = (u-w)$$

i.e., L must be larger than largest eigenvalue of $\nabla^2 f(z)$ for all z.

Appendix: Additional note: What is L?

- We want (scalar version):

$$f(u) \leq f(w) + f'(w)(u-w) + \frac{L}{2}(u-w)^2$$

  Taylor expansion:

$$f(u) = f(w) + f'(w)(u-w) + \tfrac{1}{2}f''(z)(u-w)^2$$

  so $L = \max_z f''(z)$.

- for vectors, this condition becomes:

$$v^\top [\nabla^2 f(z)] v \;\leq\; L v^\top v \qquad \text{for } v = (u-w)$$

  i.e., L must be larger than largest eigenvalue of $\nabla^2 f(z)$ for all z.

- if $f(w) = w^\top \mathbf{A} w + bw$, then $\nabla^2 f(z) = \mathbf{A}$ and
  L is maximum eigenvalue of $\mathbf{A}$: $L = \lambda_{\max}$