Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Gradient descent
- Stochastic gradient descent (SGD)
- Convexity

With credit to S. Dasgupta, T. Jaakkola, and C. Tan

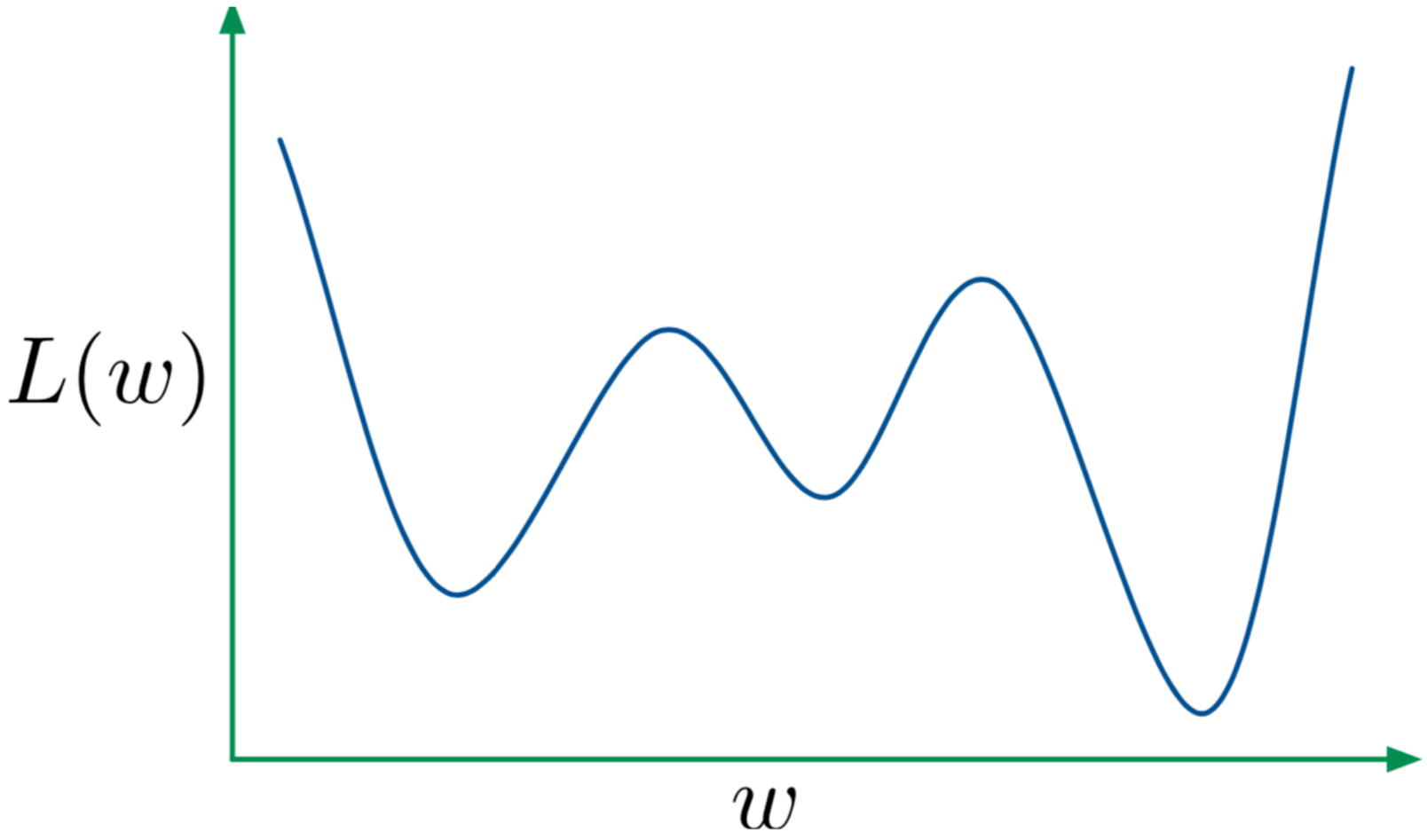
Minimizing a loss function

Usual setup in machine learning: choose a model w by minimizing a loss function $L(w)$ that depends on the data.

- Linear regression: $L(w) = \sum_i (y^{(i)} - (w \cdot x^{(i)}))^2$
- Logistic regression: $L(w) = \sum_i \ln(1 + e^{-y^{(i)}(w \cdot x^{(i)})})$

Default way to solve this minimization: **local search**.

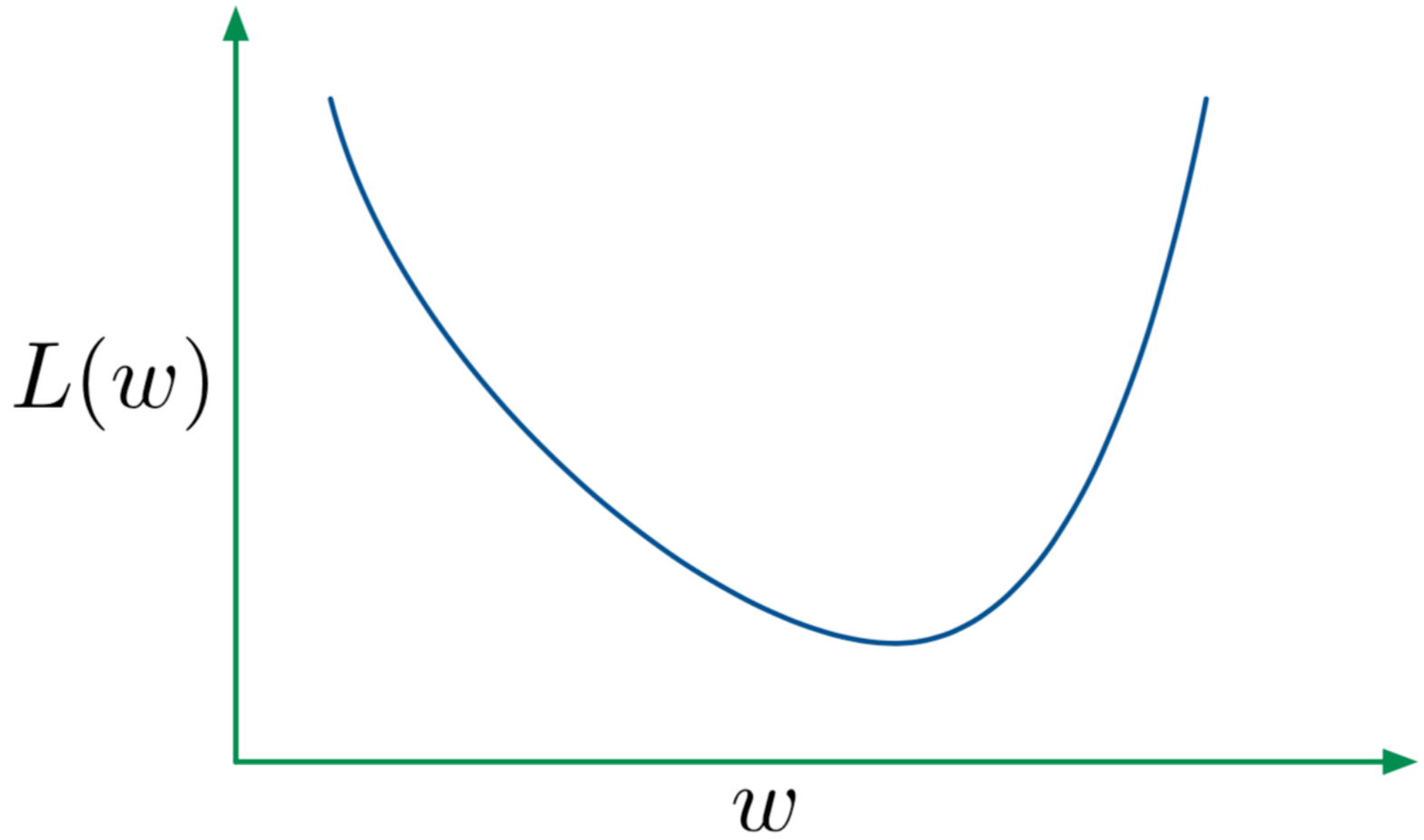
Local search



- Initialize w arbitrarily
- Repeat until w converges:
  - Find some w' close to w with $L(w') < L(w)$.
  - Move w to w'.

Local search works well when:

When the loss function is **convex**:



Idea: pick search direction by looking at **derivative** of $L(w)$.

Gradient descent

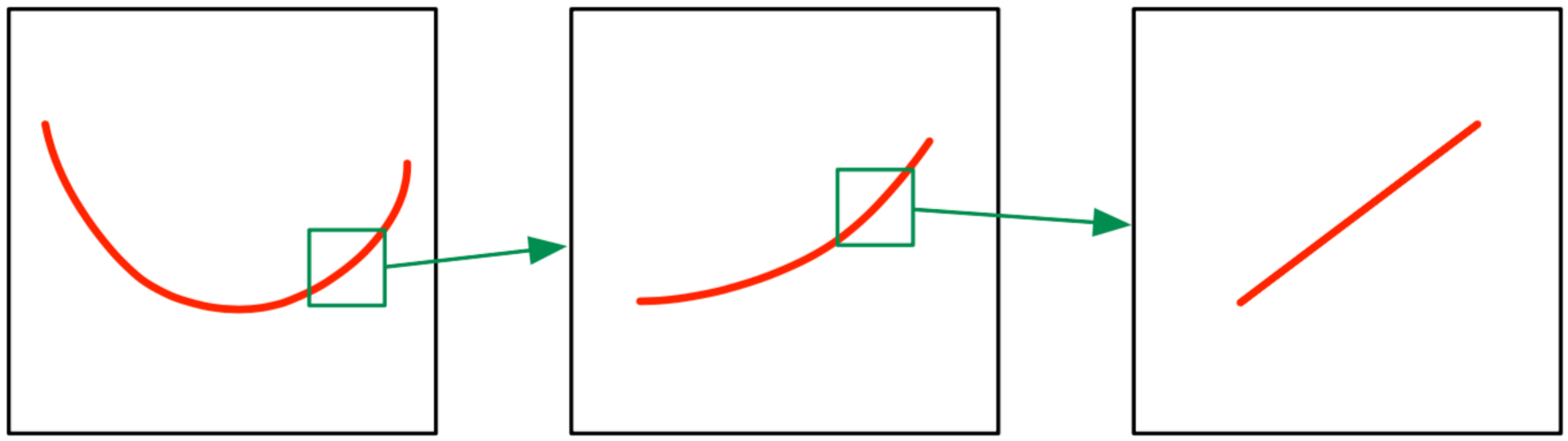
For minimizing a function $L(w)$:

- $w_o = 0, t = 0$
- while $\nabla L(w_t) \not\approx 0$:
  - $w_{t+1} = w_t - \eta_t \nabla L(w_t)$
  - $t = t + 1$

Here η_t is the *step size* at time t.

Motivation for gradient descent

"Differentiable" $\Longrightarrow$ "locally linear".



For *small* displacements $u \in \mathbb{R}^d$,

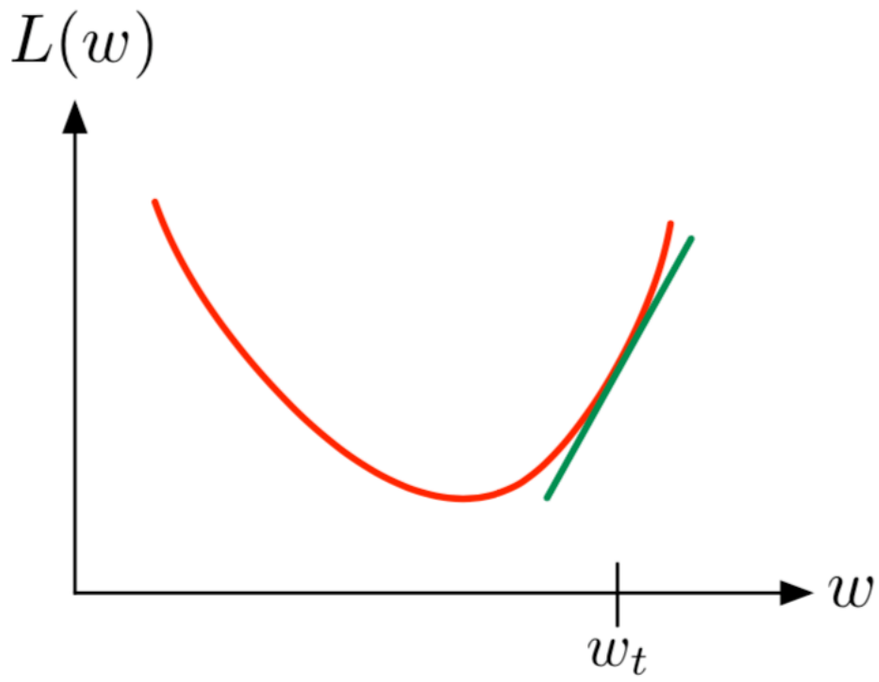
$$L(w+u) \approx L(w) + u \cdot \nabla L(w) \quad .$$

Therefore, if $u = -\eta \nabla L(w)$ is small,

$$L(w+u) \approx L(w) - \eta \|\nabla L(w)\|^2 < L(w)$$

Choice of step-size is important

Gradient descent update: $w_{t+1} = w_t - \eta_t \nabla L(w_t)$.



- Step size η_t too small: not much progress
- Too large: overshoot the mark

One option: pick η_t using a line search

$$\eta_t = \arg\min_{\alpha>0} L(w_t - \alpha \nabla L(w_t))$$

Recall: Logistic Regression

For $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$, loss function

$$L(w) = \sum_{i=1}^{n} \ln(1 + e^{-y^{(i)}(w \cdot x^{(i)})})$$

Compute derivative of L(w), to define gradient descent:

- Set $w_0 = 0$
- For $t = 0, 1, 2, \ldots,$ until convergence:

$$w_{t+1} = w_t + \eta_t \sum_{i=1}^{n} y^{(i)} x^{(i)} \Pr_{w_t}(-y^{(i)} | x^{(i)})$$

Gradient descent for Logistic Regression

- Set $w_0 = 0$
- For $t = 0, 1, 2, \ldots,$ until convergence:

$$w_{t+1} \;=\; w_t \;+\; \eta_t \sum_{i=1}^{n} y^{(i)} x^{(i)} \Pr_{w_t}(-y^{(i)} | x^{(i)})$$

Do we like this algorithm?

No, too much computation!

Each update is a function of the whole training data set!

Stochastic gradient descent

Stochastic gradient descent: update based on just one point:

- Get next data point (x, y) by cycling through data set
- $w_{t+1} = w_t + \eta_t \, y \, x \, \Pr_{w_t}(-y|x)$

Decomposable loss functions

Loss function for logistic regression:

$$L(w) = \sum_{i=1}^{n} \ln(1 + e^{-y^{(i)}(w \cdot x^{(i)})}) = \sum_{i=1}^{n} (\text{loss of } w \text{ on } (x^{(i)}, y^{(i)}))$$

Most ML loss functions are like this:
Given $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$,

$$L(w) = \sum_{i=1}^{n} \ell(w; x^{(i)}, y^{(i)})$$

where $\ell(w; x, y)$ captures the loss on a single point.

For minimizing

$$L(w) = \sum_{i=1}^{n} \ell(w; x^{(i)}, y^{(i)})$$

Gradient descent:

- $w_o = 0$
- while not converged:
  - $w_{t+1} = w_t - \eta_t \sum_{i=1}^{n} \nabla \ell(w_t; x^{(i)}, y^{(i)})$

Stochastic gradient descent:

- $w_o = 0$
- Keep cycling through data points (x, y):
  - $w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x, y)$

Mini-batch variant of SGD

Stochastic gradient descent:

- $w_o = 0$
- Keep cycling through data points (x, y):
  - $w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x, y)$

Mini-batch stochastic gradient descent:

- $w_o = 0$
- Repeat:
  - Get the next batch of points B
  - $w_{t+1} = w_t - \eta_t \sum_{(x,y) \in B} \nabla \ell(w_t; x, y)$

Regularized objectives

Compute the gradient on the regularized objective function

Logistic Regression

- Objective function (SGD)

$$\mathcal{L} = -\log p(y|x;\beta)$$

- Derivative of objective

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -(y_i - \pi_i)x_j$$

- SGD update rule

$$\beta_j \leftarrow \beta'_j + \eta\left(x_{ij}\left[y_i - \pi_i\right]\right)$$

Regularized Logistic Regression

- Objective function (SGD)

$$\mathcal{L} = -\log p(y|x;\beta) + \frac{1}{2}\mu\sum_j \beta_j^2$$

- Derivative of objective

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -(y_i - \pi_i)x_j + \mu\beta_j$$

- SGD update rule

$$\beta_j \leftarrow \beta'_j - \eta\left(\mu\beta'_j - x_{ij}\left[y_i - \pi_i\right]\right)$$

SGD for Regularized Logistic Regression

Initialize a vector β to be all zeros

For $t = 1, \ldots, T$

- For each example $\vec{x}_i, y_i$ and feature j:
  - Compute $\pi_i \equiv \Pr(y_i = 1 \mid \vec{x}_i)$
  - Set $\beta_j = \beta'_j - \eta(\mu\beta'_j - (y_i - \pi_i)x_i)$

Output the parameters $\beta_1, \ldots, \beta_d$.

Do we like this algorithm?

Yes, but "wasted" updates when x_{ij}=0.

→ There are more efficient appraoches for sparse data (e.g. lazy sparse SGD)

SGD: theoretical results

- Significant literature analyzing SGD
  - With or without mini-batches
  - Varying assumptions on convexity, and smoothness of the loss function
- Some results:

Number of iterations to get to accuracy:

$$\mathscr{L}(\beta) - \mathscr{L}(\beta^*) \leq \epsilon$$

Gradient descent:

- If $\mathscr{L}$ is strongly convex: $O(\log(1/\epsilon))$ iterations

Stochastic gradient descent:

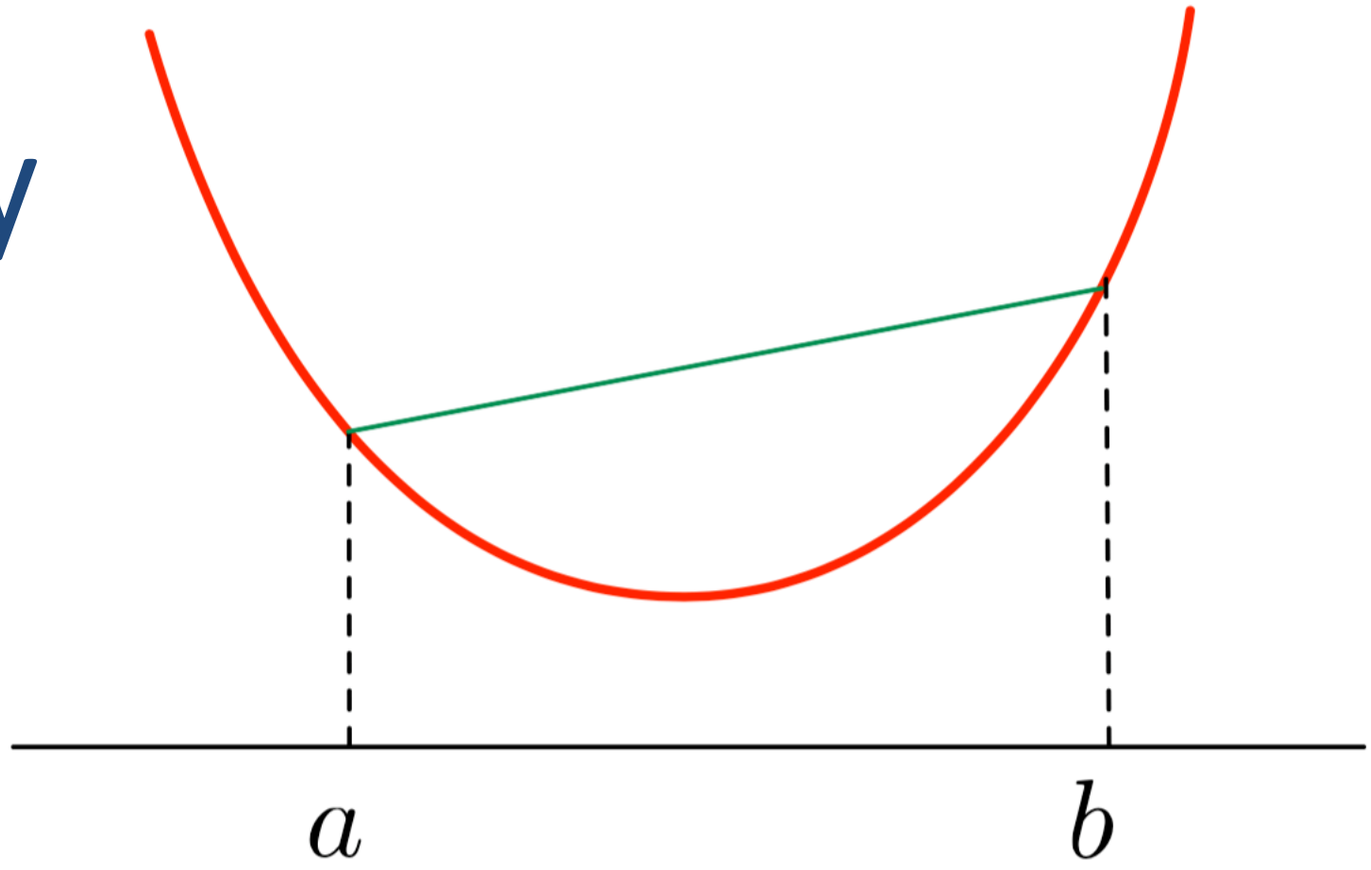
- If $\mathscr{L}$ is strongly convex: $O(1/\epsilon)$ iterations

Note: Best analyses involve algorithms in which the step-size is allowed to depend not only on time but also on dimension….

Facts about convexity

- Sum of convex functions is convex
- Composition with a linear function maintains convexity

Convexity



A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if for all $a, b \in \mathbb{R}^d$ and $0 < \theta < 1$,

$$f(\theta a + (1-\theta)b) \;\leq\; \theta f(a) + (1-\theta) f(b).$$

It is **strictly convex** if strict inequality holds for all $a \neq b$.

f is **concave** $\Leftrightarrow$ $-f$ is convex

Convexity

Function of one variable

$$F : \mathbb{R} \to \mathbb{R}$$

- Value: number
- Derivative: number
- Second derivative: number

Convex if second derivative is always ≥ 0

Function of d variables

$$F : \mathbb{R}^d \to \mathbb{R}$$

- Value: number
- Derivative: d-dimensional vector
- Second derivative: $d \times d$ matrix

Convex if second derivative matrix is always positive semidefinite

Derivatives of functions of multiple variables

For a function $f : \mathbb{R}^d \to \mathbb{R}$,

- the first derivative is a vector with d entries:

$$\nabla f(z) = \begin{pmatrix} \frac{\partial f}{\partial z_1} \\ \vdots \\ \frac{\partial f}{\partial z_d} \end{pmatrix}$$

- the second derivative is a $d \times d$ matrix, the **Hessian** $H(z)$:

$$H_{jk} = \frac{\partial^2 f}{\partial z_j \partial z_k}$$

Testing for convexity

A function of several variables, $F(z)$, is convex if its second-derivative matrix $H(z)$ is positive semidefinite for all z.

More formally:

Suppose that for $f : \mathbb{R}^d \to \mathbb{R}$, the second partial derivatives exist everywhere and are continuous functions of z. Then:

1. $H(z)$ is a symmetric matrix
2. f is convex $\Leftrightarrow$ $H(z)$ is positive semidefinite for all $z \in \mathbb{R}^d$

How to check if a matrix is PSD

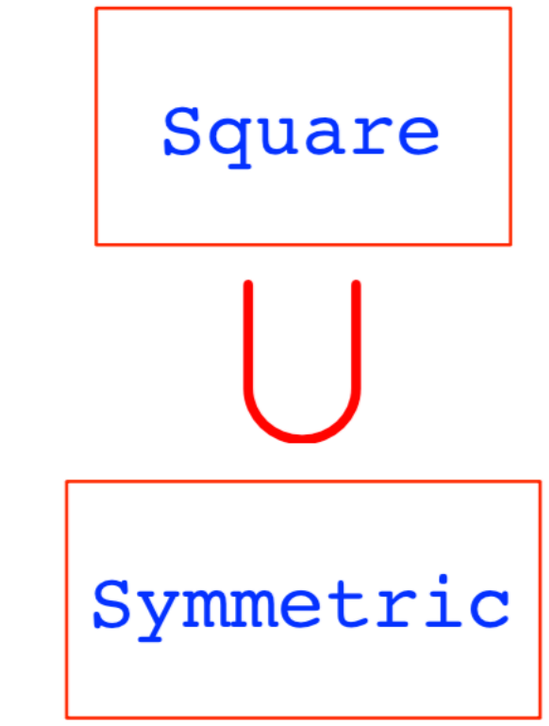
A matrix M is PSD if and only if it can be written as $M = UU^T$ for some matrix U.

Quick check: say $U \in \mathbb{R}^{r \times d}$ and $M = UU^T$.

1. M is square.
2. M is symmetric.
3. Pick any $x \in \mathbb{R}^r$. Then

$$\begin{aligned} x^T M x &= x^T U U^T x &&= (x^T U)(U^T x) \\ &&&= (U^T x)^T (U^T x) \\ &&&= \|U^T x\|^2 \geq 0. \end{aligned}$$

Another useful fact: any covariance matrix is PSD.



$$M \in \mathbb{R}^{d \times d}$$

$$M = M^T$$

$$x^T M x \geq 0 \text{ for all } x \in \mathbb{R}^d$$