Unconstrained optimization

DSE 220

Machine learning is all about optimization.

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

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.

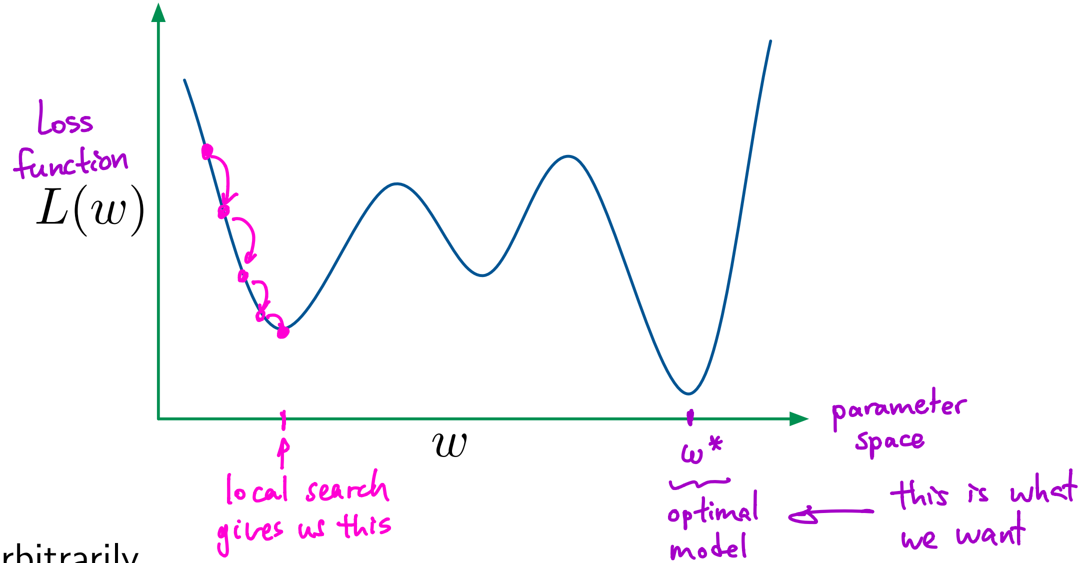
- Linear regression: $L(w) = \sum_i (y^{(i)} - (w \cdot x^{(i)}))^2$
- Logistic regression: $L(w) = \sum_i \ln(1 + e^{-y^{(i)}(w \cdot x^{(i)})})$

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

There are many local search methods.

Today: <u>gradient descent.</u>

Local search

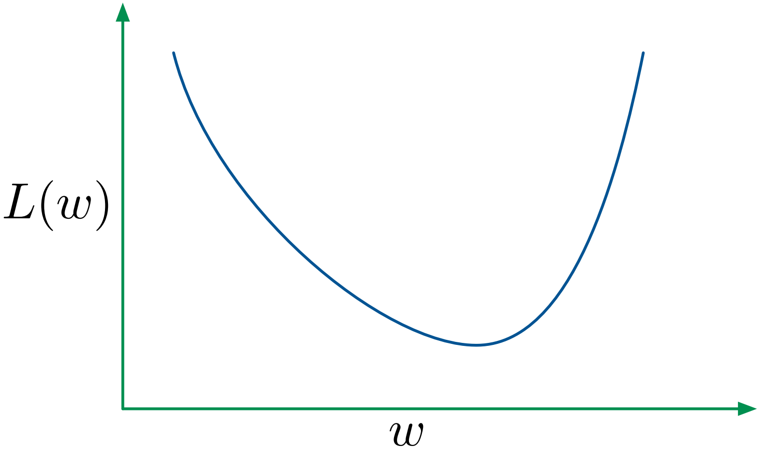


Local search

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

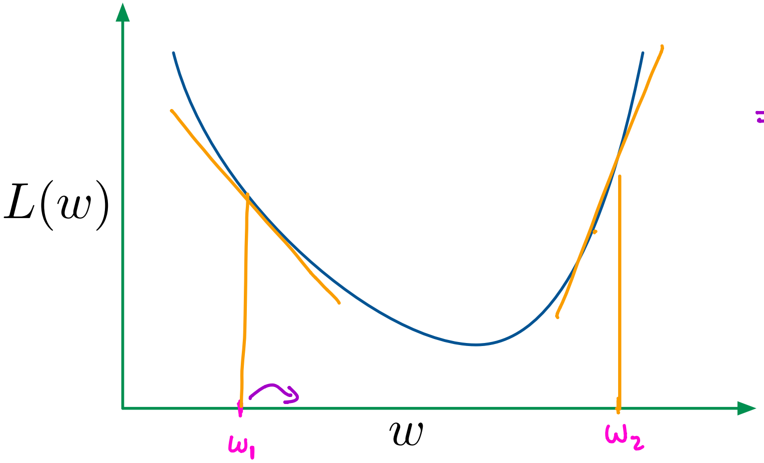
A good situation for local search

When the loss function is **convex**:



A good situation for local search

When the loss function is **convex**: bowl-shaped



Slope at w_1 is negative, ie.

$\frac{dL}{dw} < 0$ at w_1

(ie. $L'(w_1) < 0$)

$\Rightarrow$ increasing w_1 will reduce the loss.

$\therefore$ Increase w_1

Slope at w_2 is positive, ie. $L'(w_2) > 0$

$\Rightarrow$ decreasing w_2 will decrease the loss

$\therefore$ Decrease w_2

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

Typically: w is d-DIMENSIONAL... what happens then? What is the gradient?

Multivariate differentiation

$w = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$

Example: $w \in \mathbb{R}^3$ and $F(w) = 3w_1w_2 + w_3$.

$$F(w_1, w_2, w_3) = 3w_1w_2 + w_3$$

Take derivative with respect to each of w_1, w_2, w_3:

$$\left.\begin{aligned} \frac{dF}{dw_1} &= 3w_2 \\ \frac{dF}{dw_2} &= 3w_1 \\ \frac{dF}{dw_3} &= 1 \end{aligned}\right\} \xRightarrow[\text{vector}]{\text{package into a}}$$

gradient

$$\nabla F(w) = \begin{pmatrix} \frac{dF}{dw_1} \\ \frac{dF}{dw_2} \\ \frac{dF}{dw_3} \end{pmatrix} = \begin{pmatrix} 3w_2 \\ 3w_1 \\ 1 \end{pmatrix}$$

E.g. What is the gradient at $w = (2, 3, 6)$?

$$\begin{pmatrix} 9 \\ 6 \\ 1 \end{pmatrix}$$

Example: $w \in \mathbb{R}^d$ and $F(w) = w \cdot x$.

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix}$$

$$F(w_1, w_2, \ldots, w_d) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

$$\left.\begin{aligned} \frac{dF}{dw_1} &= x_1 \\ \frac{dF}{dw_2} &= x_2 \\ &\vdots \\ \frac{dF}{dw_d} &= x_d \end{aligned}\right\} \quad \underbrace{\nabla F(w)}_{\text{gradient}} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = x$$

Example: $w \in \mathbb{R}^d$ and $F(w) = \|w\|^2$.

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix}$$

$$F(w) = w_1^2 + w_2^2 + w_3^2 + \cdots + w_d^2$$

$$\left.\begin{aligned} \frac{dF}{dw_1} &= 2w_1 \\ \frac{dF}{dw_2} &= 2w_2 \\ &\vdots \\ \frac{dF}{dw_d} &= 2w_d \end{aligned}\right\} \quad \nabla F(w) = \begin{pmatrix} 2w_1 \\ 2w_2 \\ \vdots \\ 2w_d \end{pmatrix} = 2w$$

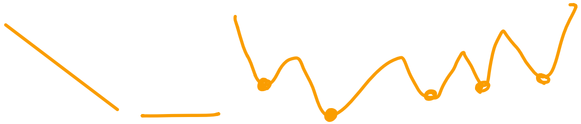
What is the gradient at $w = (1, 6, -2)$?

$$\begin{pmatrix} 2 \\ 12 \\ -4 \end{pmatrix}$$

Example: $w \in \mathbb{R}^d$ and $F(w) = w^T M w$.

Gradient descent

Say $w \in \mathbb{R}^d$



For minimizing a function $L(w)$:

initial parameter in $\mathbb{R}^d$

gradient, in $\mathbb{R}^d$

- $w_o = 0, t = 0$
- 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.

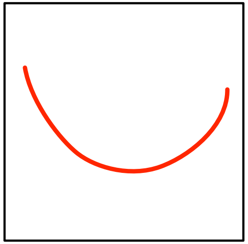
updated parameter

step size (scalar)

current parameter

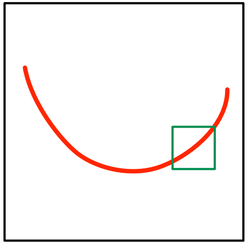
Gradient descent: rationale

"Differentiable" $\implies$ "locally linear".



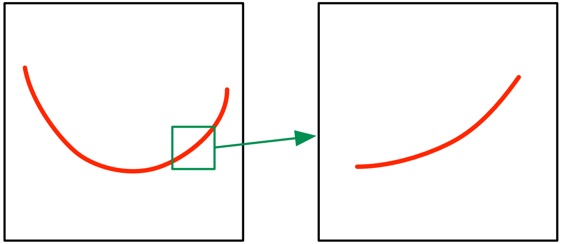
Gradient descent: rationale

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



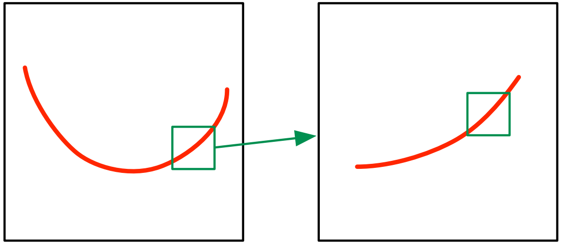
Gradient descent: rationale

"Differentiable" $\implies$ "locally linear".



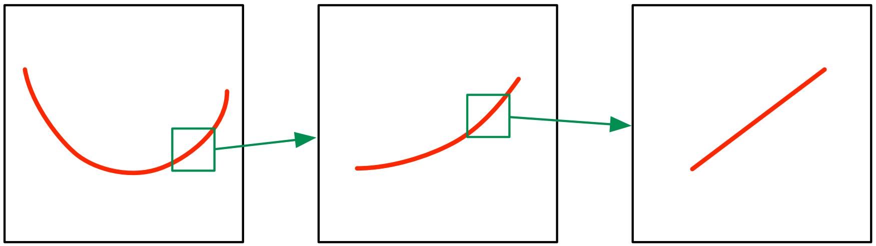
Gradient descent: rationale

"Differentiable" $\implies$ "locally linear".



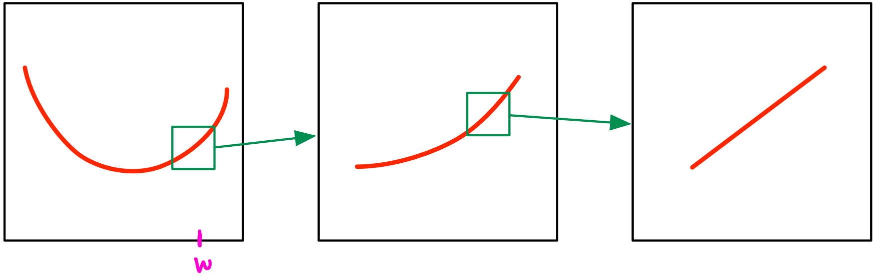
Gradient descent: rationale

“Differentiable” $\Longrightarrow$ “locally linear”.



Gradient descent: rationale

"Differentiable" $\implies$ "locally linear".



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

$$L(w+u) \;\approx\; L(w) + u \cdot \nabla L(w)$$

It is always okay to move in the direction of the negative gradient

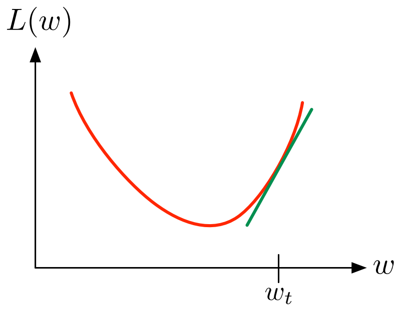
Therefore, if $u = -\eta \nabla L(w)$ is small,

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

$$\approx \; L(w) + (-\eta \nabla L(w)) \cdot \nabla L(w)$$

The step size matters

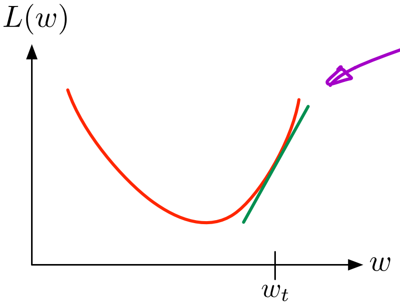
Gradient descent update: $w_{t+1} = w_t - \eta_t \nabla L(w_t)$.



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

The step size matters

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



Gradient > 0
∴ Move to the left.
But by how much?

Some options for choosing step size η_t:

① Fixed value, like $\eta_t = 0.1$

② Fixed schedule, like
$\eta_t = 1/t$

③ Line search

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

One option: pick η_t using a line search

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

pick the best step size

Example: logistic regression

$w = (w_1, \ldots, w_d)$

$\mathbb{R}^d$ -1 or +1

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

Find parameter $w \in \mathbb{R}^d$ that minimizes this loss function

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

$w_1 x_1^{(i)} + w_2 x_2^{(i)} + \ldots + w_d x_d^{(i)}$

What is the derivative?

$$\frac{dL}{dw_j} = \sum_{i=1}^{n} \frac{d}{dw_j}\left(\ln\left(1 + e^{-y^{(i)}(w \cdot x^{(i)})}\right)\right)$$

$$= \sum_{i=1}^{n} \frac{\frac{d}{dw_j}\left(1 + e^{-y^{(i)}(w \cdot x^{(i)})}\right)}{1 + e^{-y^{(i)}(w \cdot x^{(i)})}}$$

$$= \sum_{i=1}^{n} \frac{-e^{-y^{(i)}(w \cdot x^{(i)})} \frac{d}{dw_j}\left(y^{(i)}(w \cdot x^{(i)})\right)}{1 + e^{-y^{(i)}(w \cdot x^{(i)})}}$$

derivative of sum = sum of derivatives

$d(\ln u) = \frac{du}{u}$

$d(e^{-u}) = -e^{-u} du$

$$\frac{dL}{dw_j} = -\sum_{i=1}^{n} \frac{e^{-y^{(i)}(w\cdot x^{(i)})}}{1+e^{-y^{(i)}(w\cdot x^{(i)})}} \frac{d}{dw_j}\left(y^{(i)}(w\cdot x^{(i)})\right)$$

$$= -\sum_{i=1}^{n} \frac{e^{-y^{(i)}(w\cdot x^{(i)})}}{1+e^{-y^{(i)}(w\cdot x^{(i)})}} \; y^{(i)} x_j^{(i)}$$

this is $\Pr_w(-y^{(i)} \mid x^{(i)})$

$$\frac{dL}{dw_j} = -\sum_{i=1}^{n} \Pr_w(-y^{(i)} \mid x^{(i)}) \; y^{(i)} x_j^{(i)}$$

$$\Rightarrow \boxed{\nabla L(w) = -\sum_{i=1}^{n} \Pr_w(-y^{(i)} \mid x^{(i)}) \; y^{(i)} x^{(i)}}$$

Gradient descent for logistic regression

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

$$w_{t+1} = w_t + \eta_t \sum_{i=1}^{n} y^{(i)} x^{(i)} \underbrace{\Pr_{w_t}(-y^{(i)}|x^{(i)})}_{\text{doubt}_t(x^{(i)}, y^{(i)})}$$

Gradient descent for logistic regression

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

$$w_{t+1} \;=\; w_t \;+\; \eta_t \sum_{i=1}^{n} y^{(i)} x^{(i)} \underbrace{\Pr_{w_t}(-y^{(i)} | x^{(i)})}_{\text{doubt}_t(x^{(i)}, y^{(i)})}$$

How to set step size η_t?

A variant of gradient descent

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

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

Each update involves the entire data set, which is inconvenient.

A variant of gradient descent

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

could be a million or more!

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

Could be very slow if n (# data points) is large

Each update involves the entire data set, which is inconvenient.

Stochastic gradient descent: update based on just one point:

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

very fast updates

Decomposable loss functions

Loss function for logistic regression:

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

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.

LR: $\ell(w; x, y) = \ln(1 + e^{-y(w \cdot x)})$

Gradient descent and stochastic gradient descent

For minimizing

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

Gradient descent and stochastic gradient descent

For minimizing

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

Gradient descent:

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

Gradient descent and stochastic gradient descent

For minimizing

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

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

Gradient descent:

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

Stochastic gradient descent:

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

Variant: mini-batch stochastic gradient descent

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
  - $w_{t+1} = w_t - \eta_t \sum_{(x,y) \in B} \nabla \ell(w_t; x, y)$