Unconstrained optimization

Topics we'll cover

1. Optimization by local search
2. Gradient descent
3. Stochastic gradient descent

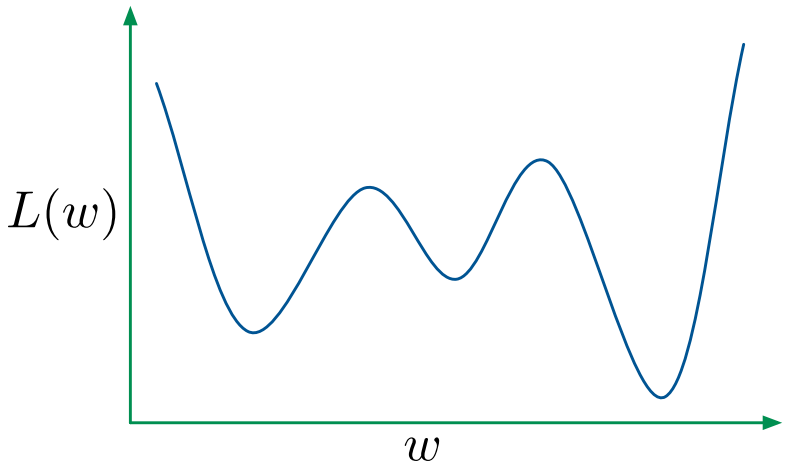
Minimizing a loss function

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**.

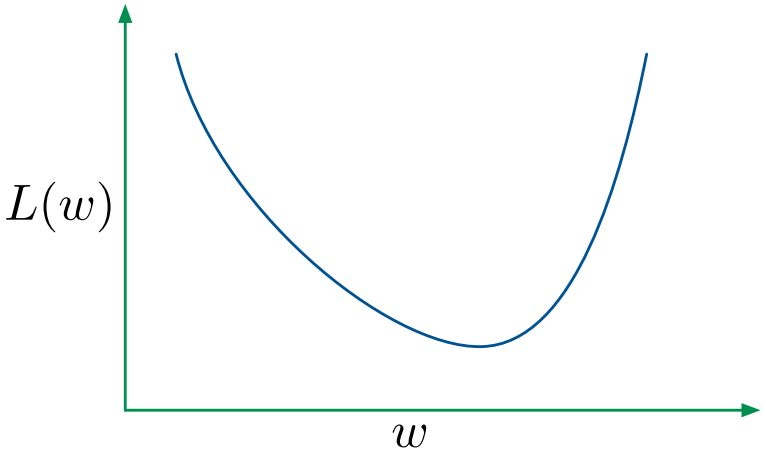
Local search



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

A good situation for local search

When the loss function is **convex**:



Idea for picking search direction:
Look at the **derivative** of $L(w)$ at the current point 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.

Multivariate differentiation

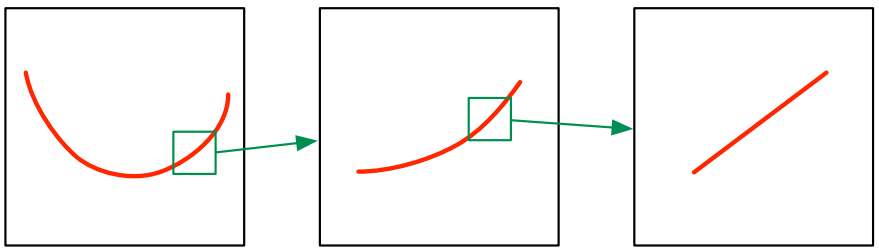
Example: $w \in \mathbb{R}^3$ and $F(w) = 3w_1w_2 + w_3$.

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

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

Gradient descent: rationale

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



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

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

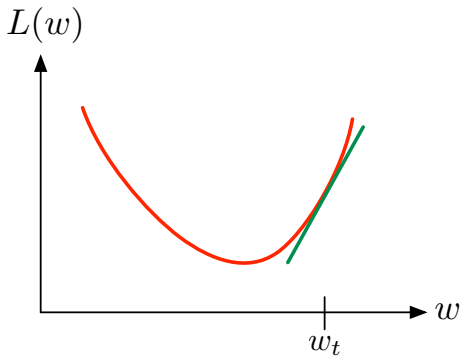
Therefore, if $u = -\eta \nabla L(w)$ is small,

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

The step size matters

Update rule: $w_{t+1} = w_t - \eta_t \nabla L(w_t)$

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



Some choices:

- Set η_t according to a fixed schedule, like $1/t$
- Choose by line search to minimize $L(w_{t+1})$

Example: logistic regression

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

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

What is the derivative?

Gradient descent for logistic regression

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

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

Gradient descent for large data sets?

- 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.

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: for data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$,

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

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

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:
  - $w_{t+1} = w_t - \eta_t \sum_{i=1}^{n} \nabla \ell(w_t; x^{(i)}, y^{(i)})$

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