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

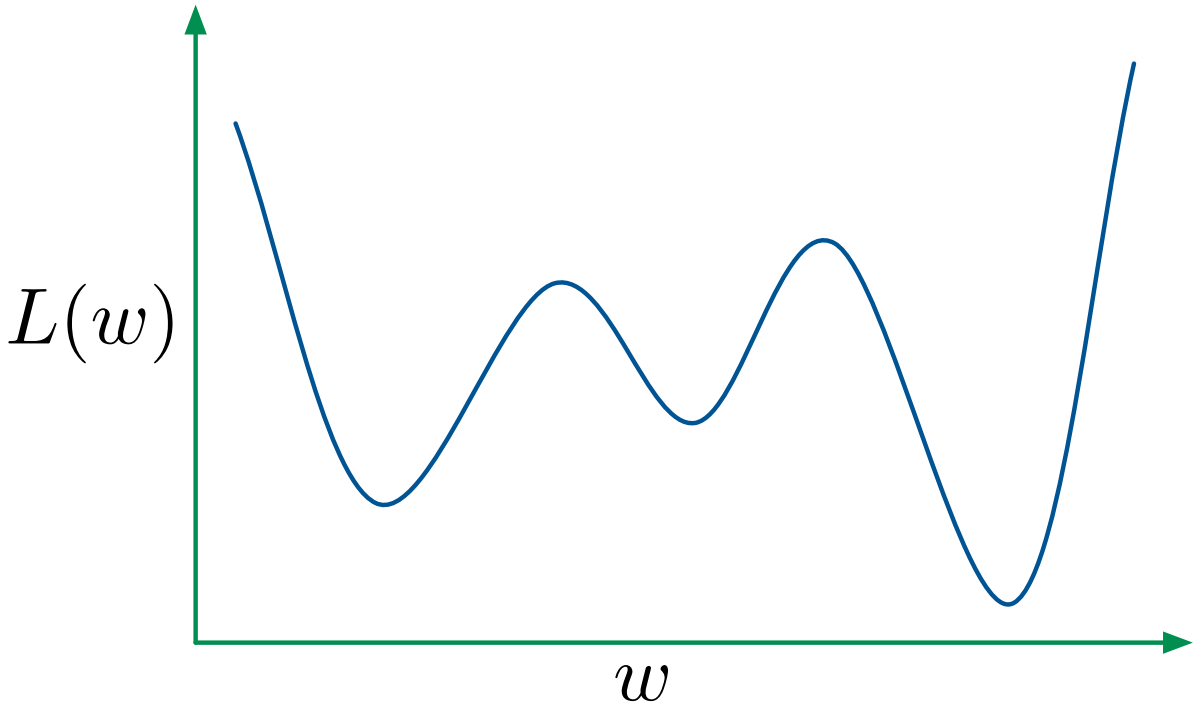
Minimizing a loss function

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**.

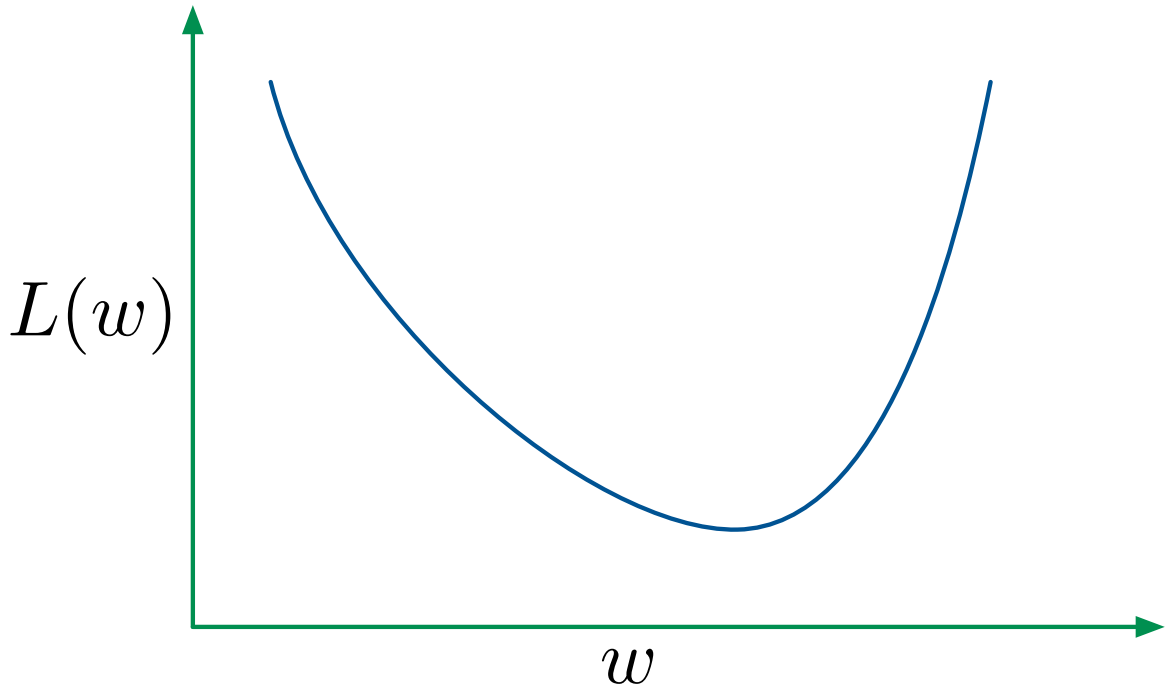
Local search



- 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

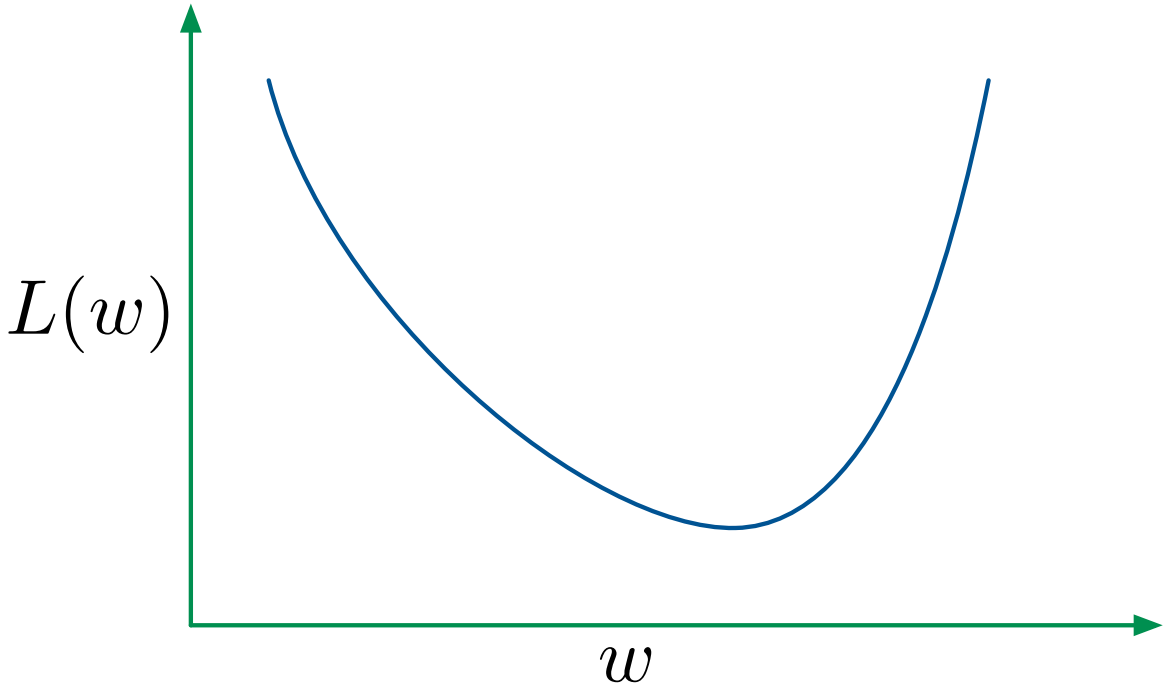
When the loss function is **convex**:



E.g. least-squares regression and logistic regression.

Choosing a search direction

One idea: look at the **derivative** of $L(w)$ at the current point w.



Multivariate differentiation

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

Multivariate differentiation

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

$$\frac{\partial F(w)}{\partial w_1} = 3w_2$$
$$\frac{\partial F(w)}{\partial w_2} = 3w_1$$
$$\frac{\partial F(w)}{\partial w_3} = 1$$
$$\nabla F(w) = \begin{bmatrix} 3w_2 \\ 3w_1 \\ 1 \end{bmatrix}$$

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

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

$$F(w) = \sum_{i=1}^{d} w_i x_i$$

$$\frac{\partial F(w)}{\partial w_k} = x_k$$

$$\nabla F(w) = x$$

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

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

$$F(w) = \sum_{i=1}^{d} w_i^2$$

$$\frac{\partial F(w)}{\partial w_k} = 2w_i$$

$$\nabla F(w) = 2w$$

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

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

$$
\begin{aligned}
F(w) &= \sum_{1 \le i,j \le d} w_i w_j M_{ij} \\
\frac{\partial F(w)}{\partial w_k} &= \sum_{1 \le i \le d} w_i (M_{ik} + M_{ki}) \\
&= (M + M^T) w
\end{aligned}
$$

Gradient descent

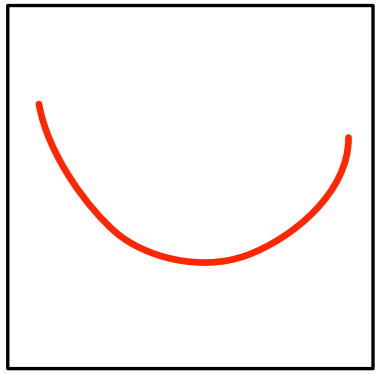
For minimizing a function $L(w)$:

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

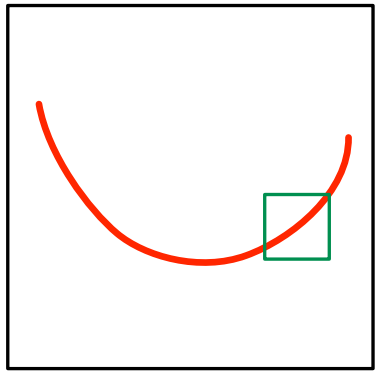
Gradient descent: rationale

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



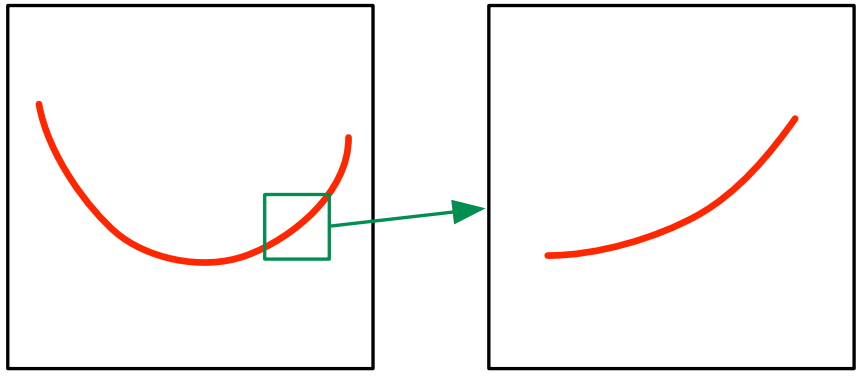
Gradient descent: rationale

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



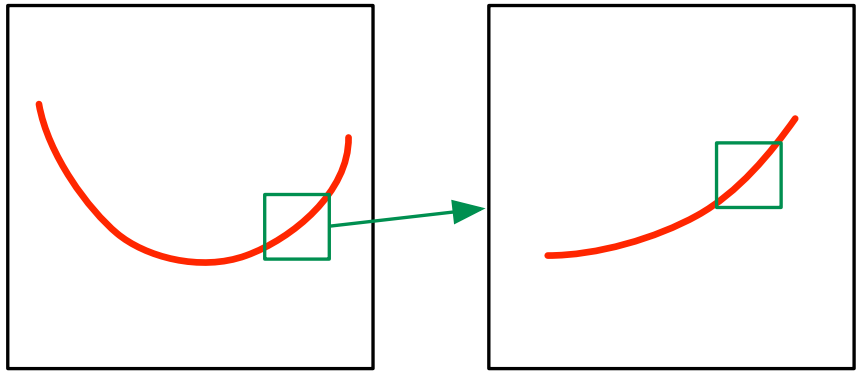
Gradient descent: rationale

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



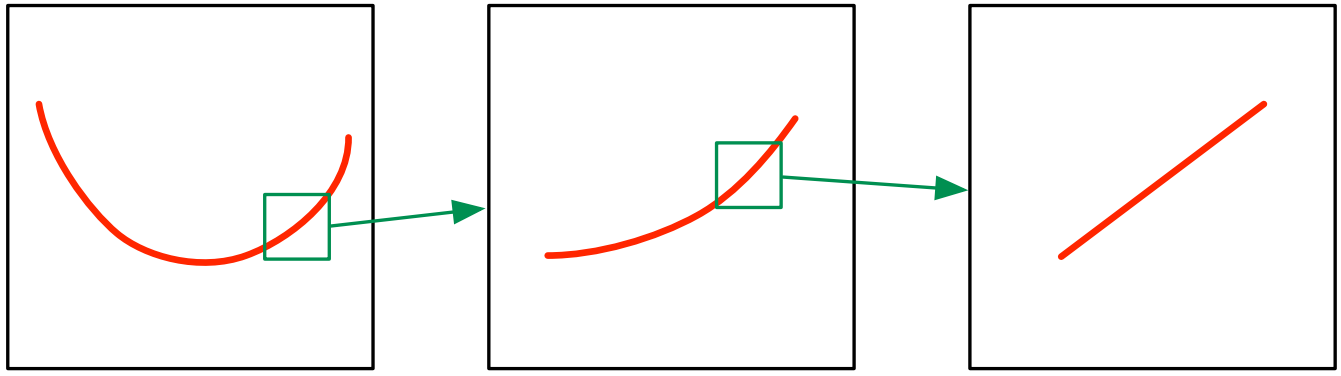
Gradient descent: rationale

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



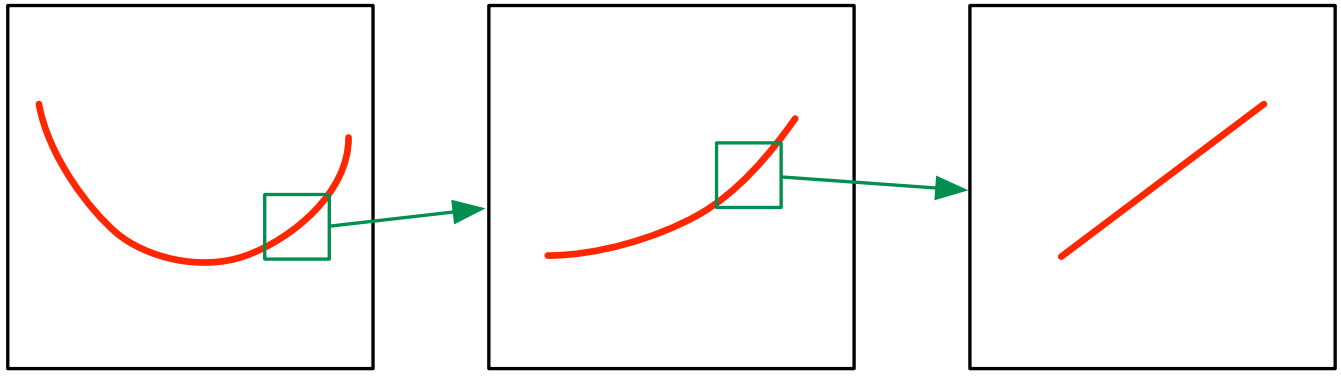
Gradient descent: rationale

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



Gradient descent: rationale

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



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

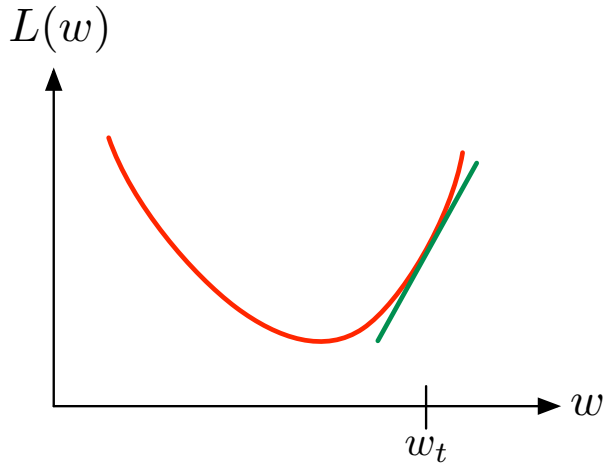
$$L(w+u) \approx L(w) + u \cdot \nabla L(w) \quad .$$

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

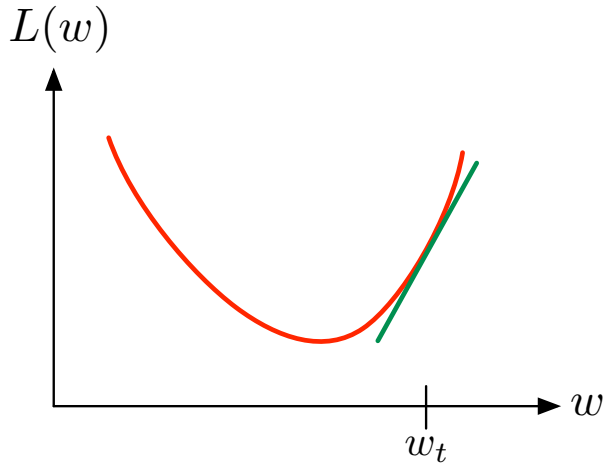
Gradient descent update: $w_{t+1} = w_t - \eta_t \nabla L(w_t)$.



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

The step size matters

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



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

One option: pick η_t using a line search

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

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?

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?

$$\begin{aligned}
\frac{\partial L(w)}{\partial w_k} &= \sum_{i=1}^{n} \frac{1}{1 + e^{-y^{(i)}(w \cdot x^{(i)})}} \frac{\partial}{\partial w_k}(1 + e^{-y^{(i)}(w \cdot x^{(i)})}) \\
&= \sum_{i=1}^{n} \frac{1}{1 + e^{-y^{(i)}(w \cdot x^{(i)})}} e^{-y^{(i)}(w \cdot x^{(i)})} \frac{\partial}{\partial w_k}(-y^{(i)}(w \cdot x^{(i)})) \\
&= -\sum_{i=1}^{n} y^{(i)} x_k^{(i)} \frac{e^{-y^{(i)}(w \cdot x^{(i)})}}{1 + e^{-y^{(i)}(w \cdot x^{(i)})}}
\end{aligned}$$

Gradient descent for logistic regression

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

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

Many choices for step size η_t:

- Choose by line search to minimize $L(w_{t+1})$
- Fix it beforehand
- Fix it to a value, and halve upon convergence

Variant: stochastic gradient descent

Gradient descent update for logistic regression: at time t

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

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

Variant: stochastic gradient descent

Gradient descent update for logistic regression: at time t

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

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), e.g., cycling through data set
- $w_{t+1} = w_t + \eta_t \, y \, x \Pr_{w_t}(-y|x)$

Convenient for very large data sets.

Variant: stochastic gradient descent

Gradient descent update for logistic regression: at time t

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

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), e.g., cycling through data set
- $w_{t+1} = w_t + \eta_t \, y \, x \Pr_{w_t}(-y|x)$

Convenient for very large data sets.

Or: make updates based on **mini-batches** of data points.