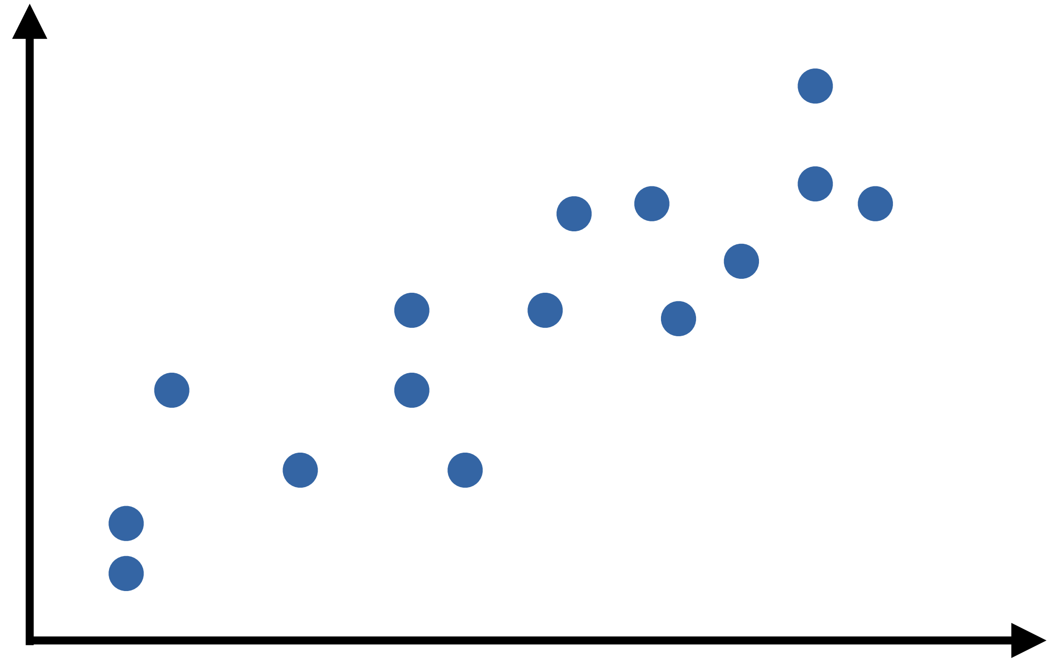
Optimization & Gradient Descent

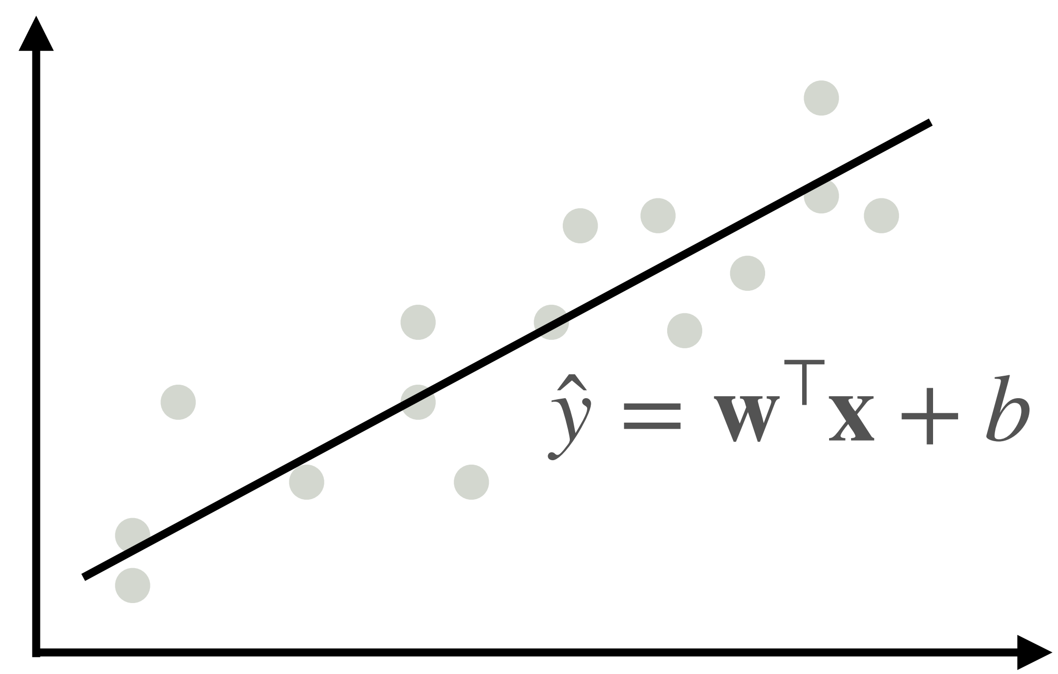
Data

- Input: $\mathbf{x}$ (tensor)
- Output: y



Model

- Structure: e.g. Linear, Logistic or multinomial logistic regression
- Parameters: θ (e.g. $\mathbf{w}, b$)



Loss function

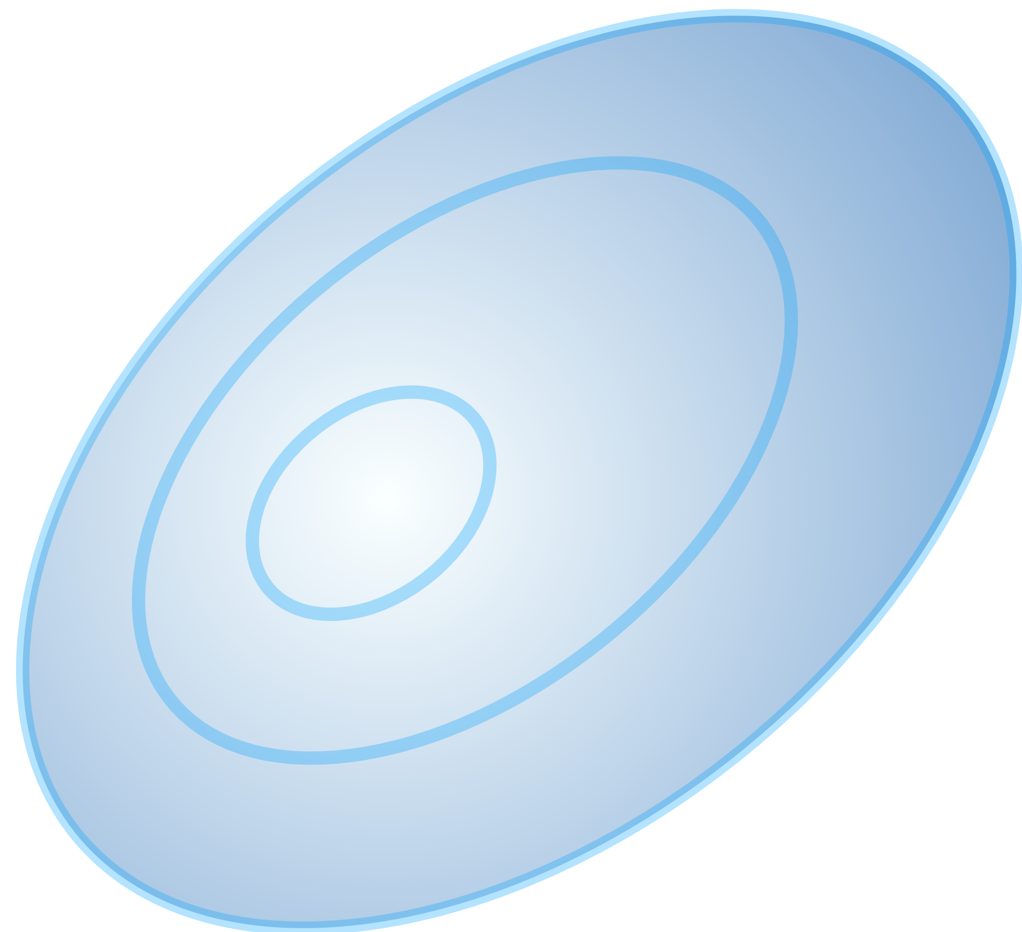
- Loss: $L(\theta) = \sum_i \ell(\theta \,|\, \mathbf{x}_i, y_i)$

Gradients

- L2 $\ell(\hat{y}, y) = \frac{1}{2}\|\hat{y} - y\|^2$ where $\hat{y} = \mathbf{w}^\top \mathbf{x} + b$
  - $\nabla_{\mathbf{w}} \ell(\hat{y}, y) = (\hat{y} - y)\mathbf{x}$
    $\nabla_b \ell(\hat{y}, y) = \hat{y} - y$
- Sigmoid $\ell(o, y) = -\log p(y \mid o)$ where $o = \mathbf{w}^\top \mathbf{x} + b$
  - $\nabla_{\mathbf{w}} \ell(o, y) = (\sigma(o) - y)\mathbf{x}$
    $\nabla_b \ell(o, y) = \sigma(o) - y$
- Softmax $\ell(\mathbf{o}, y) = -\log p(y \mid \mathbf{o})$ where $\mathbf{o} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$
  - $\nabla_{\mathbf{W}_i} \ell(\mathbf{o}, y) = (\mathrm{softmax}(\mathbf{o})_i - [y = i])\mathbf{x}$
    $\nabla_{\mathbf{b}_i} \ell(\mathbf{o}, y) = \mathrm{softmax}(\mathbf{o})_i - [y = i]$

Optimization

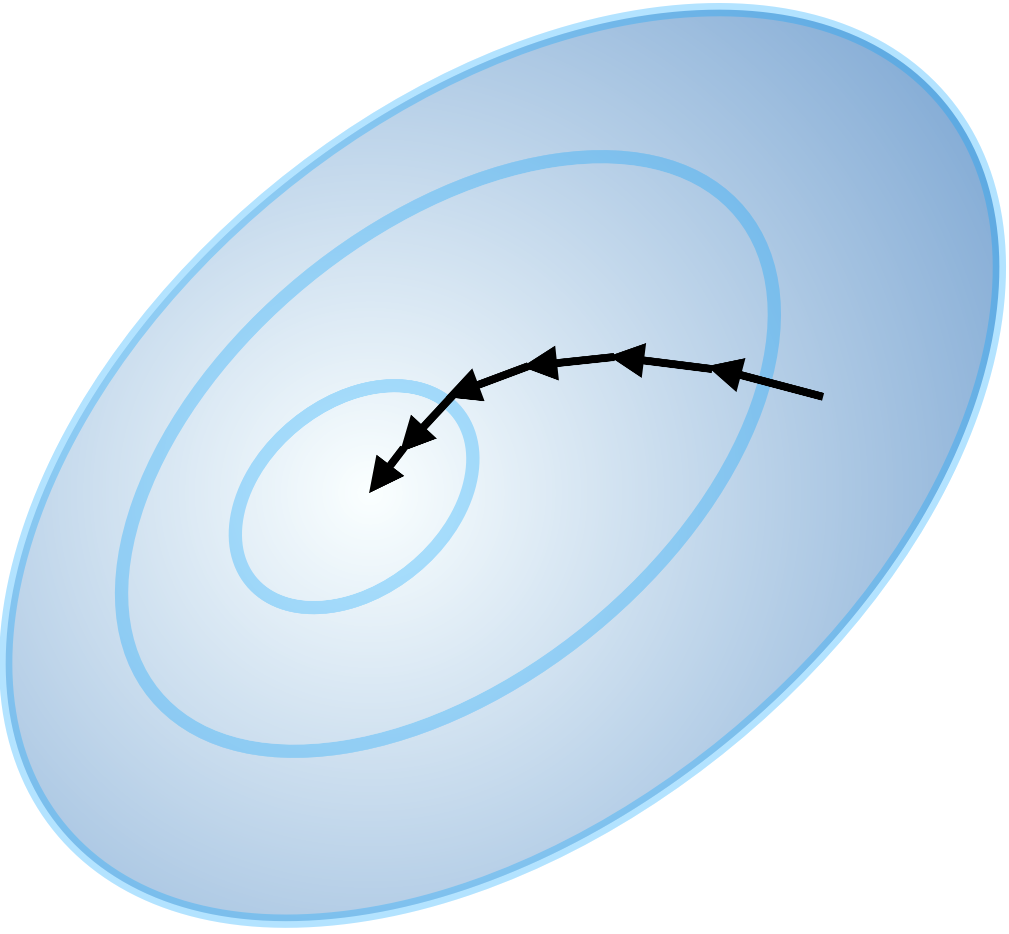
- Find parameters θ
- With lowest loss $L(\theta)$



Gradient descent

- Start at random θ
- Update: $\theta' = \theta - \epsilon \frac{\partial L(\theta)}{\partial \theta}$
- $L(\theta') < L(\theta)$ if $\frac{\partial L(\theta)}{\partial \theta} != 0$

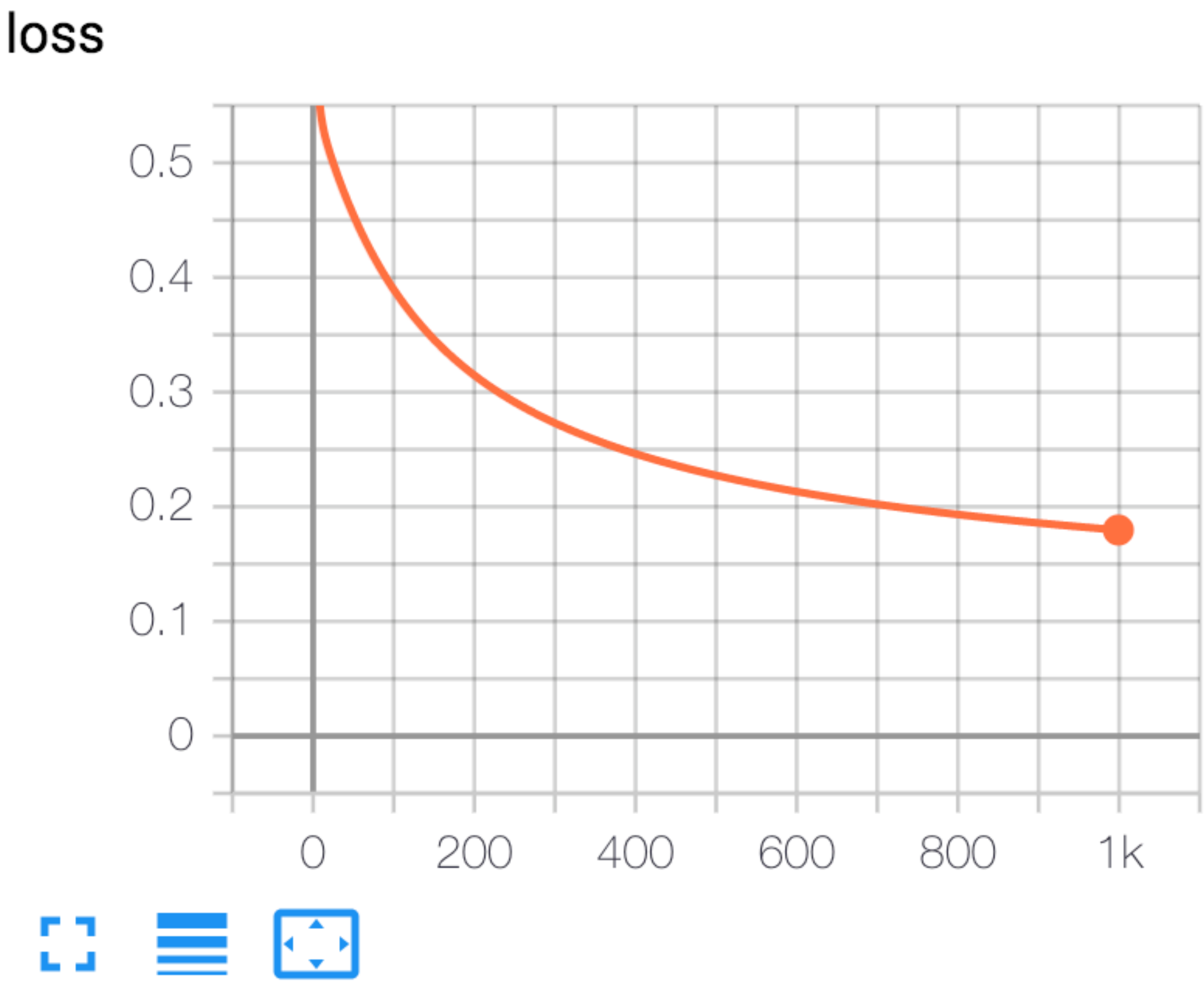
  and ϵ small enough



Gradient descent algorithm

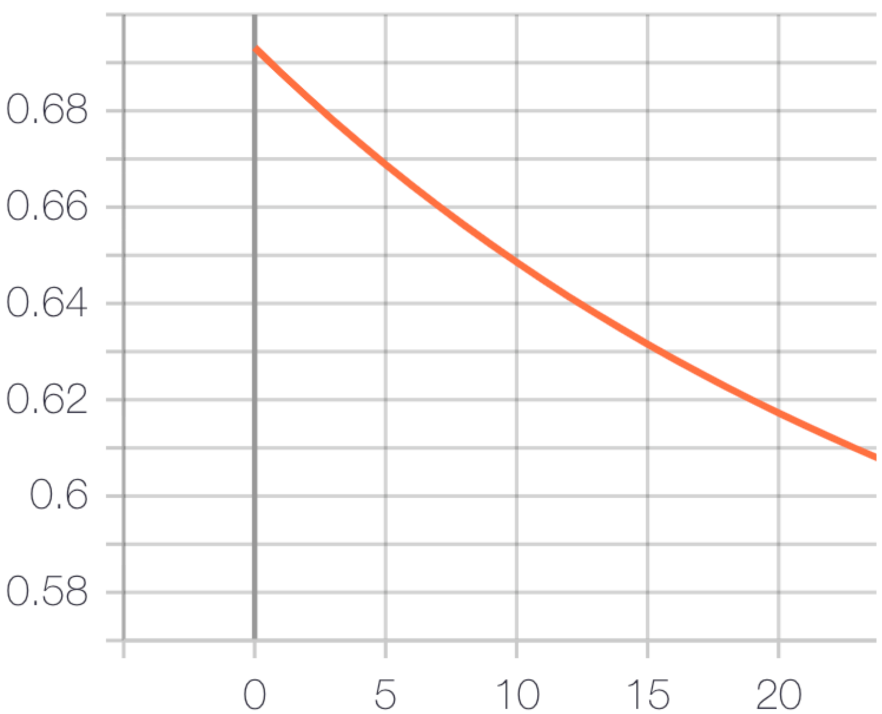
- Randomly initialize θ
- for n iterations
  - compute loss $L(\theta)$ and gradient $g = \frac{\partial L(\theta)}{\partial \theta}$
  - Update: $\theta -= \epsilon g$

Gradient descent in action

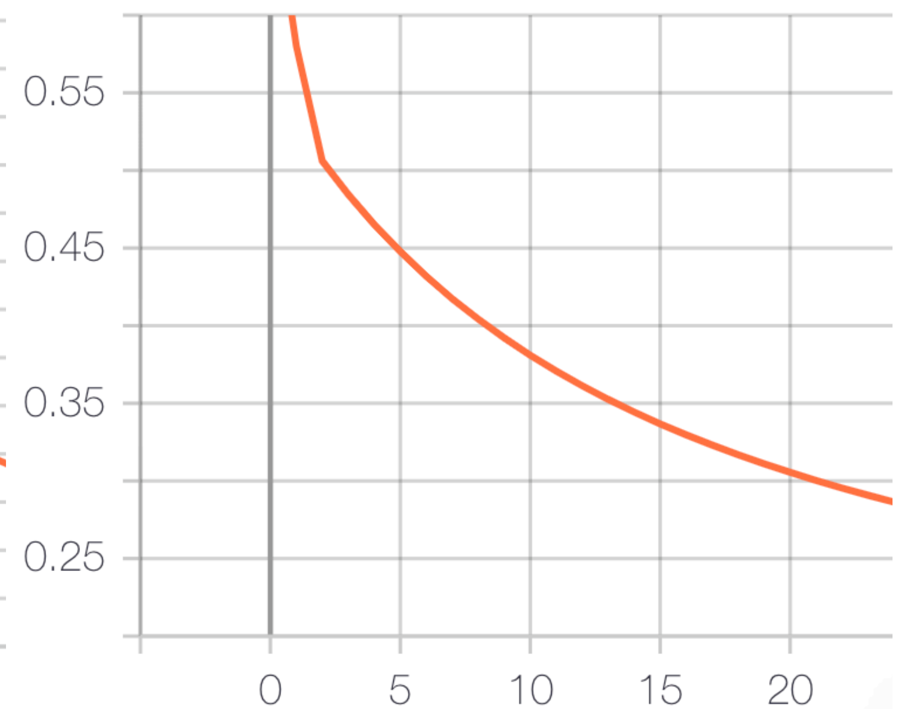


Learning rate matters

Too small



Just right



Too big