Lecture 2 recap

Slides

- We make the slides available on this website

http://vision.in.tum.de/teaching/ss2017/dl4cv/coursematerial

- Password: dl4cvTUM

- Please do not distribute!

Machine learning

Unsupervised learning

Supervised learning

Reinforcement learning

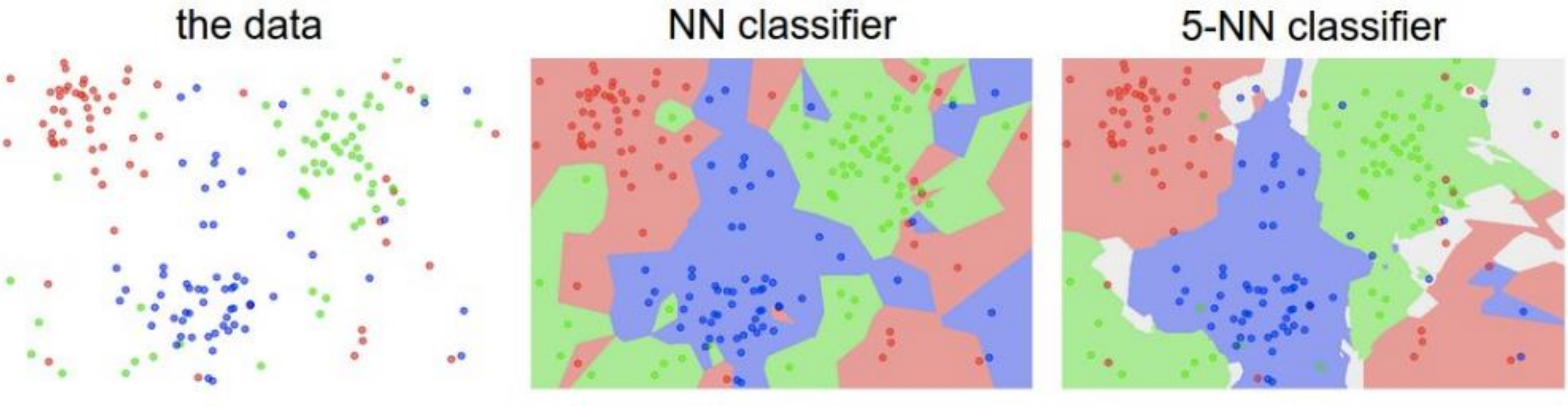
Agents → interaction → Environment

Machine learning

- How can we learn to perform image classification?

Task	Performance measure	Experience
Image classification	Accuracy	Data

Nearest Neighbor

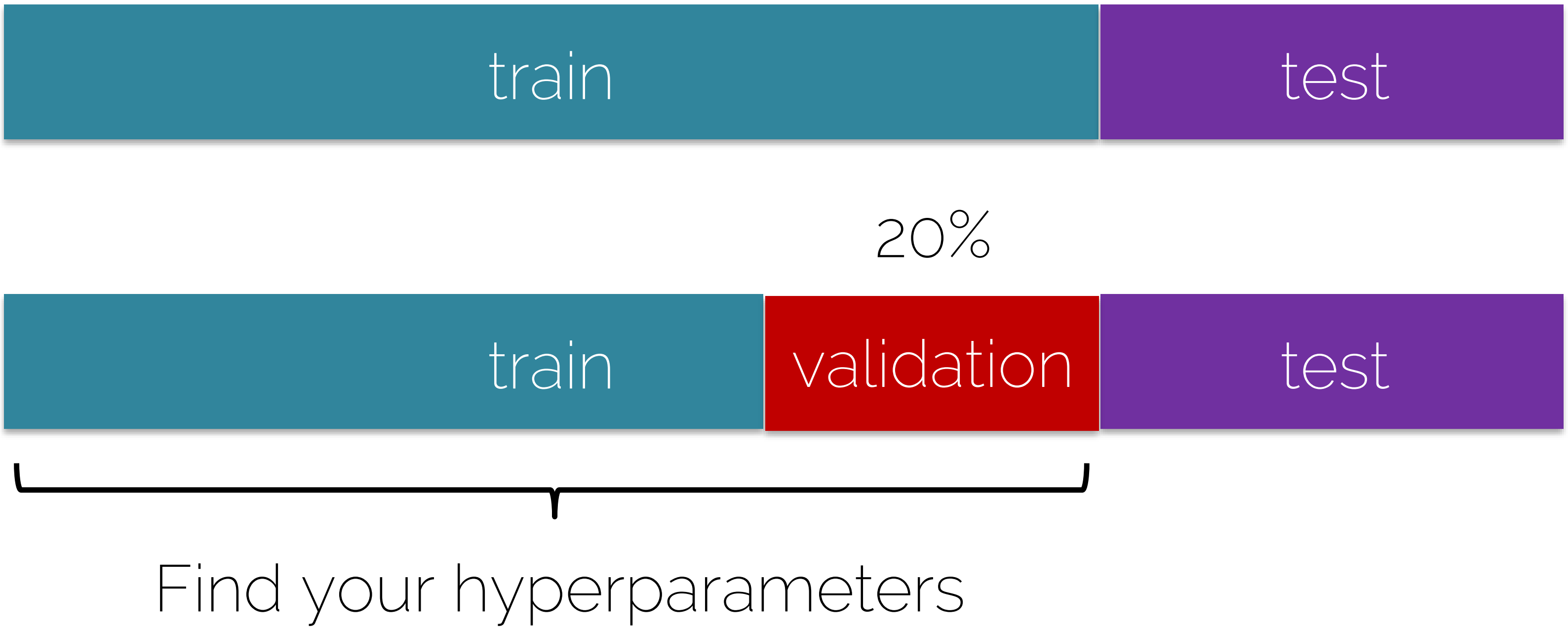


What is the performance on training data for NN classifier?

What classifier is more likely to perform best on test data?

Courtesy of Stanford course cs231n

Cross validation

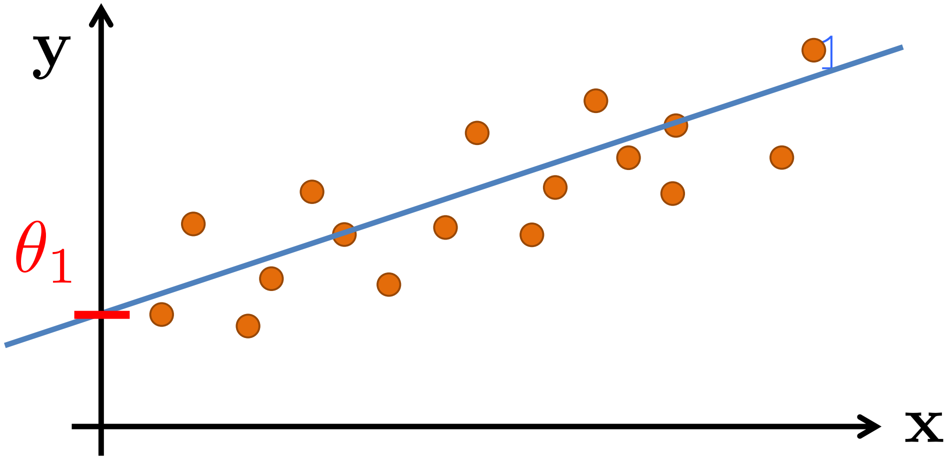


Linear prediction

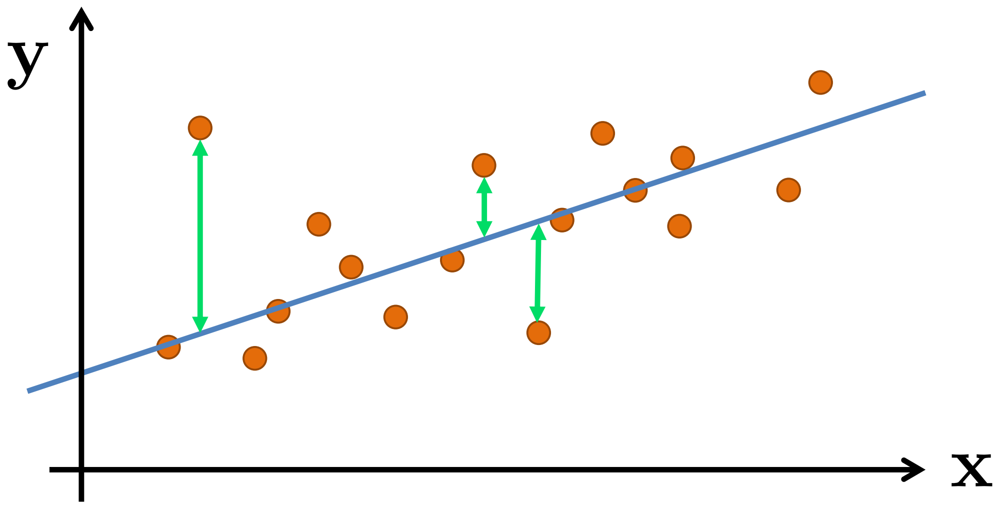
- A linear model is expressed in the form

$$\hat{y}_i = \sum_{j=1}^{d} x_{ij}\theta_j = x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{id}\theta_d$$

bias



Linear regression



Minimizing

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Objective function
Energy
Loss

Optimization

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y} = 0$$

$$\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Inputs: Outside temperature, number of people...

Output: Temperature of the building

Maximum Likelihood Estimate

$p_{data}(\mathbf{x})$ True underlying distribution

$p_{model}(\mathbf{x}; \boldsymbol{\theta})$ Parametric family of distributions

Controlled by a parameter

Back to linear regression

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

$$-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{x}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{x}\boldsymbol{\theta})$$

$$\downarrow \frac{\partial}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

How can we find the estimate of theta?

Back to linear regression

$$-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{x}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{x}\boldsymbol{\theta})$$

Can you derive the estimate of sigma?

Back to linear regression

$$\frac{\partial}{\partial \sigma}\left(-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{x}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{x}\boldsymbol{\theta})\right) = 0$$

$$-\frac{1}{\sigma^3}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})$$

Back to linear regression

$$\frac{\partial}{\partial \sigma}\left(-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{x}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{x}\boldsymbol{\theta})\right) = 0$$

Chain rule

$$F(x) = f(g(x))$$

$$\frac{\partial}{\partial x}F(x) = \frac{\partial}{\partial g(x)}f(g(x))\frac{\partial}{\partial x}g(x)$$

Back to linear regression

$$\frac{\partial}{\partial \sigma}\left(-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{x}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{x}\boldsymbol{\theta})\right) = 0$$

$$\frac{\partial}{\partial x}\log_b(x) = \frac{1}{x \ln b}$$

$$-\frac{n}{2 * 2\pi\sigma^2} \quad 4\pi\sigma = -\frac{n}{\sigma}$$

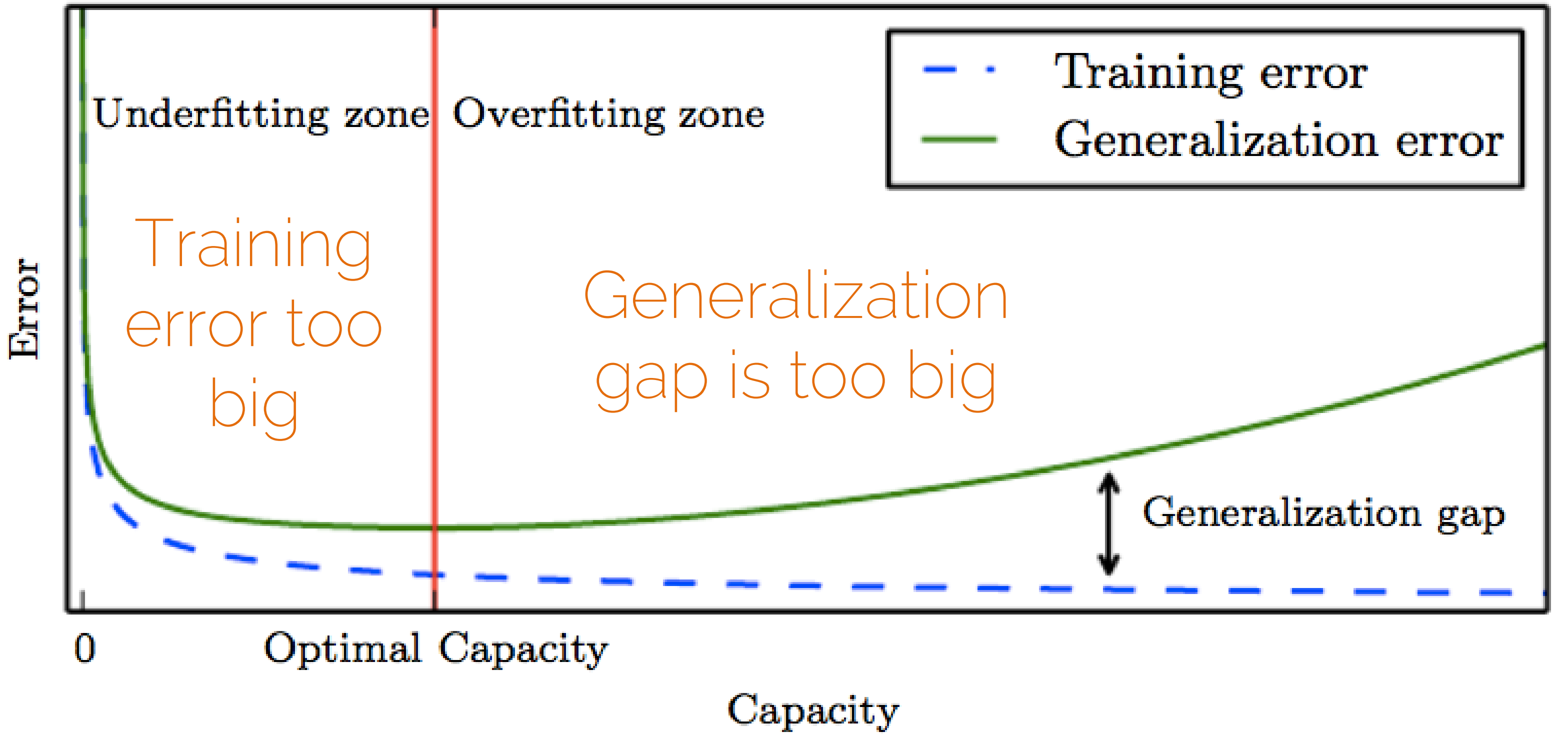
Back to linear regression

$$\frac{\partial}{\partial\sigma}\left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})\right) = 0$$

$$= -\frac{n}{\sigma} - \frac{1}{\sigma^3}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}) = 0$$

$$\sigma^2 = \frac{1}{n}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})$$

Overfitting and underfitting



Credits: Deep Learning. Goodfellow et al.

Regularization

Loss $$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

L2 regularization

L1 regularization

Max norm regularization

Dropout

Can you find the relationship between this loss and the Maximum a Posteriori (MAP) estimate?

Maximum a Posteriori (MAP)

- We want to have a point estimate (as opposed to ML)
- Find the point of maximum posterior probability

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y})$$

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

Note the difference

Maximum a Posteriori (MAP)

- We want to have a point estimate (as opposed to ML)
- Find the point of maximum posterior probability

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y}) = \arg\max_{\theta} p(\mathbf{y}|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta})$$

Bayes rule $$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y})}$$

Maximum a Posteriori (MAP)

- We want to have a point estimate (as opposed to ML)
- Find the point of maximum posterior probability

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y}) \;= \arg\max_{\theta} p(\mathbf{y}|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta})$$

Bayes rule $$p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X})}$$

Maximum a Posteriori (MAP)

- We want to have a point estimate (as opposed to ML)
- Find the point of maximum posterior probability

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y}) = \arg\max_{\theta} p(\mathbf{y}|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta})$$

Recognition HARD

Generation EASY

Prior of the model

Maximum a Posteriori (MAP)

- We want to have a point estimate (as opposed to ML)
- Find the point of maximum posterior probability

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y}) \; = \arg\max_{\theta} p(\mathbf{y}|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta})$$

Maximum Likelihood Term

Maximum a Posteriori (MAP)

- We want to have a point estimate (as opposed to ML)
- Find the point of maximum posterior probability

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{X},\mathbf{y}) \; = \arg\max_{\theta} p(\mathbf{y}|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta})$$

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; 0, \frac{1}{\lambda}\mathbf{I}^2) \longrightarrow \lambda\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Regularization

Loss $$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

Prior of the model

Maximum Likelihood Estimate

Loss cheat sheet

- Softmax loss

$$L_i = \frac{e^{s_i}}{\sum_k e^{s_k}}$$

Scores or predictions

$$s_i = \mathbf{x}_i \boldsymbol{\theta}$$

- Multi-class SVM loss or Hinge loss

$$L_i = \sum_{j \neq y_i} \max(0, s_i - s_{y_i} + 1)$$

Optimization

Back to linear regression

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

$$\downarrow \frac{\partial}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Optimization

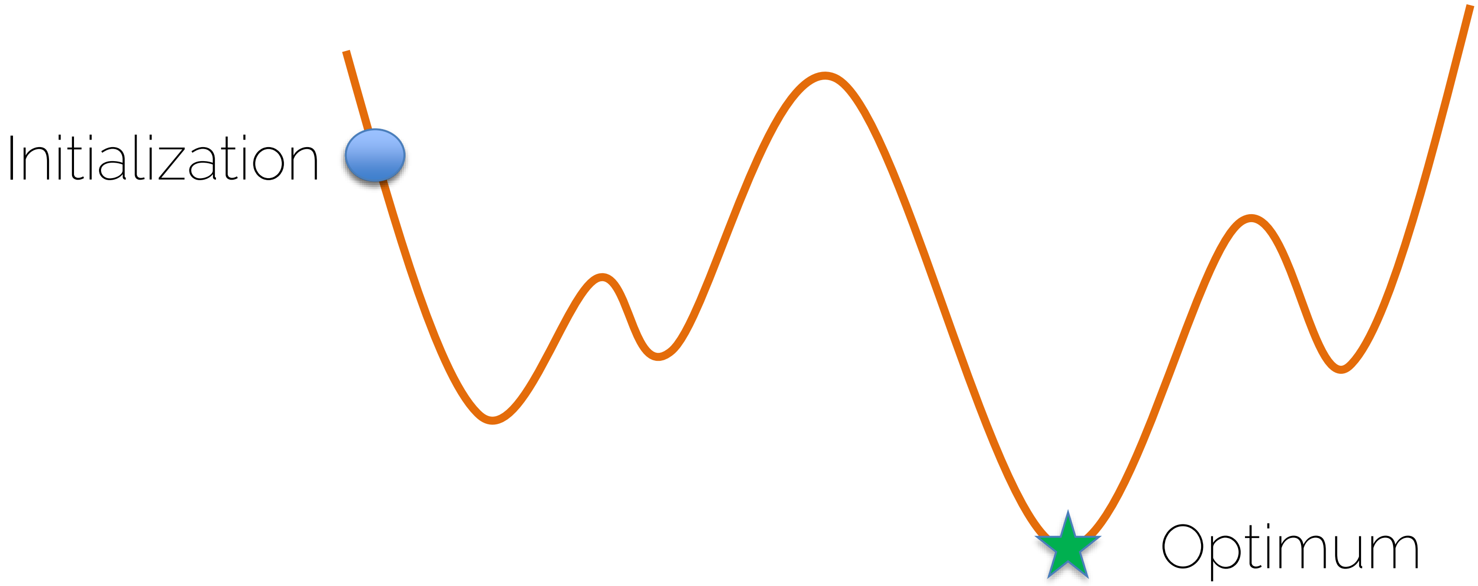
$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

- Complex function that cannot be derived in closed form
- Fast way to find a minimum
- Scales to large datasets

Gradient descent

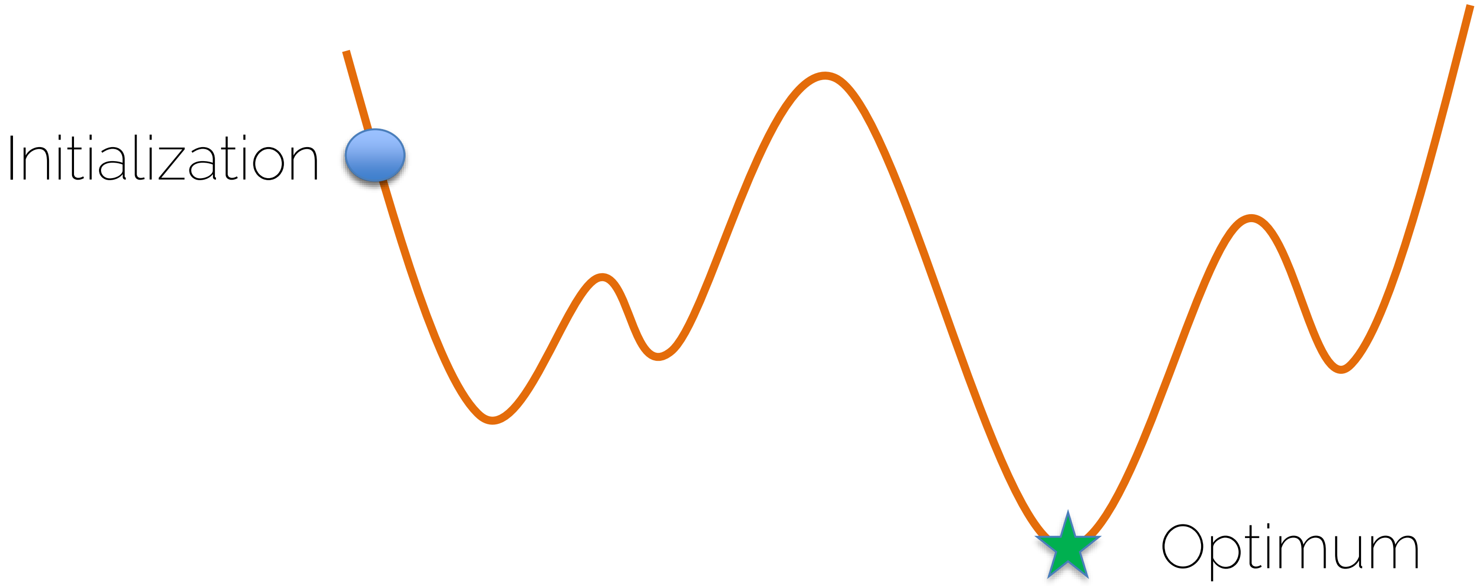
Following the slope

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



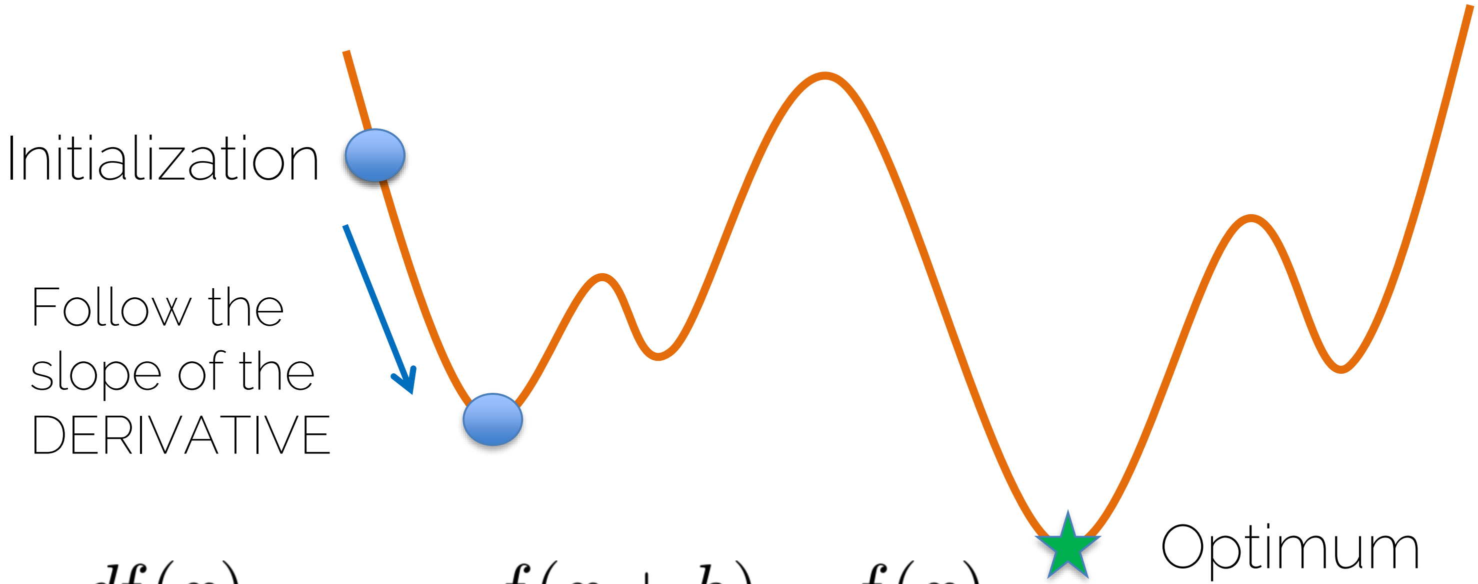
Following the slope

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



Following the slope

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

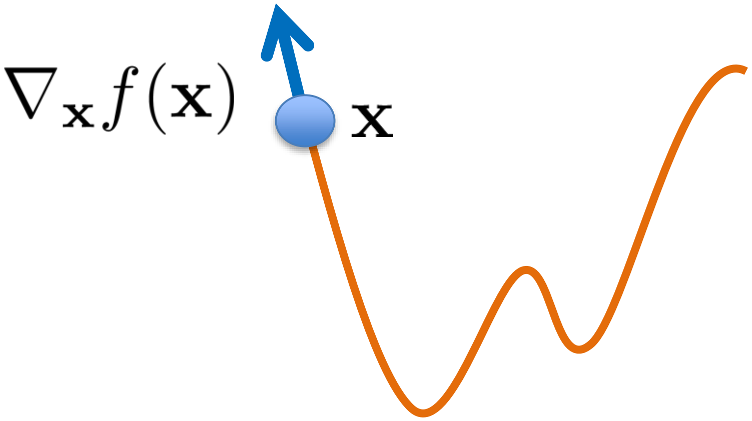
Gradient steps

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

Direction of greatest increase of the function

- Gradient steps in direction of negative gradient



$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

Learning rate

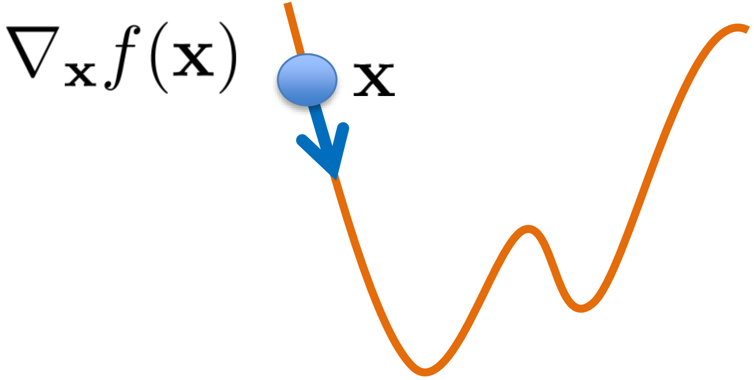
Gradient steps

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

Direction of greatest increase of the function

- Gradient steps in direction of negative gradient



$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

SMALL Learning rate

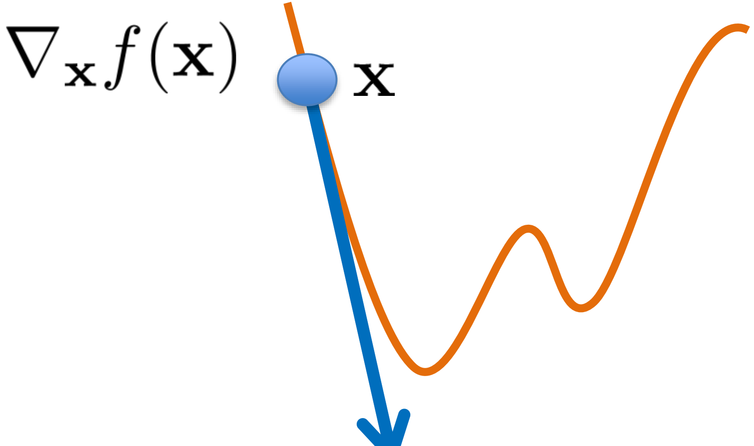
Gradient steps

Direction of greatest increase of the function

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

- Gradient steps in direction of negative gradient

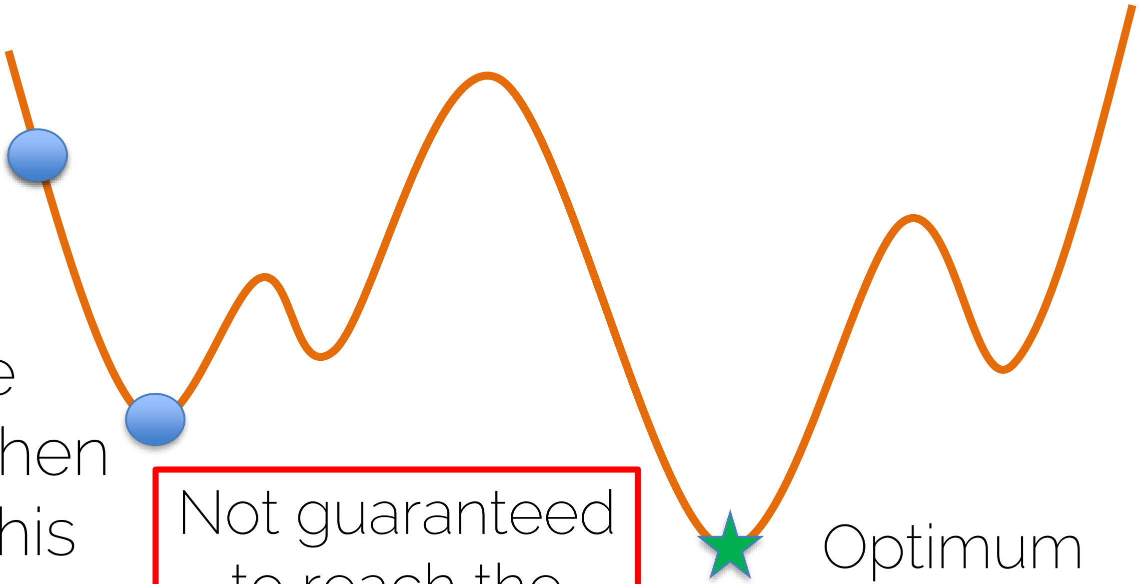


$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

LARGE Learning rate

Convergence

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



Numerical gradient

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Approximate
- Slow evaluation

Analytical gradient

- Exact and fast

Remember Linear Regression

$$f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$f(\boldsymbol{\theta}) = \frac{1}{n}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Analytical gradient → $2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y}$

Gradient descent for least squares

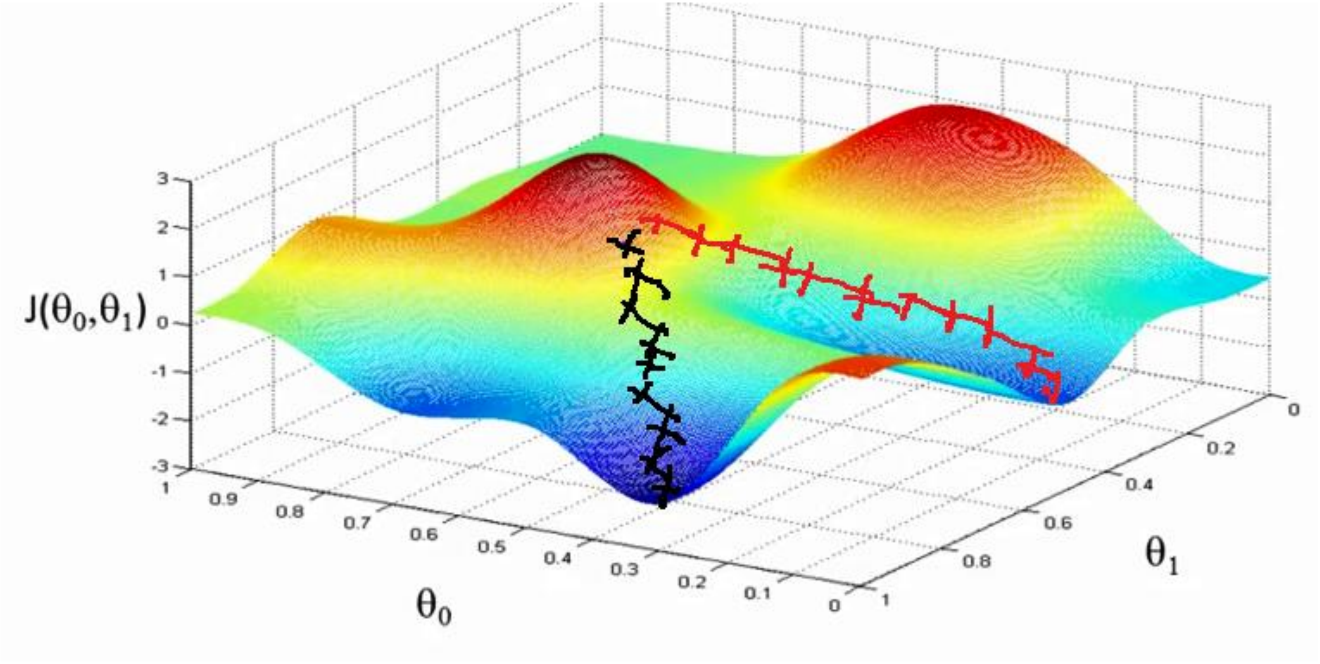
$$f(\boldsymbol{\theta}) = \frac{1}{n}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon\, 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y}$$

Convex, always converges to the same solution

Non-linear least squares

- Not necessarily convex



Stochastic Gradient Descent

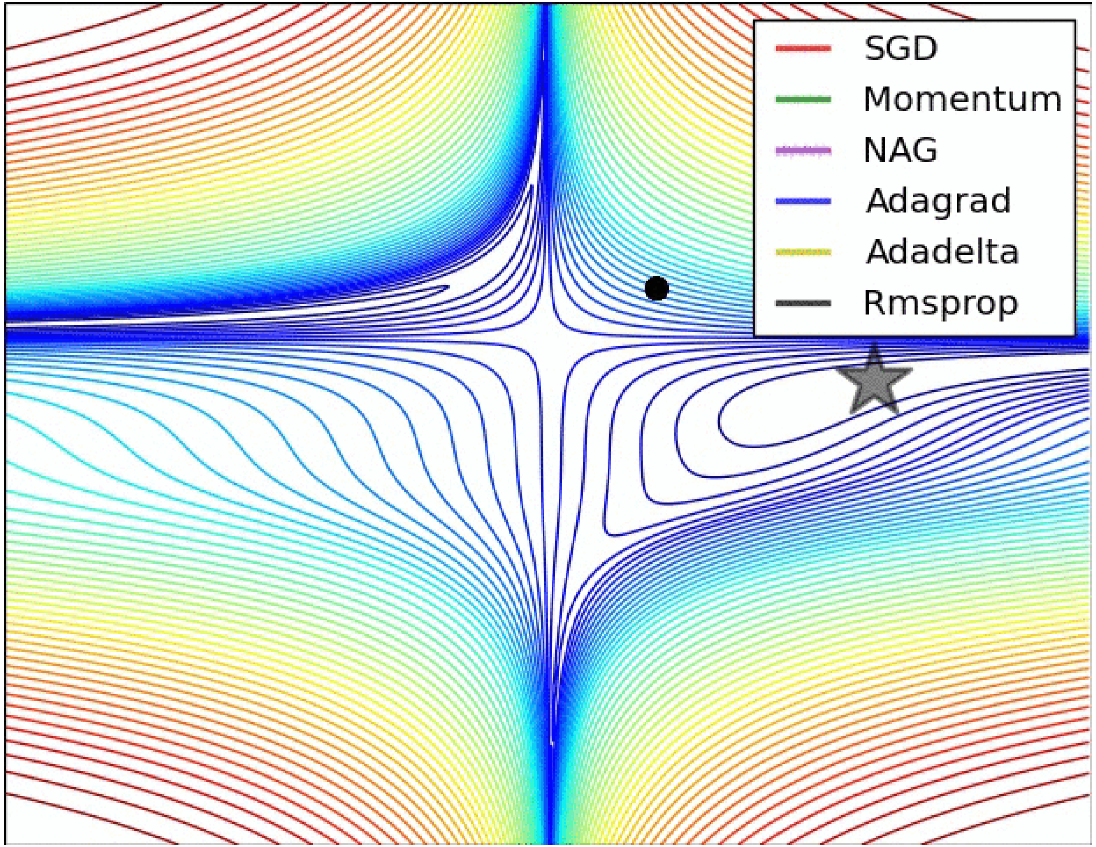
- If we have m training samples we need to compute the gradient for all of them which is $\mathcal{O}(m)$

- Gradient is an expectation, and so it can be approximated with a small number of samples

Minibatch $\mathbb{B} = \{x^1, \cdots, x^{m'}\}$

Epoch = complete pass through all the data

Convergence



Stochastic gradient descent

Gradient

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \frac{1}{m} \sum_i \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$$

Model

Loss

Ignore the sum for convenience ☺

SGD $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$

Momentum update

- Designed to accelerate training
- Define a new term called velocity $\mathbf{v}$

$$\mathbf{v}_{k+1} = \alpha \mathbf{v}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_{k+1}$$

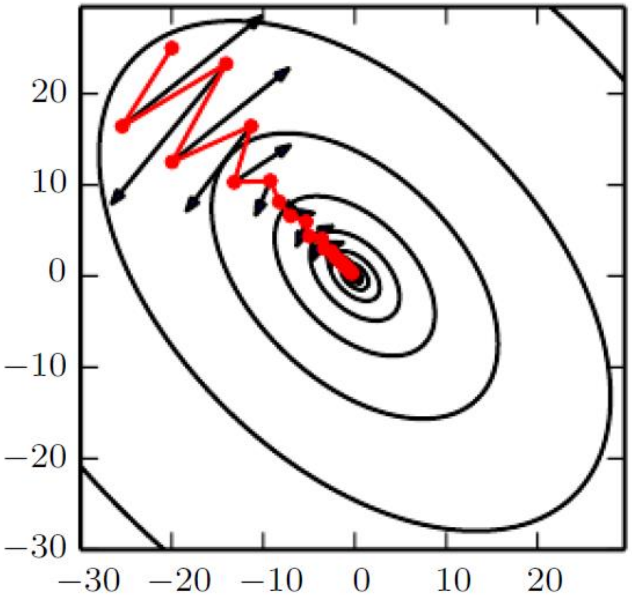
- The velocity accumulates gradients

SGD $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$

Polyack 1964

Momentum update

$$\mathbf{v}_{k+1} = \alpha \mathbf{v}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i) \qquad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_{k+1}$$

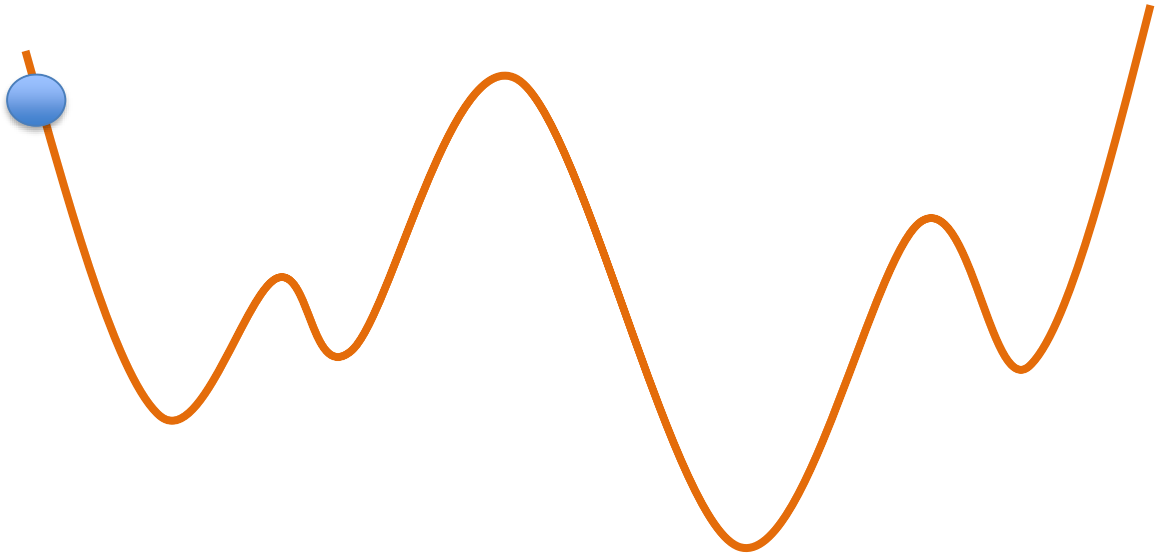


Step will be largest when a sequence of gradients all point to the same direction

Image: Goodfellow et al.

Momentum update

- Can it overcome local minima?



$$\mathbf{v}_{k+1} = \alpha \mathbf{v}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i) \quad \alpha = \{0.5, 0.9, 0.99\}$$

Nesterov's momentum

- *Look-ahead* momentum

$$\widetilde{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_k$$

$$\mathbf{v}_{k+1} = \alpha \mathbf{v}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\widetilde{\boldsymbol{\theta}}_{k+1}, \mathbf{x}^i, \mathbf{y}^i)$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_{k+1}$$

SGD $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$

Sutskever 2013, Nesterov 1983

Nesterov's momentum

- *Look-ahead* momentum

$$\widetilde{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_k$$

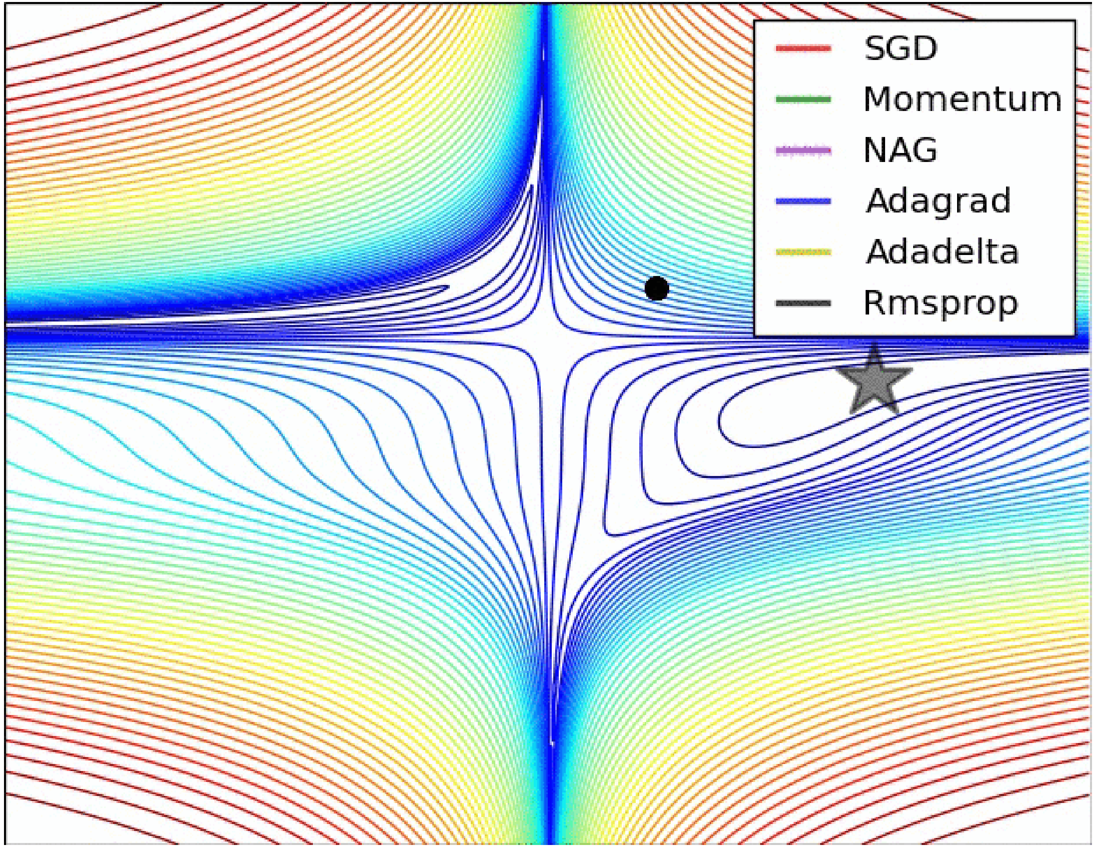
$$\mathbf{v}_{k+1} = \alpha \mathbf{v}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\widetilde{\boldsymbol{\theta}}_{k+1}, \mathbf{x}^i, \mathbf{y}^i)$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_{k+1}$$

SGD $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$

Sutskever 2013, Nesterov 1983

Convergence



More parameters...

$$\mathbf{v}_{k+1} = \alpha \mathbf{v}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\widetilde{\boldsymbol{\theta}}_{k+1}, \mathbf{x}^i, \mathbf{y}^i)$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$$

Can we relax the dependence on the hyperparameters?

AdaGrad update

- Adapt the learning rate of all model parameters

$$\mathbf{g} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{k+1}, \mathbf{x}^i, \mathbf{y}^i)$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \boxed{\mathbf{g} \odot \mathbf{g}}$$

Element-wise multiplication

Diagonal matrix with entries that are the square of the gradient

Duchi 2011

AdaGrad update

- Adapt the learning rate of all model parameters

$$\mathbf{g} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{k+1}, \mathbf{x}^i, \mathbf{y}^i)$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \mathbf{g} \odot \mathbf{g}$$

Accumulating gradients

AdaGrad update

- Adapt the learning rate of all model parameters

$$\mathbf{g} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \mathbf{g} \odot \mathbf{g}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\epsilon}{\delta + \sqrt{\mathbf{r}_{k+1}}} \odot \mathbf{g}$$

Learning rate (ϵ)

Small constant for numerical stability (δ)

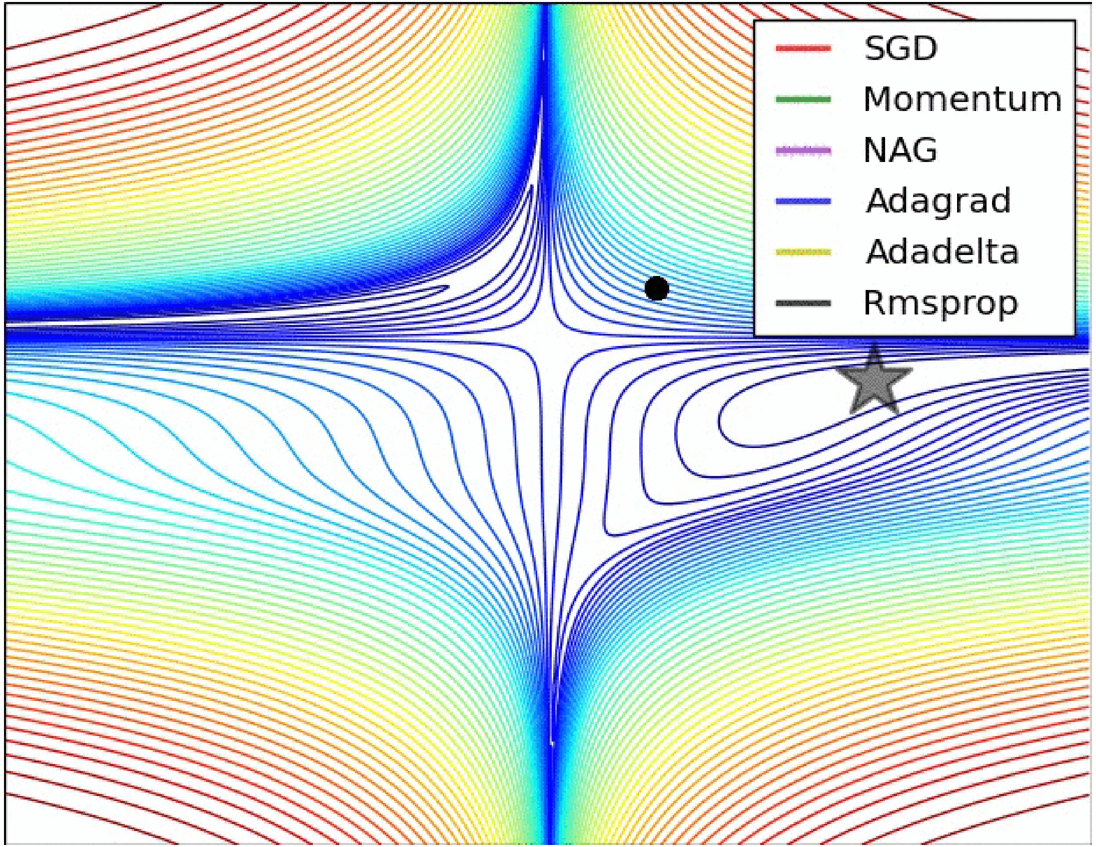
AdaGrad update

- Theory: more progress in regions where the function is more flat

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\epsilon}{\delta + \sqrt{\mathbf{r}_{k+1}}} \odot \mathbf{g}$$

- Practice: for most deep learning models, accumulating gradients from the beginning results in excessive decrease in the effective learning rate

Convergence

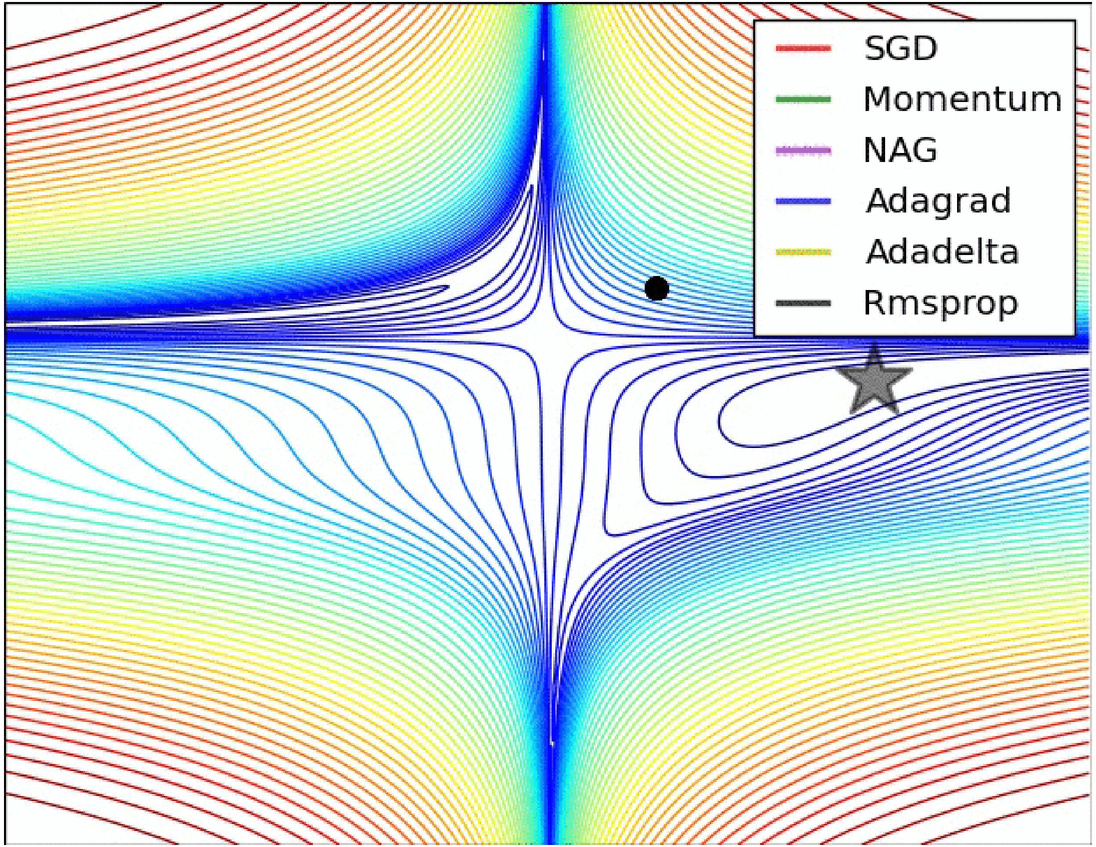


RMSProp and Adadelta

- Improvements to AdaGrad to avoid the problem of diminishing learning rate
- Decaying factor applied to the accumulation of gradients
- Old gradients are slowly forgotten

Zeiler, 2012. Hinton, 2012

Convergence



Adam

- Optimizer of choice for most neural networks
- Adam = adaptive moments
- It can be seen as an RMSProp with momentum

Kingma and Ba 2014

AdaGrad

$$\mathbf{g} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \mathbf{g} \odot \mathbf{g}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\epsilon}{\delta + \sqrt{\mathbf{r}_{k+1}}} \odot \mathbf{g}$$

Adam

$$\mathbf{g} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$$

Second order moment

$$\mathbf{r}_{k+1} = \rho_2 \mathbf{r}_k + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \frac{\hat{\mathbf{s}}}{\delta + \sqrt{\hat{\mathbf{r}}_{k+1}}} \odot \mathbf{g}$$

Adam

We can consider it as momentum

Gradient $\mathbf{g} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$

First order moment $\mathbf{s}_{k+1} = \rho_1 \mathbf{s}_k + (1 - \rho_1)\mathbf{g}$

Second order moment $\mathbf{r}_{k+1} = \rho_2 \mathbf{r}_k + (1 - \rho_2)\mathbf{g} \odot \mathbf{g}$

Unbias the moments $\hat{\mathbf{s}}_{k+1} = \dfrac{\mathbf{s}_{k+1}}{1 - \rho_1}$ $\hat{\mathbf{r}}_{k+1} = \dfrac{\mathbf{r}_{k+1}}{1 - \rho_2}$

Update step $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \dfrac{\hat{\mathbf{s}}}{\delta + \sqrt{\hat{\mathbf{r}}_{k+1}}} \odot \mathbf{g}$

Adam

Unbias the moments $$\hat{\mathbf{s}}_{k+1} = \frac{\mathbf{s}_{k+1}}{1-\rho_1} \qquad \hat{\mathbf{r}}_{k+1} = \frac{\mathbf{r}_{k+1}}{1-\rho_2}$$

- Both moments are initialized to zero, which means that specially at the beginning they have a tendency to converge to zero

$$\rho_1 = 0.9 \qquad \rho_2 = 0.999$$

Go-to optimizer

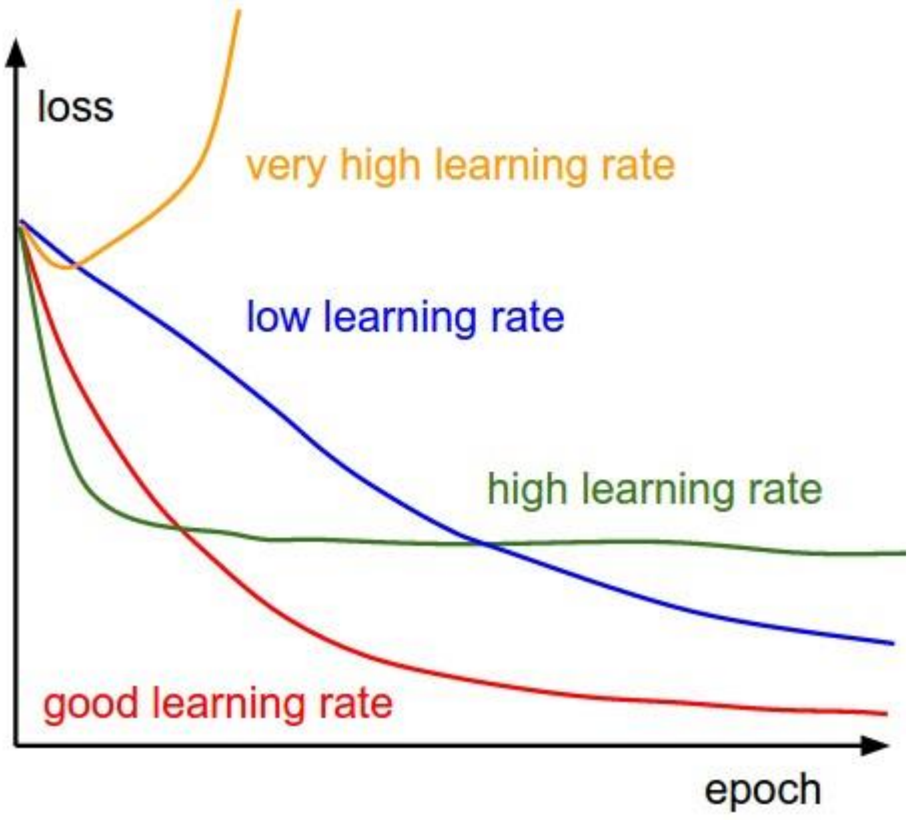
So far

- Classic optimizers: SGM, Momentum, Nesterov's momentum

- Adaptive learning rates: AdaGrad, Adadelta, RMSProp and Adam

Can we get rid of the learning rate?

Importance of the learning rate



Jacobian and Hessian

- Derivative $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}$ $\frac{df(x)}{dx}$
- Gradient $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}$ $\nabla_{\mathbf{x}} f(\mathbf{x})$ $\left(\frac{df(x)}{dx_1}, \frac{df(x)}{dx_2}\right)$
- Jacobian $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ $\mathbf{J} \in \mathbb{R}^{n \times m}$
- Hessian $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}$ $\mathbf{H} \in \mathbb{R}^{m \times m}$

SECOND DERIVATIVE

Newton's method

- Approximate our function by a second-order Taylor series expansion

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

First derivative

Second derivative (curvature)

https://en.wikipedia.org/wiki/Taylor_series

Newton's method

- SGD (green)
- Newton's method exploits the curvature to take a more direct route

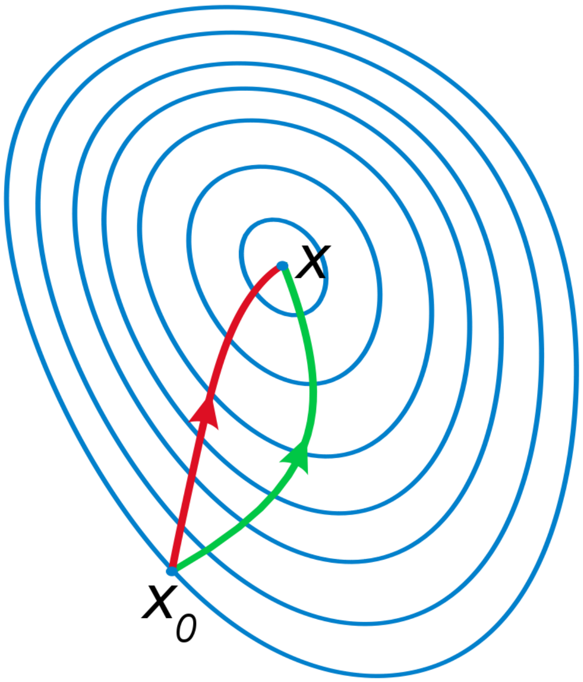


Image from Wikipedia

Newton's method

- Differentiate and equate to zero

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \mathbf{H}^{-1}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Update step

We got rid of the learning rate!

SGD $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k, \mathbf{x}^i, \mathbf{y}^i)$

Newton's method

- Differentiate and equate to zero

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 \boxed{- \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})}$$

Update step

Parameters of a network (millions)	Number of elements in the Hessian	Computational complexity of inversion per iteration
k	k^2	$\mathcal{O}(k^3)$

Only small networks can be trained with this method

Newton's method

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

Can you apply Newton's method for linear regression? What do you get as a result?

BFGS and L-BFGS

- Broyden-Fletcher-Goldfarb-Shanno algorithm
- Belongs to the family of quasi-Newton methods
- Have an approximation of the inverse of the Hessian

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boxed{\mathbf{H}^{-1}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

- BFGS $\mathcal{O}(n^2)$
- Limited memory: L-BFGS $\mathcal{O}(n)$

Which, what and when?

- Standard: Adam
- Fall-back option: SGD with momentum
- L-BFGS if you can do full batch updates (forget applying it to minibatches!!)

Backprop

The importance of gradients

- All optimization schemes are based on computing gradients

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

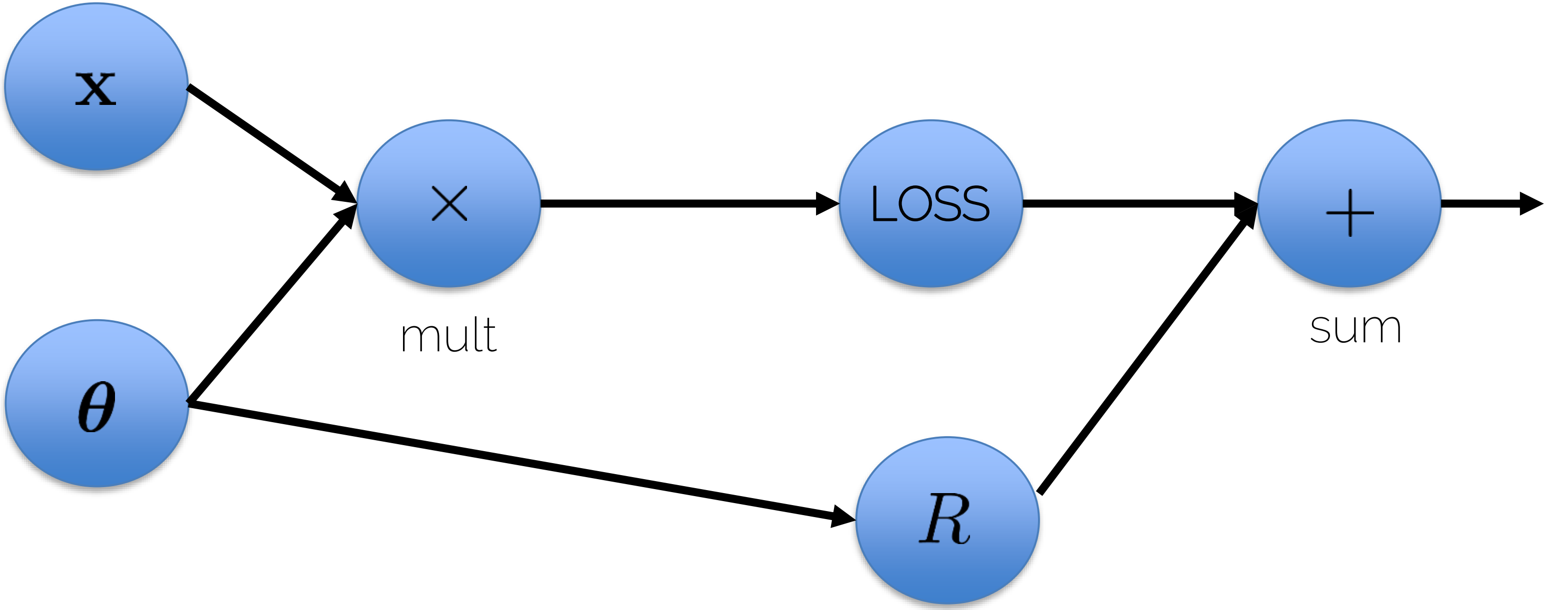
- We have seen how to compute gradients analytically but what if our function is too complex?

- Break down gradient computation **Backpropagation**

Rumelhart 1986

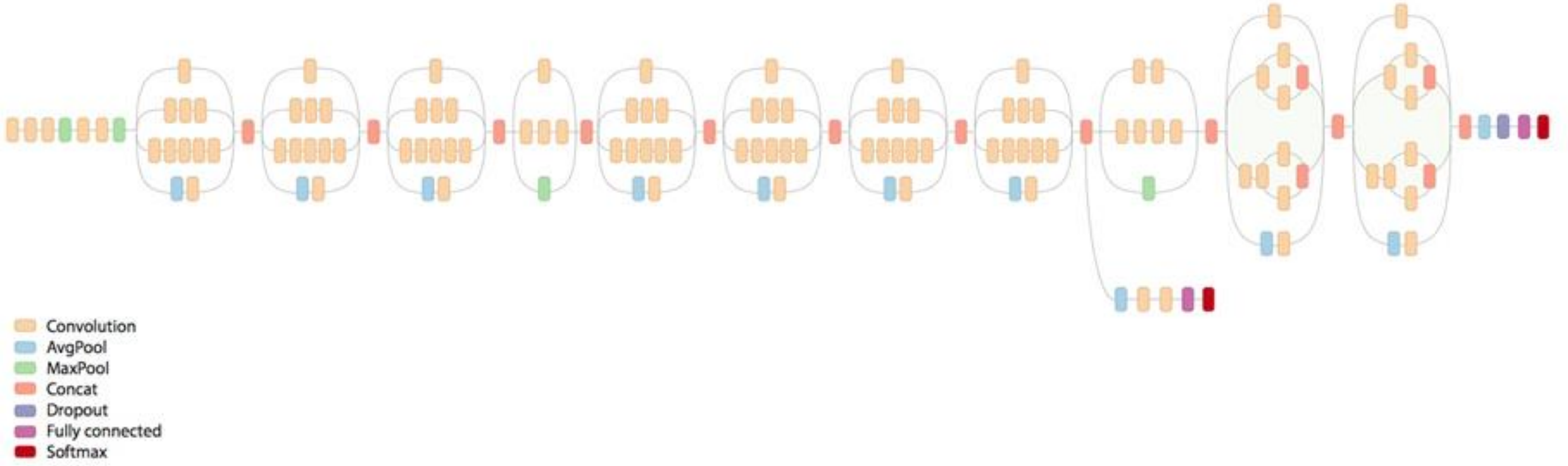
Computational graphs

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$



Computational graphs

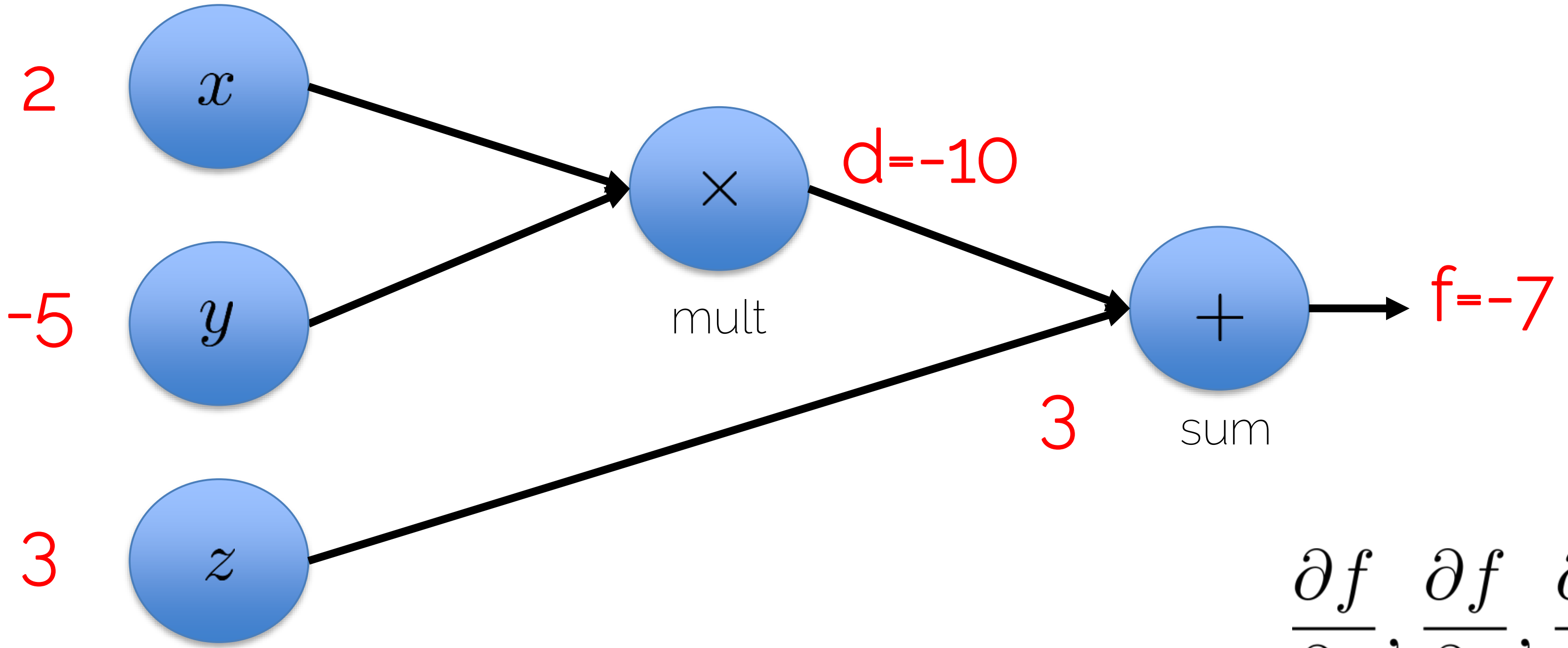
- These graphs can be huge!



Another view of GoogLeNet's architecture.

An example: forward pass

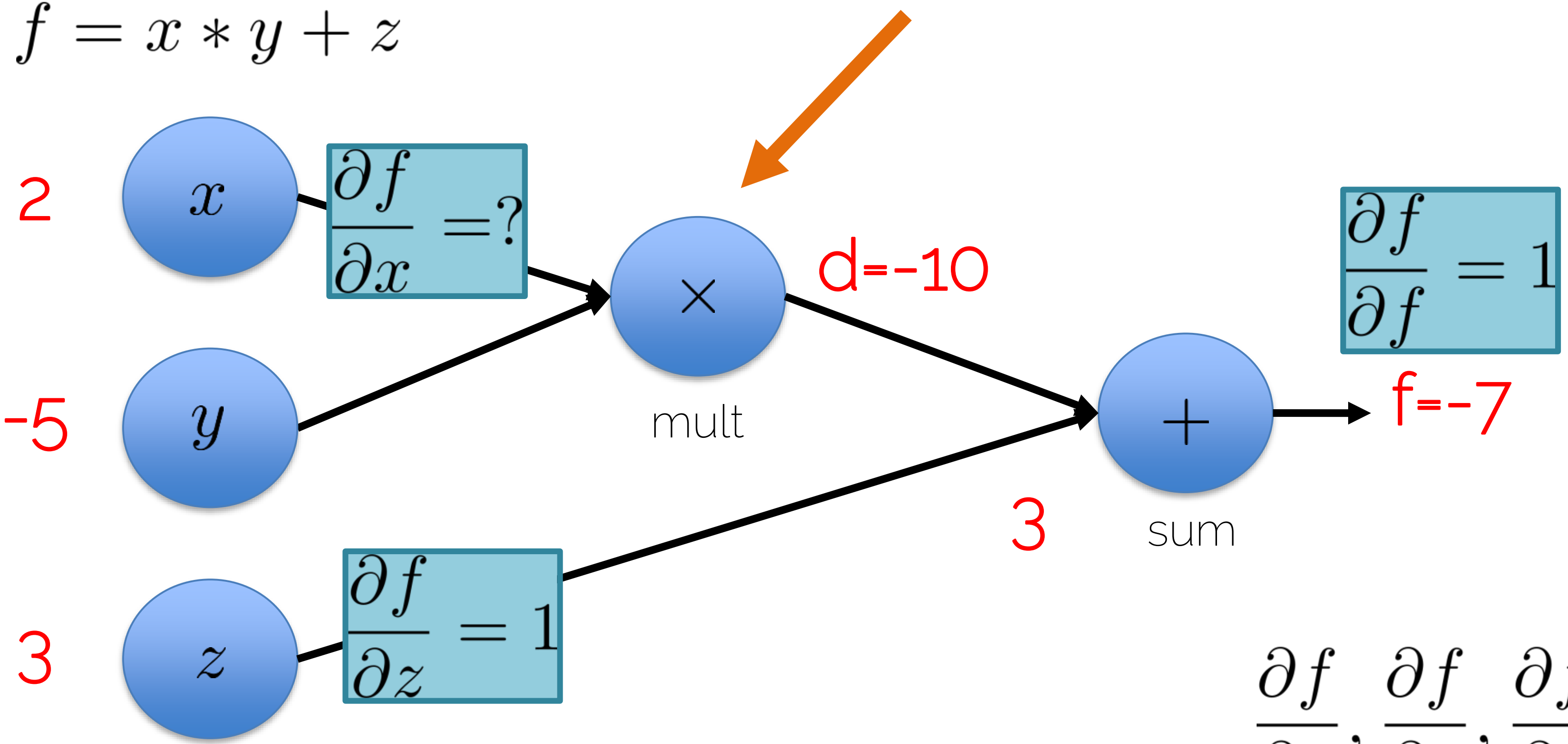
$f = x * y + z$ Initialization $x = 2, y = -5, z = 3$



$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

An example: backward pass

$$f = x * y + z$$

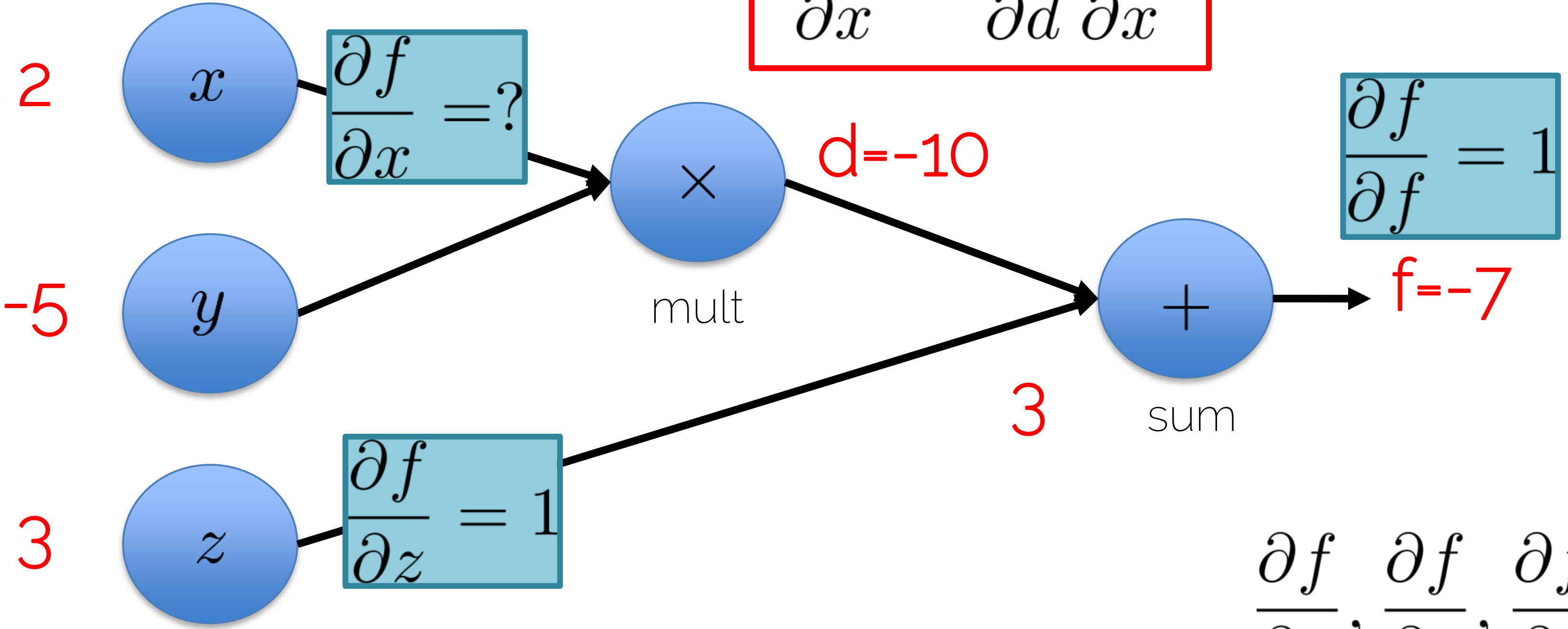


$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

An example: chain rule

$$f = x * y + z$$

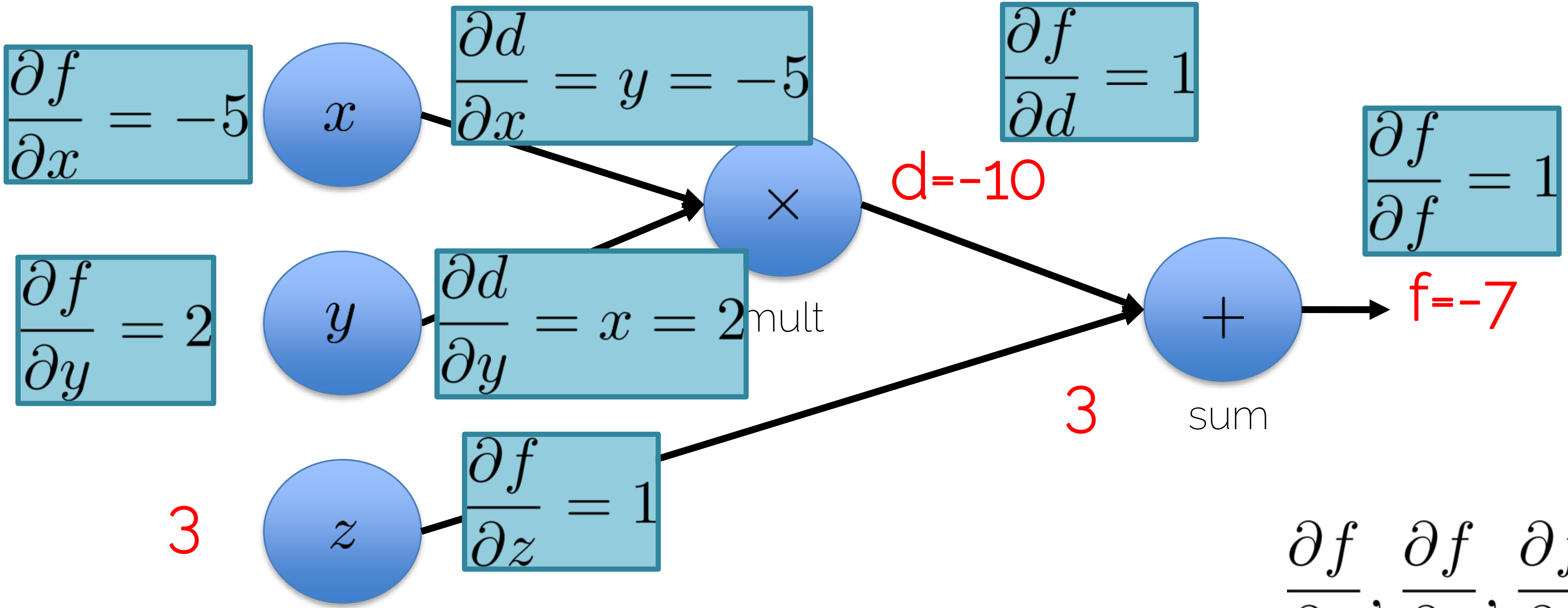
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial x}$$



$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

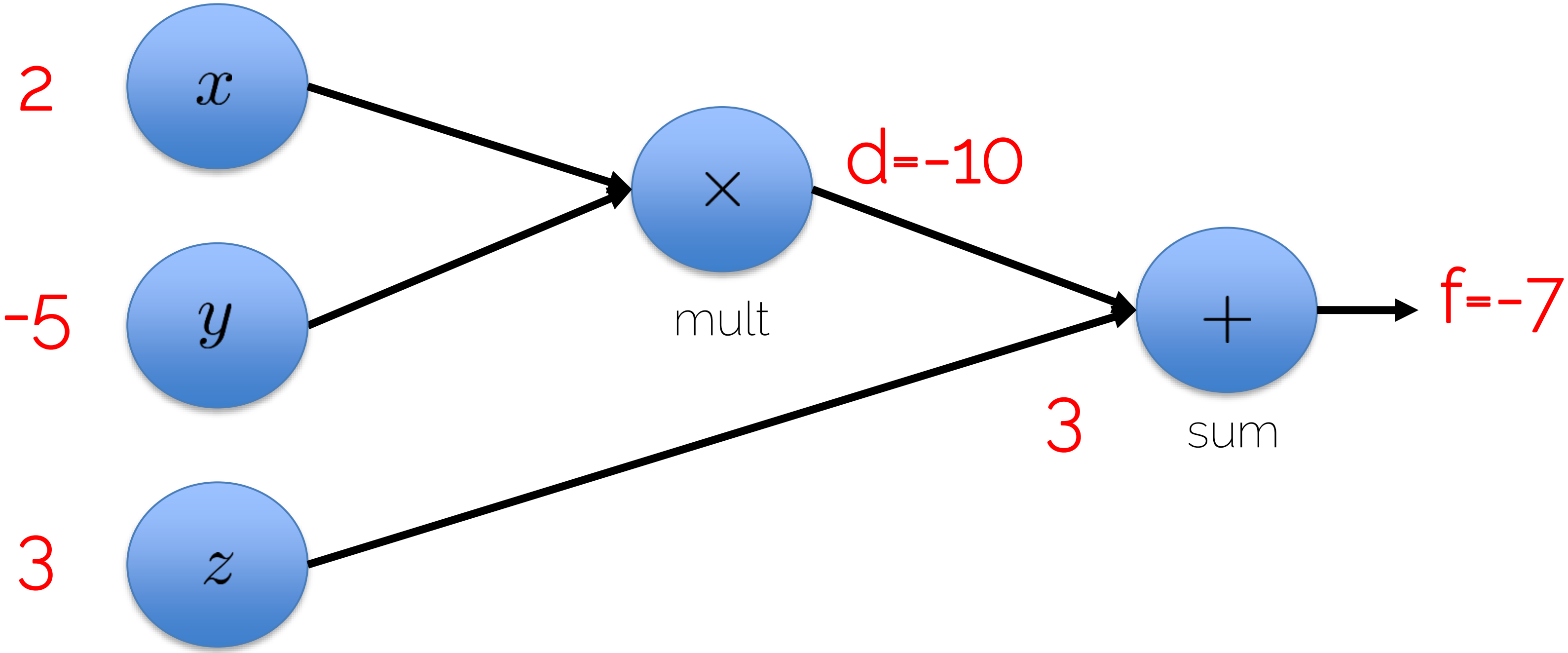
An example: chain rule $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial x}$

$$f = x * y + z$$



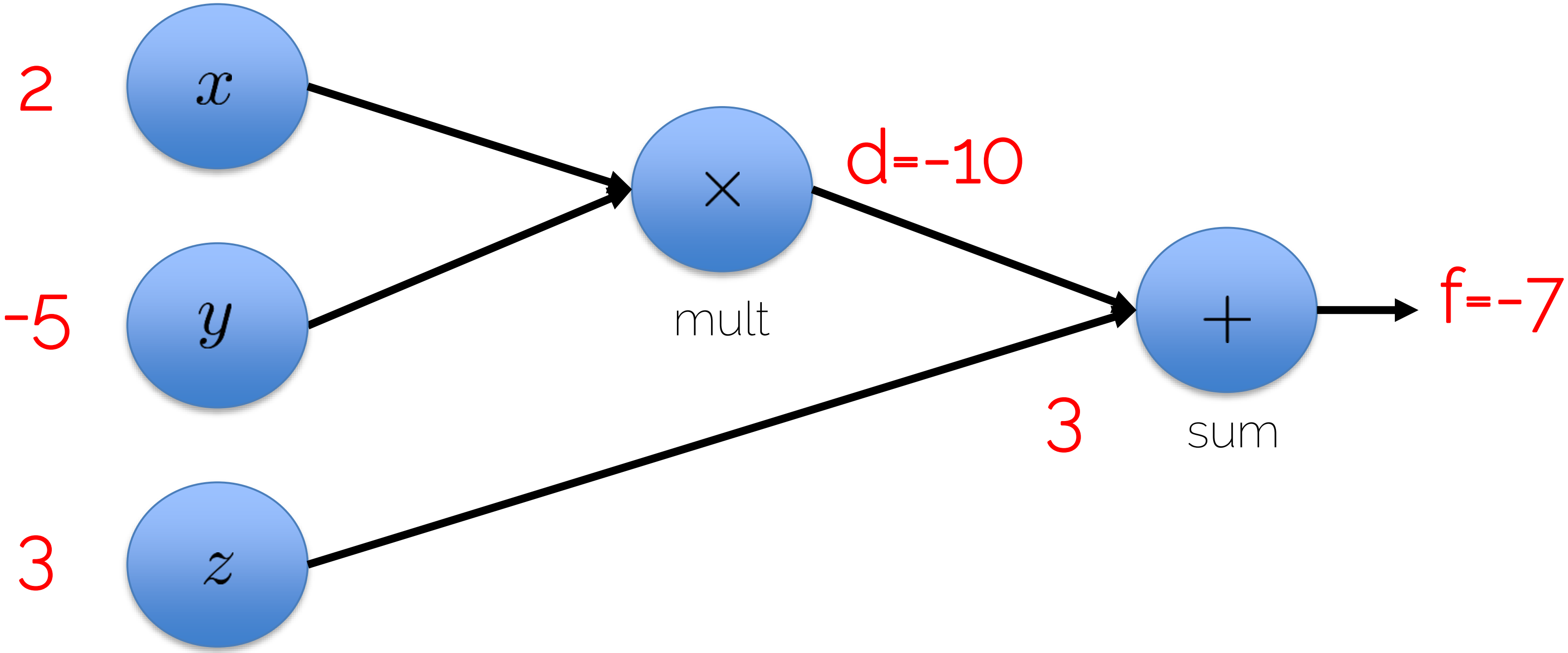
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

An example: the chain rule



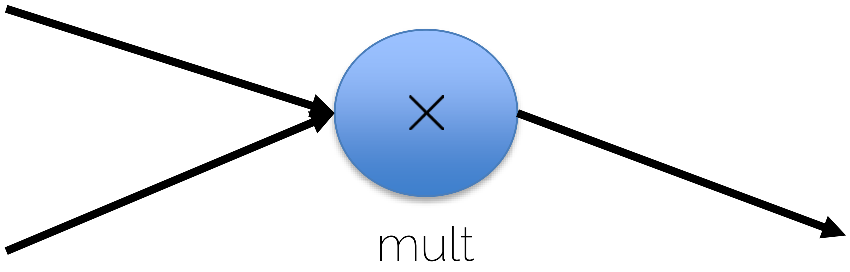
An example: the chain rule

- Each node is only interested in its own inputs and outputs

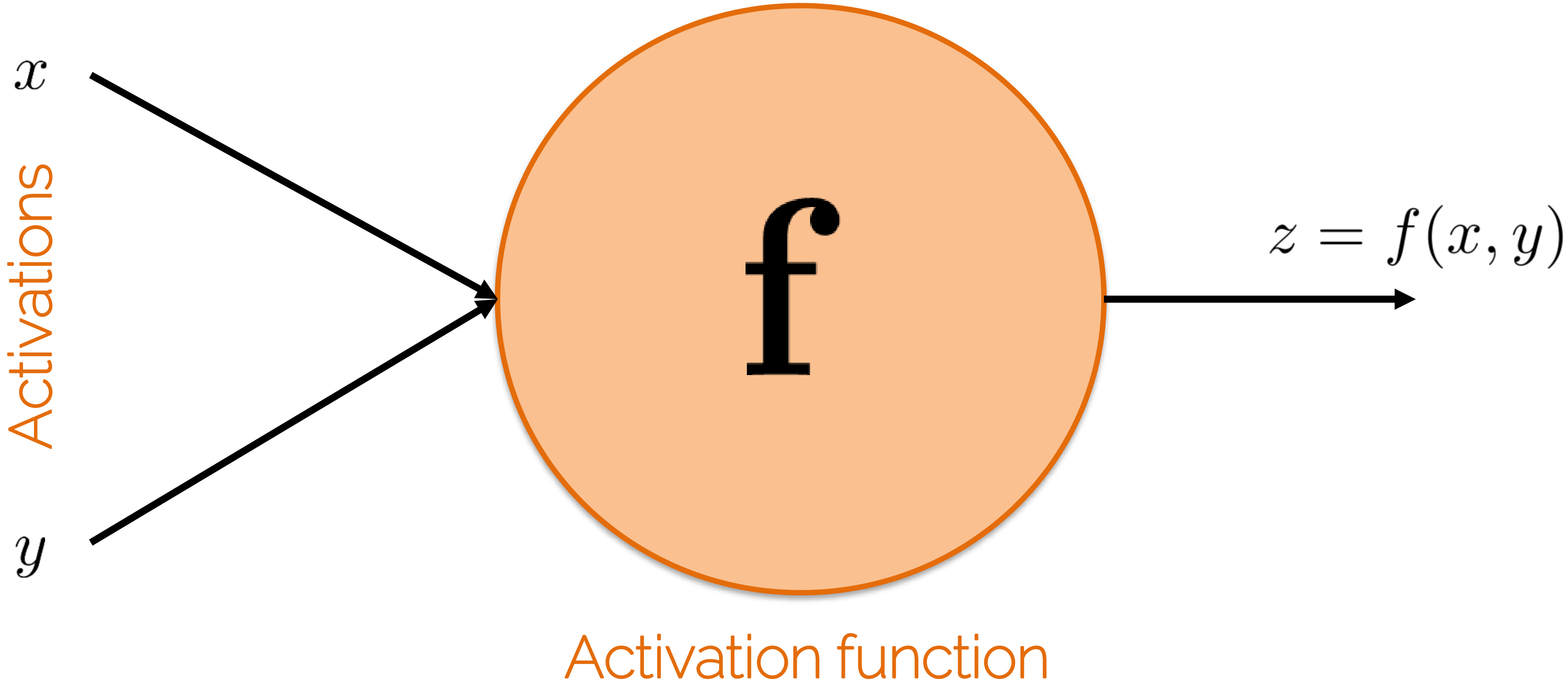


An example: the chain rule

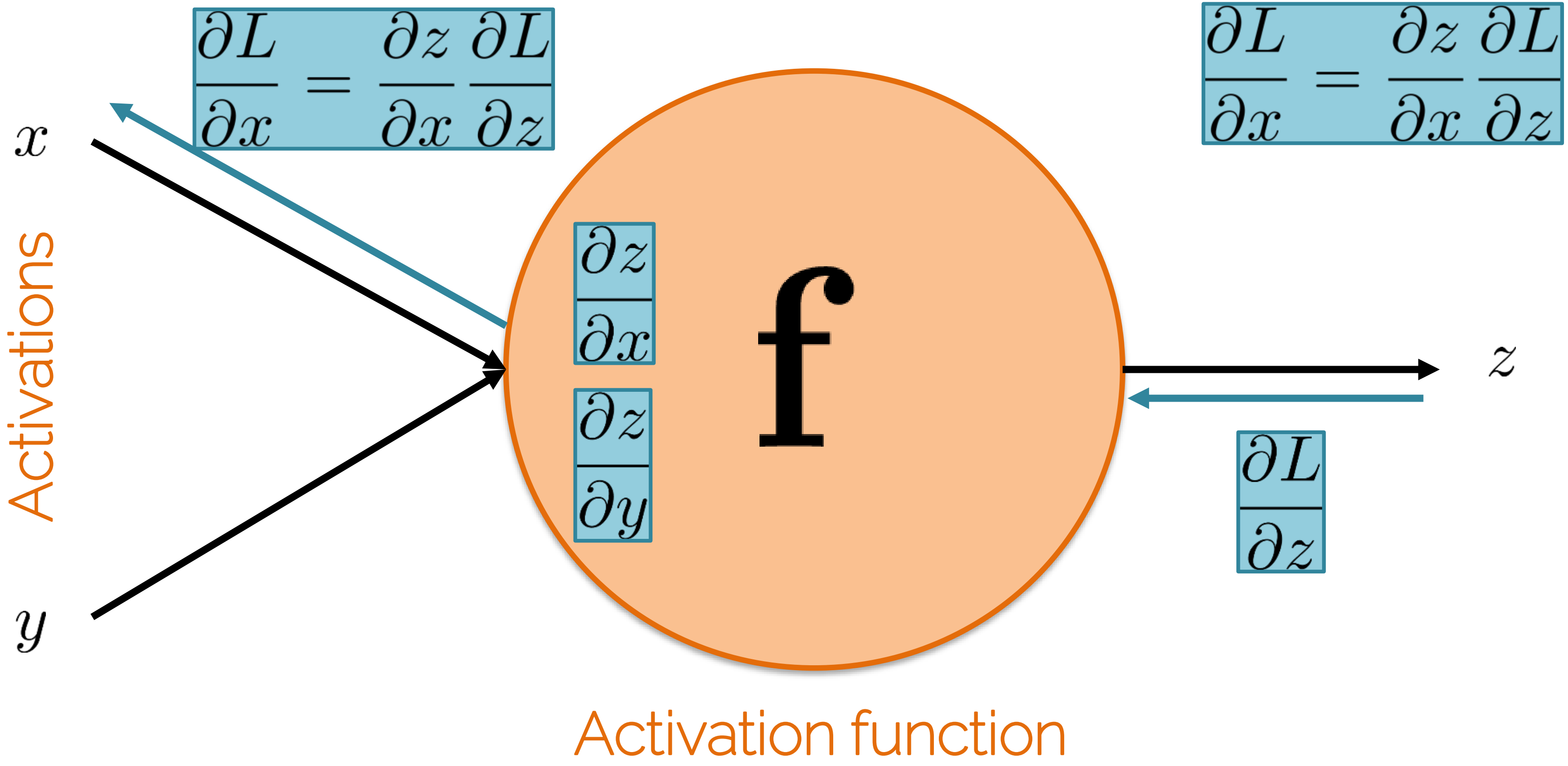
- Each node is only interested in its own inputs and outputs



The flow of the gradients



The flow of the gradients



The flow of the gradients

- Many many many many of these nodes form a neural network

NEURONS

- Each one has its own work to do

FORWARD AND BACKWARD PASS

Next lecture

- First exercise starts tomorrow!
- Next Thursday 11th of May: more on backprop, introduction to neural networks!