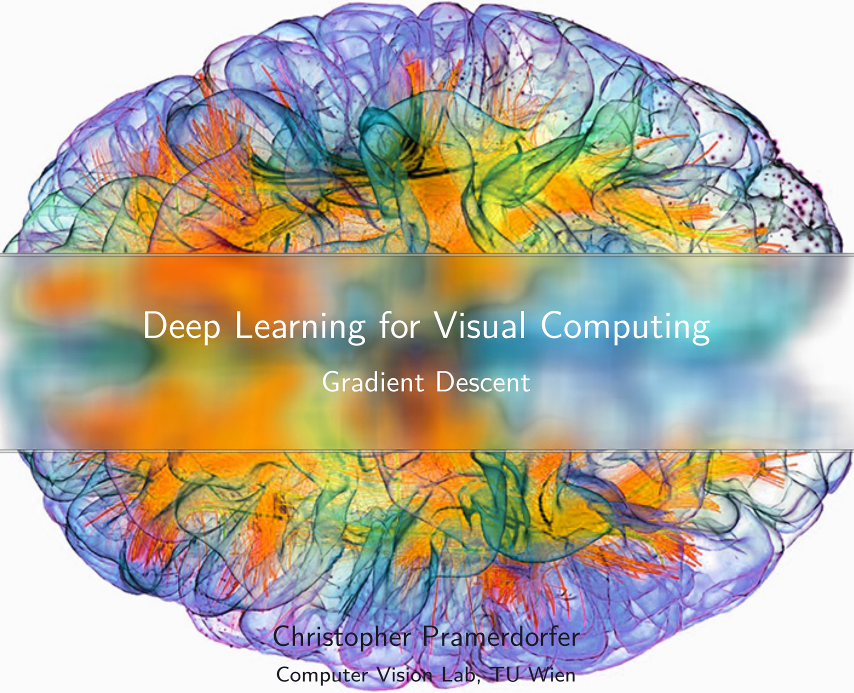

Deep Learning for Visual Computing

Gradient Descent

Christopher Pramerdorfer
Computer Vision Lab, TU Wien

Image by Neuroscape Lab

Topics

Optimization

- Gradient descent
- Momentum

Optimization

For training any parametric model, we need

- A loss function
- An optimization algorithm

We'll use cross-entropy loss (previous lecture)

Optimization

Recall that cross-entropy loss $L(\boldsymbol{\theta})$

- Measures performance of classifier $\mathbf{w} = f(\mathbf{x}\,;\boldsymbol{\theta})$
- On some dataset $\mathcal{D} = \{(\mathbf{x}_s, \mathbf{w}_s)\}_{s=1}^{S}$
- With respect to parameters $\boldsymbol{\theta}$

Optimization

$L(\boldsymbol{\theta})$ measures how dissimilar softmax($\mathbf{w}$) and $\mathbf{w}_s$ are

- softmax($\mathbf{w}$) and $\mathbf{w}_s$ are discrete probability distributions

We want to minimize loss (dissimilarity) by changing $\boldsymbol{\theta}$

- For this we need an optimization algorithm

Optimization

$L(\boldsymbol{\theta})$ is not linear in $\boldsymbol{\theta}$

- Need a nonlinear optimization algorithm

We'll use gradient descent (steepest descent)

- Most popular algorithm in Deep Learning

Assume terrain corresponds to $L(\boldsymbol{\theta})$ with $\dim(\boldsymbol{\theta}) = 2$

Optimization

Problem Definition

How do I get from location $\boldsymbol{\theta}$ to location of minimum $\hat{\boldsymbol{\theta}}$?

Without actually seeing $L(\boldsymbol{\theta})$?

Feel slope with feet, step in direction that feels steepest

Optimization
Gradient Descent

Again and again, until I cannot get lower

Gradient Descent

Iterative algorithm

In every iteration we

- Compute gradient $\boldsymbol{\theta}' = \nabla L(\boldsymbol{\theta})$
- Update parameters $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\boldsymbol{\theta}'$

Hyperparameter $\alpha > 0$ is called learning rate

- Final step size is $\alpha \left\|\boldsymbol{\theta}'\right\|$

Gradient Descent
Gradients

Let $f(x_1, \ldots, x_n)$ be a differentiable, real-valued function

The partial derivative f_{x_i} of f with respect to x_i

- Is also a real-valued function $f_{x_i}(x_1, \ldots, x_n)$

$f_{x_i}(\mathbf{x})$ encodes

- How fast f changes with argument x_i
- At some location $\mathbf{x}$

Gradient ∇f is vector of all partial derivatives of f

- $\nabla f = (f_{x_1}, \ldots, f_{x_n})$
- Vector-valued function $\mathbb{R}^n \mapsto \mathbb{R}^n$

$\nabla f(\mathbf{x}) = (f_{x_1}(\mathbf{x}), \ldots, f_{x_n}(\mathbf{x}))$ encodes

- How fast f changes with all arguments $x_1 \cdots x_n$
- At some location $\mathbf{x}$

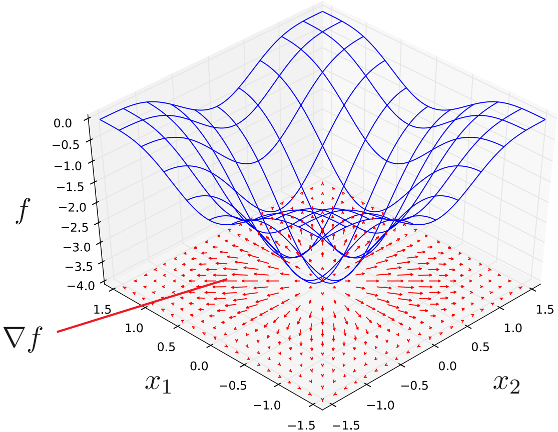

Image adapted from wikimedia.org

$\nabla f(\mathbf{x})$ specifies how f changes locally at $\mathbf{x}$

- Points in direction of greatest increase
- Norm equals magnitude of increase

Exactly what we need to minimize L

- Compute direction of greatest increase $\nabla L(\boldsymbol{\theta})$
- Move in the opposite direction

We stop if $\nabla L(\boldsymbol{\theta}) = \mathbf{0}$ (if norm is 0)

- No information where to go next
- L is flat at current location
- The case if we are at $\hat{\boldsymbol{\theta}}$ (but not only then)

Gradient Descent
Limitations

Simple and general algorithm

- Requires only that f is differentiable, real-valued

Several (possible) limitations

- Performs poorly for many f
- But generally well for loss functions of neural networks

Algorithm stops if $\nabla L(\boldsymbol{\theta}) = \mathbf{0}$

- Applies to all critical points, not only minimum
- Should stop only at minimum

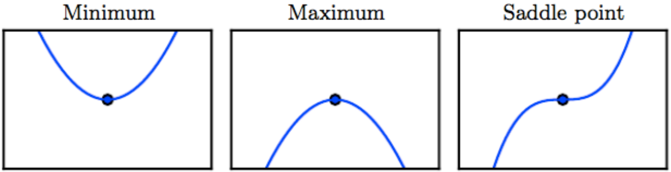

Image from [1]

Could look at second derivatives of L (Hessian)

- Describes curvature of L
- Second-order optimization methods do this

For loss functions of large neural networks

- Estimating Hessian very expensive
- Critical points usually not problematic

Algorithm stops at first minimum as $\nabla L(\boldsymbol{\theta}) = \mathbf{0}$

- But L generally has several local minima
- Algorithm usually finds only a local minimum

For loss functions of large neural networks

- Most local minima are close to global minimum
- So local minima usually not problematic

Gradient Descent
Limitations – Local Minima

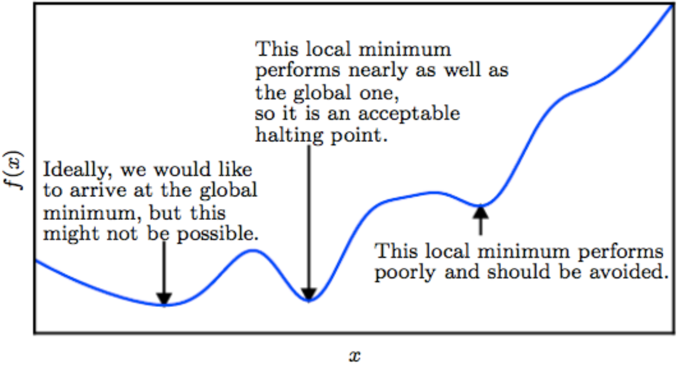

Image from [1]

Gradient Descent
Limitations – Local Minima

In practice we don't even arrive at any critical point

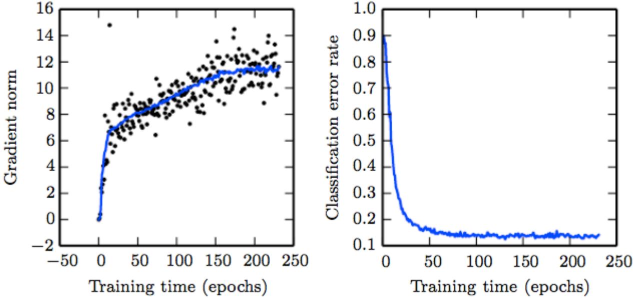

Image from [1]

Very different curvature in different directions (canyon-like)

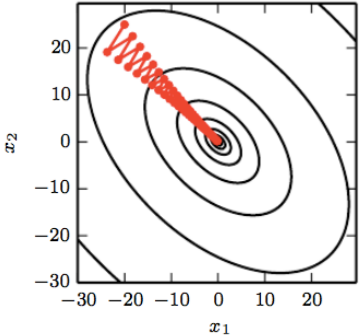

Image from [1]

Gradient descent wastes time jumping between canyon walls

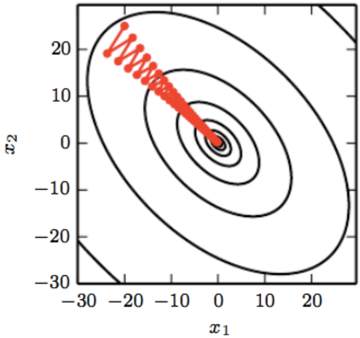

Image from [1]

Momentum improves speed of convergence by

- Dampening oscillations (previous slide)
- Increasing step size dynamically

Use exponential moving average of gradients for direction $\mathbf{v}$

- Influence of older gradients decays exponentially

Iteration of gradient descent with momentum

- Update velocity $\mathbf{v} = \beta\mathbf{v} - \alpha\nabla L(\boldsymbol{\theta})$
- Update parameters $\boldsymbol{\theta} = \boldsymbol{\theta} + \mathbf{v}$

Hyperparameter $\beta \in [0,1)$ called momentum

- Defines decay speed and maximum step size

Gradient Descent
Momentum

$\mathbf{v}$ builds up momentum if successive gradients are similar

- Improves speed of convergence

Maximum step size is $\alpha\|\mathbf{g}\|/(1-\beta)$

- Assuming the gradient is always $\mathbf{g}$
- At $\beta = 0.9$ maximum increase by factor of 10

Gradient Descent
Momentum

Red is path, black are steepest descent directions

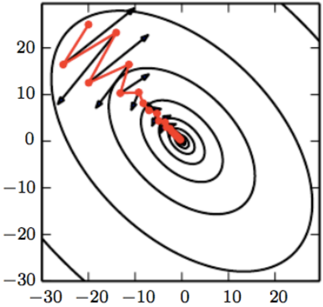

Image from [1]

Evaluate gradient at $\boldsymbol{\theta} + \mathbf{v}$ instead of $\boldsymbol{\theta}$

Iteration of gradient descent with Nesterov momentum

- Update velocity $\mathbf{v} = \beta\mathbf{v} - \alpha\nabla L(\boldsymbol{\theta} + \mathbf{v})$
- Update parameters $\boldsymbol{\theta} = \boldsymbol{\theta} + \mathbf{v}$

Often works better than standard momentum

Gradient Descent

Before we can apply gradient descent, we must know

- How to select $\mathcal{D}$ (data for loss function)
- How to initialize parameters $\boldsymbol{\theta}$ properly
- How to actually compute gradient

We'll cover this in next lecture

Bibliography

[1] *Deep learning*, 2016, [Online]. Available: http://www.deeplearningbook.org.