New learning methods

Tamás Grósz

Backpropagation

- Local optimizer
- It needs only the gradient of the loss function wrt. the parameters
- Easy to implement, widely used
- It could get stuck in local minimum
- It is sensitive to hyperparameters (learning rate)

Momemtum method

- SGD has trouble navigating ravines, which are common around local optima.
- SGD oscillates across the slopes of the ravine and makes little progress along the bottom towards the local optimum
- Momentum helps by accelerating SGD in the relevant direction and dampening oscillations

Momemtum method

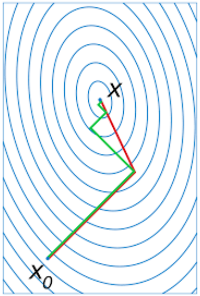
- SGD has trouble navigating ravines, which are common around local optima.
- SGD oscillates across the slopes of the ravine and makes little progress along the bottom towards the local optimum
- Momentum helps by accelerating SGD in the relevant direction and dampening oscillations

The new update rule considers the past time gradients to calculate the current update vector:

$$M_t = \alpha * \Delta W_t + (1 - \alpha) * M_{t-1}$$
$$W_{t+1} = W_t + \epsilon M_t$$

Conjugate gradient

- Much more complicated than SGD
- Optimizes the quadratic form og the Loss function
- Line search in the direction of the gradient
- Takes ortogonal steps



Adagrad

Adaptive Gradient method

Adagrad is an algorithm for gradient-based optimization that does just adapts the learning rate, performing smaller updates for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features.

$$W_{t+1} = W_t + \frac{\alpha}{\sqrt{G_t + \epsilon}} * \Delta W_t$$

where G_t accumulates the past (squared) gradient values.

Adagrad

Adaptive Gradient method

Adagrad is an algorithm for gradient-based optimization that does just adapts the learning rate, performing smaller updates for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features.

$$W_{t+1} = W_t + \frac{\alpha}{\sqrt{G_t + \epsilon}} * \Delta W_t$$

where G_t accumulates the past (squared) gradient values.

Problem

The elements of G could become large!

Adadelta (RMSProp)

- Iz is an extension of Adagrad
- To avoid large G_t values, it uses the momentum of the gradients.

$$W_{t+1} = W_t + \frac{RMS(\Delta W)_{t-1}}{RMS(G)_t} * G_t$$

where RMS is the root mean square function.

Adam

- Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter
- It stores an exponentially decaying average of past squared gradients like Adadelta and RMSprop
- Adam also keeps an exponentially decaying average of past gradients similar to momentum
- Basicly it uses the first and second order momentum of the gradients
- This method prefers flat minima in the error surface

Other optimization methods

Newton method

- Second order optimizer
- We use the second derivate of the Loss function
- For this we need to calculate the Hessian matrix

Other optimization methods

Newton method

- Second order optimizer
- We use the second derivate of the Loss function
- For this we need to calculate the Hessian matrix

Kvazi Newton method

- Calculating the Hessian matix is problematic (time consuming)
- Let's approximate it
- L-BFGS method

Practice

Python tutorial: project_demo.ipynb