Optimization

Hans Peeter Tulmin
University of Tartu, 2014

What we will discuss

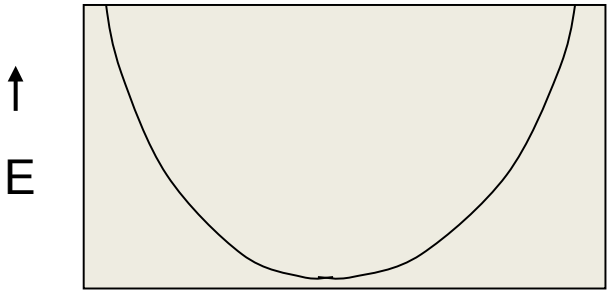
- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

What we will discuss

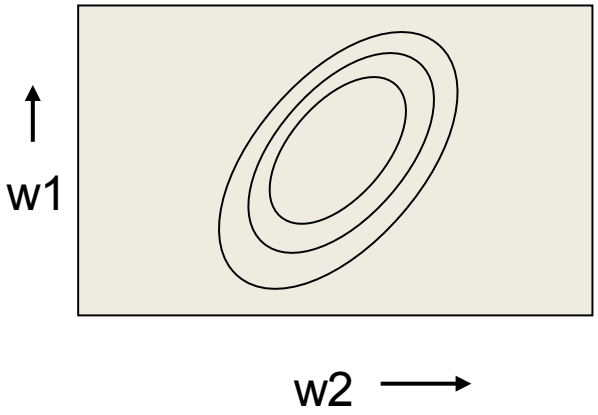
- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Reminder

- For a linear neuron with a squared error, the error surface is a quadratic bowl



- Multi layered NN-s more complicated, but locally this is a good approximation



What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

What is Gradient descent

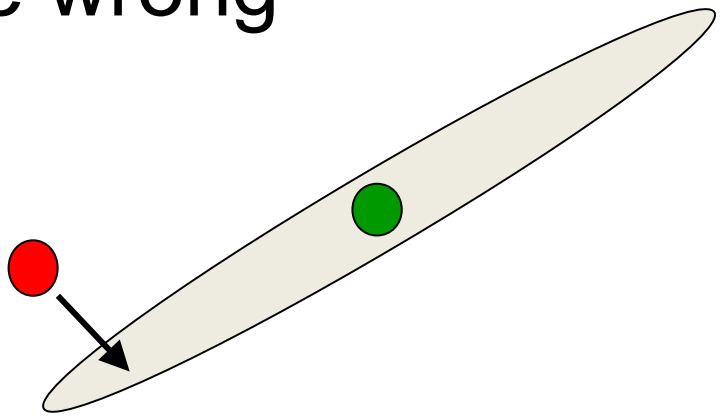
- Gradient: $\nabla f = \frac{\partial f}{\partial x_1}\mathbf{e}_1 + \cdots + \frac{\partial f}{\partial x_n}\mathbf{e}_n$

  For us, gradient means slope
- Gradient descent: $\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$
- For us, this means going down(or up) the steepest slope
- Update value by current value minus the largest slope multiplied with learning rate
- Note that the step is a sum over gradients

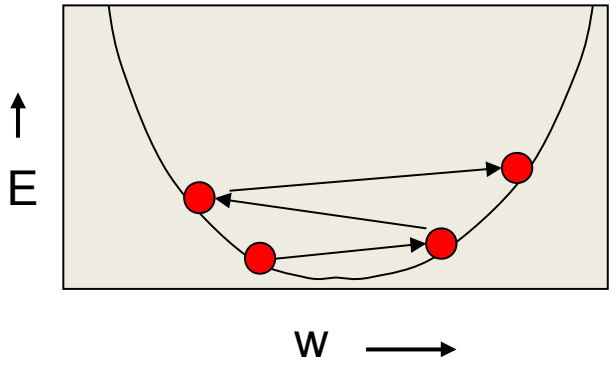
Problems with Gradient Descent

- Gradient points to minimal only in a circle

  This means learning in the wrong direction



- Too high learning rate, oscillation diverges



What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Stochastic gradient descent

- The gradient is a sum: $Q(w) = \sum_{i=1}^{n} Q_i(w),$

  What would happen if we would sample a random term and move in that direction?

Stochastic gradient descent

- The gradient is a sum: $Q(w) = \sum_{i=1}^{n} Q_i(w),$

  What would happen if we would sample a random term and move in that direction?

  Stochastic gradient descent!

- Intuitively, we take small steps until we are not moving enough.
- Also known as On-line gradient descent

What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Mini-batch

- Central path of the two methods
- Stochastic gradient descent with multiple variables
- Faster than pure stochastic gradient descent due to vectorization
- Most common way to train artificial neural networks(combined with backpropagation)

Tips for using Mini-batch

- Probe learning rate
- Turn down learning rate near the end of learning, helps with fluctuation
- Turn down learning rate when error stops decreasing
- Test error on validation set
- Decorrelate inputs, for example PCA
- Convert to circular by dividing with sqrt(eigen)

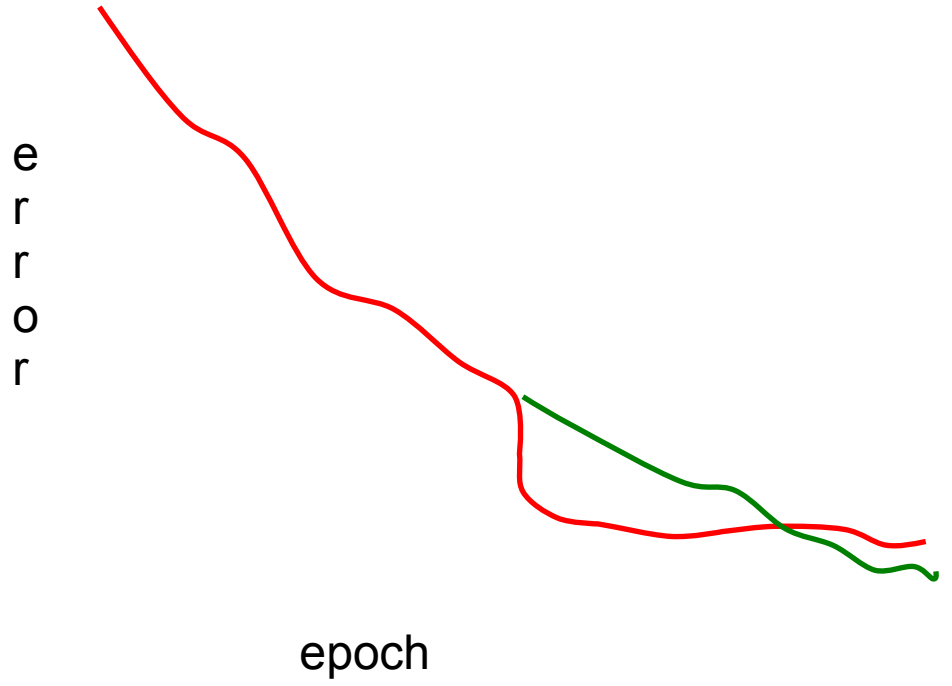
Weights and Inputs

- Initialize weights with small random change
- Initialize weights proportional to sqrt(fan-in)
- Shtift inputs so that the mean of the shift is 0
- Hidden unit as hyperbolic tangent (-1,1), activations roughly 0
- Logistic ignores fluctuations
- Also, scaling inputs can help

  PCA is better as it gurantees a circle

Turning down learning rate

- Not too soon, a quick win might not be as beneficial
- Turning down learning rate means slower learning. Not turning down might be faster



Increase speed

- Use Momentum
- Use separate learning rates
- Rmsprop
- Use fancy optimization methods

Increase speed

- Use Momentum
- Use separate learning rates
- Rmsprop
- Use fancy optimization methods

All but fancy optimization will be discussed!

Problems to watch out for

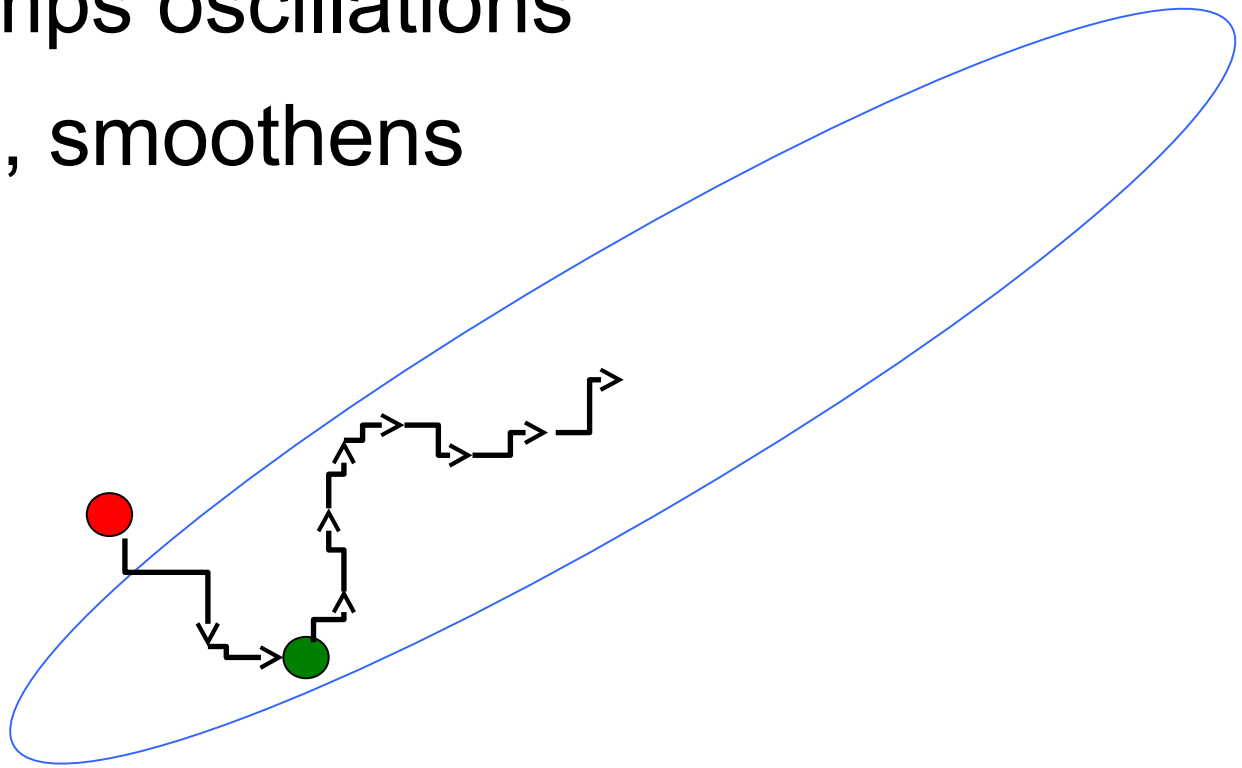
- Big learning rate blows stuff up
  - Input does not affect neurons
  - Error becomes tiny, seems like minima, is a plateau
- Classification net problems(output unit is the fraction that it is 1)
  - Strategy found quickly but improvement is very slow
  - May make us assume that it is a local minima

What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Momentum method

- Update momentum with gradient instead
- Like a ball. Starts rolling slowly, but after a while rolling in the same direction, accelerates
- This damps oscillations
- As seen, smoothens learning



Equations

- Velocity decays by a, which $\mathbf{v}(t) = \alpha\, \mathbf{v}(t-1) - \varepsilon \frac{\partial E}{\partial \mathbf{w}}(t)$

  Is less than one
- Weigth changes according to current velocity

  $\Delta \mathbf{w}(t) = \mathbf{v}(t)$
- Weight as previous weight and current gradient

$$\Delta \mathbf{w}(t) = \mathbf{v}(t) \quad = \alpha\, \mathbf{v}(t-1) - \varepsilon \frac{\partial E}{\partial \mathbf{w}}(t)$$

$$= \alpha\, \Delta \mathbf{w}(t-1) - \varepsilon \frac{\partial E}{\partial \mathbf{w}}(t)$$

Tips for using momentum

- Initialize small momentum
- Raise momentum gradually(gradient smaller)
- Faster: $\mathbf{v}(\infty) = \frac{1}{1-\alpha}\left(-\varepsilon \frac{\partial E}{\partial \mathbf{w}}\right)$

  Intuitively: update gradient, jump
- Use Nesterovs method:

  Make a jump and only then fix the position according to current gradient

What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Separate adaptive rates

- Learning rates vary between weights
  - Gradients are different
  - So is fan-in(changing can „overshoot“)

- Solution

  use global learning rate multiplied with local gain

  Learning rates are different locally

Local gains

- Initialize local gains to one
- Increase if sign not changed
- Additive increase, multiplicative decrease
  - Oscillation dies out
  - Random gradient gives local gain of one
- Limit the gains
- Can be used with momentum(sign agreement between gradient and velocity)

$$\Delta w_{ij} = -\varepsilon \, g_{ij} \, \frac{\partial E}{\partial w_{ij}}$$

$$if \quad \left(\frac{\partial E}{\partial w_{ij}}(t) \frac{\partial E}{\partial w_{ij}}(t-1) \right) > 0$$

$$then \quad g_{ij}(t) = g_{ij}(t-1) + .05$$

$$else \quad g_{ij}(t) = g_{ij}(t-1) * .95$$

What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Rprop

- Resilient backpropagation
- Full batch only
- Motivation: Gradient variance is big, hard to choose global learning rate
- Use gradient sign and depend on local learning rates
- Check last two signs, if they are the same, multiply by 1.2 otherwise by 0.5
- Give bounds, a millionth and 50 are sensible

Rprop problems

- Does not work on mini-batches
- Rprop ignores sign, thus it would consider all batches equivalent. So, when one batch changes the weight by a lot, it is not noticed

- The dream: rprop + mini-batch + effective averaging of mini-batch gradients

Rmsprop

- Mini-batch Rprop
- Hinton's favourite
- Running average of squared gradient for each weight:

$$MeanSquare(w, t) = 0.9\ MeanSquare(w,\ t-1) + 0.1 \left(\partial E / \partial w (t) \right)^2$$

- We divide the gradient by $\sqrt{MeanSquare(w,\ t)}$

Suggestions for improvement

- Use standard momentum(not as big of an increase as normally)
- Use Nesterov momentum(divide correction rather than jump)
- Adaptive learning rates
  - Hinton does not know
- Google LeCun, do what he does
  - „No more pesky learning rates“

    http://arxiv.org/pdf/1206.1106v2.pdf

What we will discuss

- Reminder from previous lectures
- Gradient Descent
- Stochastic Gradient Descent
- Mini-batch
- Momentum method
- Adaptive learning
- Rprop and Rmsprop
- Summary

Summary

- Small datasets – Use full-batch
  - Adaptive learning rates
  - Conjugate gradient
- Big redundant dataset – Mini-batch
  - Try momentum
  - Try rmsprop
- Problematic because Neural Nets are very different(structure, goals etc.)