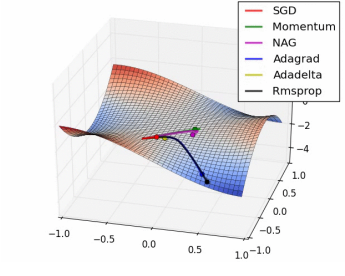
Optimization in Machine Learning

First order methods
Adam and friends

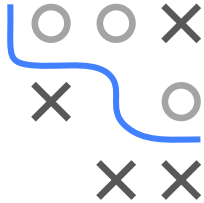


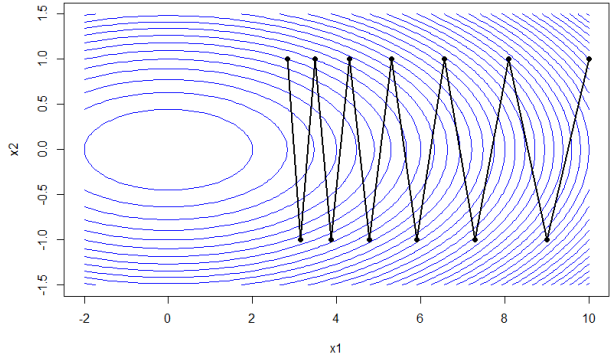
Learning goals

- Adaptive step sizes
- AdaGrad
- RMSProp
- Adam

ADAPTIVE STEP SIZES

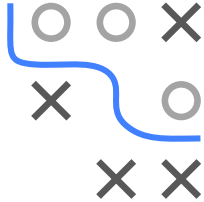
- Step size is probably the most important control parameter
- Has strong influence on performance
- Natural to use different step size for each input individually and automatically adapt them





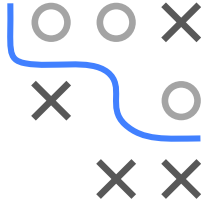
ADAGRAD

- AdaGrad adapts step sizes by scaling them inversely proportional to square root of the sum of the past squared derivatives
  - Inputs with large derivatives get smaller step sizes
  - Inputs with small derivatives get larger step sizes
- Accumulation of squared gradients can result in premature small step sizes (Goodfellow et al., 2016)



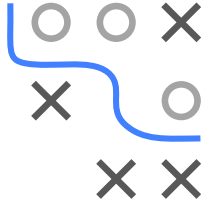
RMSPROP

- Modification of AdaGrad
- Resolves AdaGrad's radically diminishing step sizes.
- Gradient accumulation is replaced by exponentially weighted moving average
- Theoretically, leads to performance gains in non-convex scenarios
- Empirically, RMSProp is a very effective optimization algorithm. Particularly, it is employed routinely by DL practitioners.

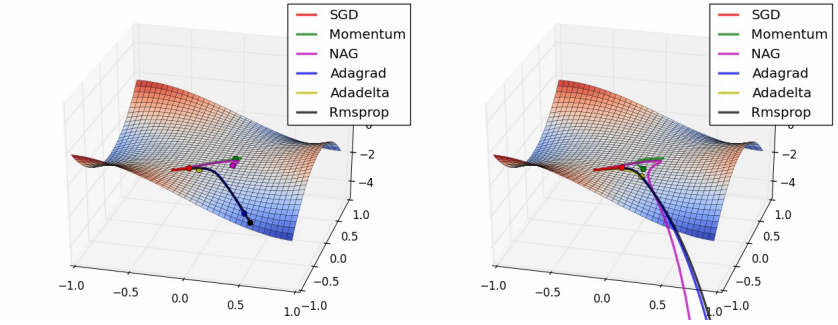


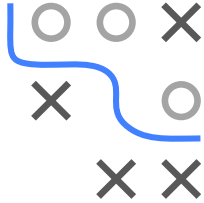
ADAM

- Adaptive Moment Estimation also has adaptive step sizes
- Uses the 1st and 2nd moments of gradients
  - Keeps an exponentially decaying average of past gradients (1st moment)
  - Like RMSProp, stores an exp-decaying avg of past squared gradients (2nd moment)
  - Can be seen as combo of RMSProp + momentum.



COMPARISON OF OPTIMIZERS: ANIMATION





Credits: Dettmers (2015) and Radford

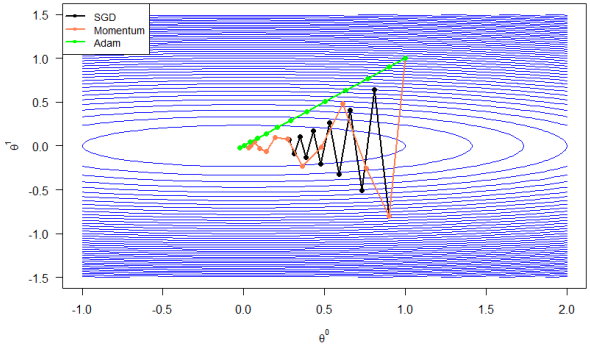
Comparison of SGD optimizers near saddle point.

Left: After start. **Right:** Later.

All methods accelerate compared to vanilla SGD.

Best is RMSProp, then AdaGrad. (Adam is missing here.)

COMPARISON ON QUADRATIC FORM



SGD vs. SGD with Momentum vs. Adam on a quadratic form.