1. Gradient descent

* The method we use to train all our models (ANNs, CNNs, RNNs, etc.)
* Backbone of deep learning
* Can be used to train other ML models
* K-mean clustering
* Hidden Markov models
* Matrix factorization
* If you’re an expert in mathematics, you may be interested in:
* ‘Gradient Descent: Convergence Analysis’ in extra\_reading.txt
* Why do we need gradient descent
* What problem we are trying to solve
* For model (e.g., linear regression), we come up with a cost/loss
* We want to minimize the cost (L) with respect to parameters (w)
* Calculus – how do we minimize a function in calculus?
* We find its derivative and set it to 0
  + E.g.:
* Works because: **Slope at the minimum / maximum of a function is 0**
* How do we know if it’s a minimum or maximum of a function?
* We built the loss function
* What about local minima or saddle points?
* Not a problem in modern deep learning
* Why can’t we just take the gradient and set it to 0, then solve for w?
* Some equations are simply not solvable
* Numerical approximations
* What do we do when we can’t solve an equation?
* Approximate it
* If you can’t solve an integral, you can simply draw little trapezoids and find the area of the trapezoids instead
* Estimation of area under the curve
* How does gradient descent work?
* Idea: repeatedly take small steps in the direction of the gradient to find a new w
* At each step, L(w) decreases provided the step size is small enough
* Eventually, it will converge to the minimum
* Gradient descent in code

A close-up of a black text

Description automatically generated

* Example
* Minimize
* Gradient is
* Inside the loop:
* The solution is
* Hyperparameters
* Each iteration of the loop is called an epoch, and we must choose the number of epochs high enough so the cost converges
* is the learning rate – must be small enough so that the cost does not blow up, but large enough so that you don’t have to wait longer than necessary
* Trial and error is how you choose them
* You can also use automated methods like Bayesian optimization
* Practice is best

1. Stochastic gradient descent (SGD)

* A simpler problem
* Suppose we want to measure the average height of everyone in the world
* 7 billion people
* Too many to survey
* What if just asked 1000 people (chosen randomly)
* Do we expect the average of this 1000 to be similar to the population average? Yes
* Advantage: asking 1000 people takes less time
* How is this related to deep learning?
* Recall: we must calculate the cost and the gradient of the cost
* Cost depends on number of samples in the dataset
* What if we are using the ImageNet dataset and we have N = 1 million
* Take long time to add up errors of 1 million sample
* What if we just took the average error over 1000 images instead?
* This average error is probably close to the average error on the entire dataset
* Gradient is also similar, but 1000x less computation!
* Stochastic (batch) gradient descent
* Typically we use smaller batch sizes, like 32, 64, or 128

A screenshot of a computer code

Description automatically generated

1. Momentum

* Most performance-improving in plain SGD
* Without momentum:
* Gradient descent momentum
* Every time we want the box to move, we have to push it again
* Difficult
* Gradient descent, without momentum
* If is 0 -> parameter doesn’t change!
* Gradient descent, with momentum
* 2 steps: sliding on ice & pushing the box
* Typical values of are 0.9, 0.95, 0.99, etc.
* Without any g, the box ‘slows down’
* Effect of
* We just get back regular gradient descent
* Effect of momentum – speeds up training

A graph of a line

Description automatically generated with medium confidence

* Another perspective

A diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of

Description automatically generated

1. Variable and adaptive learning rates

* We’ve seen that momentum can greatly speed up training