CPSC 340:
Machine Learning and Data Mining

Stochastic Gradient

Bonus Slides

Mini-batches: Using more than 1 example

- Does it make sense to use more than 1 random example?
  - Yes, you can use a "mini-batch" B^t of examples.

$$w^{t+1} = w^t - \alpha^t \frac{1}{|B^t|} \sum_{i \in B^t} \nabla f_i(w^t)$$

Random "batch" of examples.

  - Radius of ball is inversely proportional to the mini-batch size.
    - If you double the batch size, you half the radius of the ball.
      - Big gains for going from 1 to 2, less big gains from going from 100 to 101.
    - You can use a bigger step size as the batch size increases ("linear scaling" rule).
      - Gets you to the ball faster (though diverges if step-size gets too big).
  - Useful for vectorizing/parallelizing code.
    - Evaluate one gradient on each core.

Polyak-Ruppert Iterate Averaging

- Another practical/theoretical trick is averaging of the iterations.
  1. Run the stochastic gradient algorithm with $\alpha^t = O(1/\sqrt{t})$ or α^t constant.
  2. Take some weighted average of the w^t values.

$$\bar{w}^t = \sum_{k=1}^{t} v^k w^k$$

Here, v^k is a scalar "weight" of iteration 'k'

Uniform average:

$$\bar{w}^t = \frac{1}{t} \sum_{k=1}^{t} w^k$$

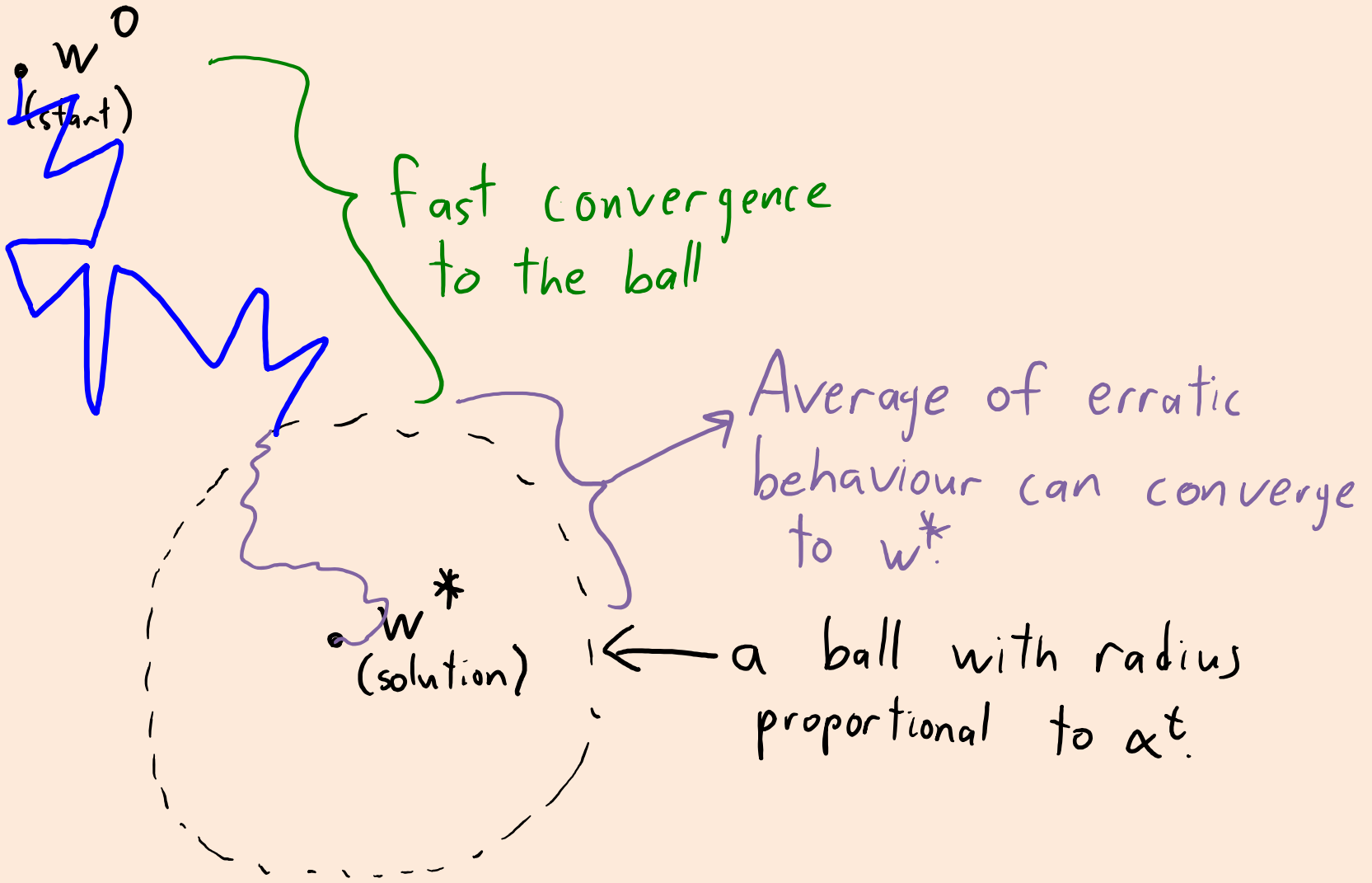
$v^k = \frac{1}{t}$

- Average does not affect the algorithm, it's just "watching".
- Surprising result shown by Polyak and by Ruppert in the 1980s:
  - Asymptotically converges as fast as stochastic Newton's method.

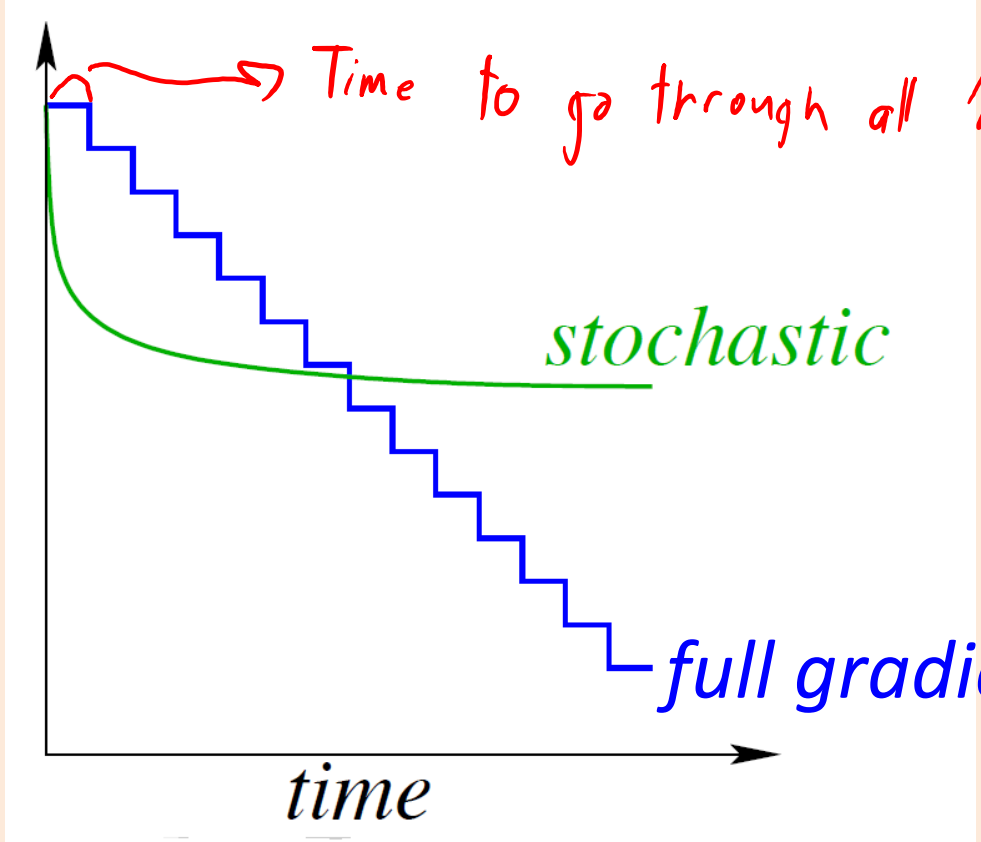
Stochastic Gradient with Averaging

Often, you average the second half of the iterations.

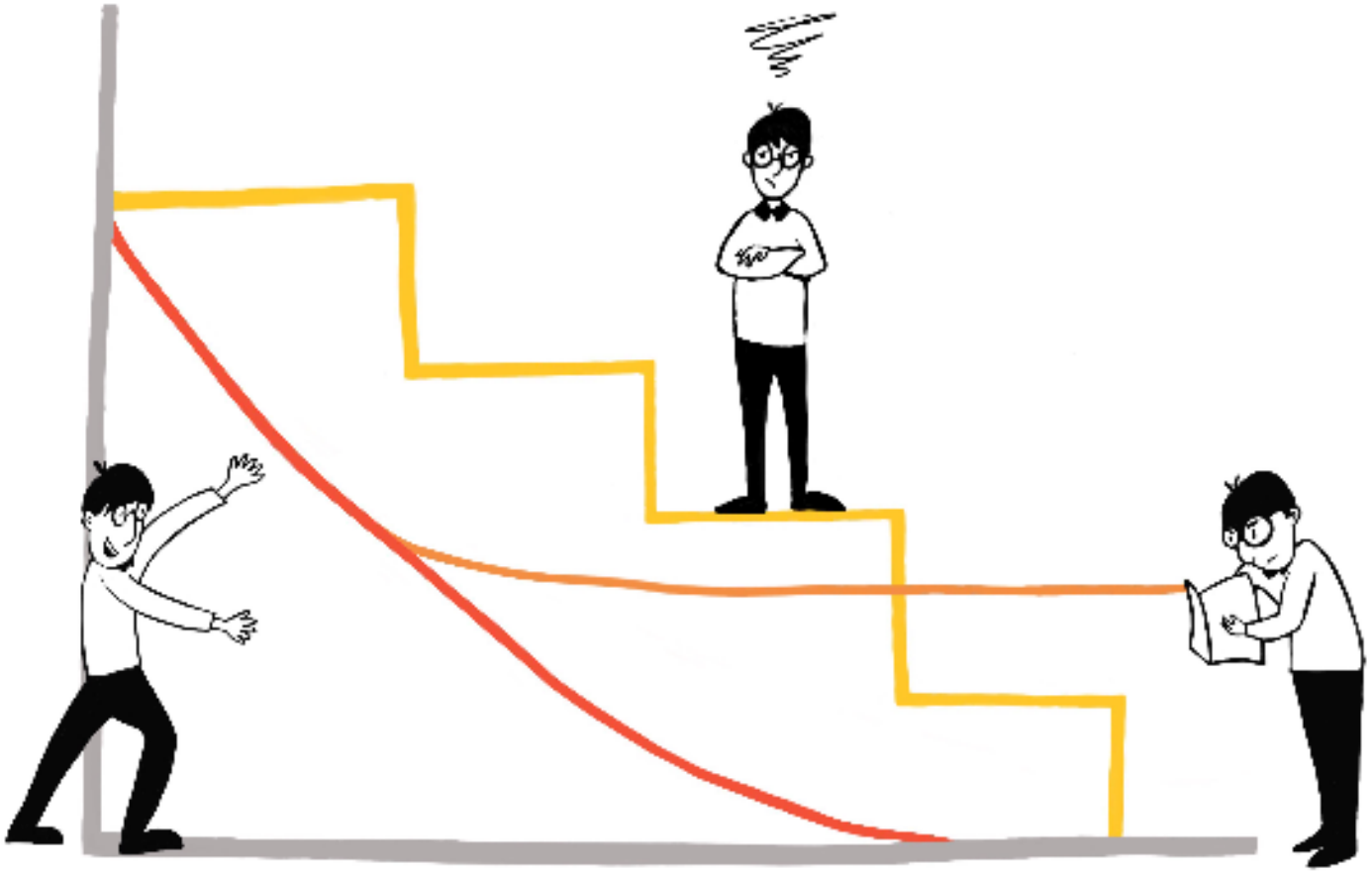
Set $\bar{w}^t = \frac{1}{t/2} \sum_{k=t/2}^{t} w^k$



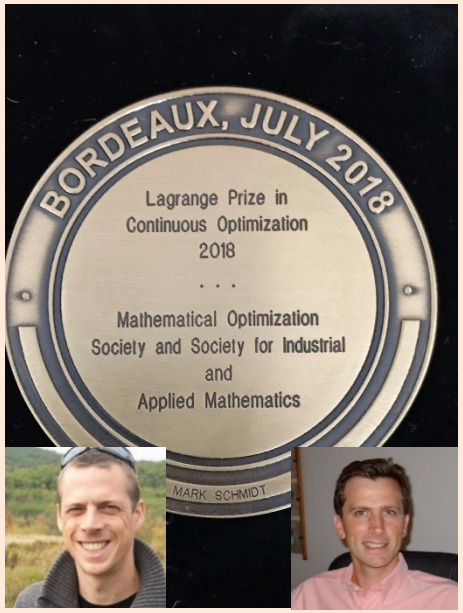
Gradient Descent vs. Stochastic Gradient



- 2012: methods with cost of stochastic gradient, progress of full gradient.
  - Key idea: if 'n' is finite, you can use a memory instead of having α_t go to zero.
  - First was stochastic average gradient (SAG), "low-memory" version is SVRG.



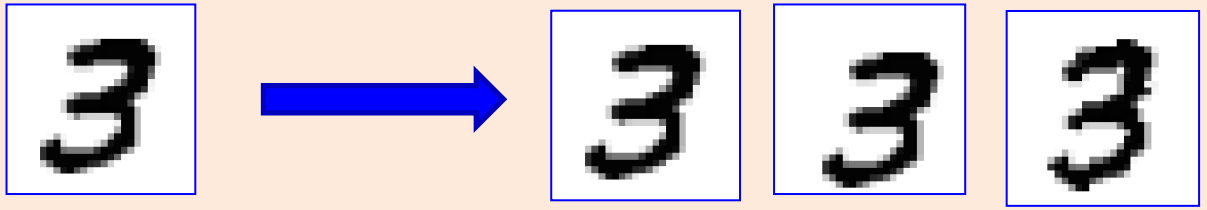
This graph shows how algorithms have become fast and more efficient over time. The horizontal axis represents time and the vertical axis represents error. Older algorithms (yellow) were very slow but had very little error. Faster algorithms were created by only analyzing some of the data (orange). The method was faster but had an accuracy limit. Schmidt's algorithm is faster and has no accuracy limit. *Aiken Lao / The Ubyssey*



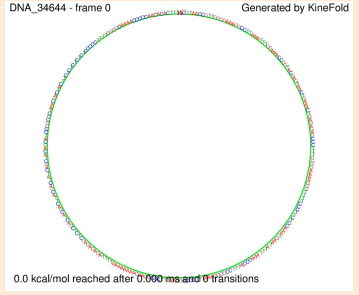
https://www.ubyssey.ca/science/schmidt-sloan-fellowship/

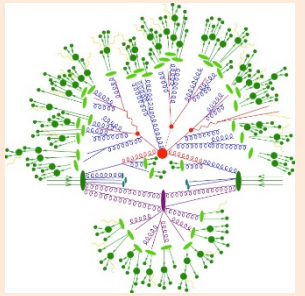
Machine Learning with “n = ∞”

- Here are some scenarios where you effectively have “n = ∞”:
  - A dataset that is so large we cannot even go through it once (Gmail).
  - A function you want to minimize that you can’t measure without noise.
  - You want to encourage invariance with a continuous set of transformation:
    - You consider infinite number of translations/rotations instead of a fixed number.



  - Learning from simulators with random numbers (physics/chem/bio):





http://kinefold.curie.fr/cgi-bin/form.pl
https://sciencenode.org/feature/sherpa-and-open-science-grid-predicting-emergence-jets.php

Stochastic Gradient with Infinite Data

- Previous slide gives examples with infinite sequence of IID samples.
- How can you practically train on infinite-sized datasets?
- Approach 1 (exact optimization on finite 'n'):
  - Grab 'n' data points, for some really large 'n'.
  - Fit a regularized model on this fixed dataset ("empirical risk minimization").
- Approach 2 (stochastic gradient for 'n' iterations):
  - Run stochastic gradient iteration for 'n' iterations.
  - Each iteration considers a new example, never re-visiting any example.

Stochastic Gradient with Infinite Data

- Approach 2 works because of an amazing property of stochastic gradient:
  - The classic convergence analysis does not rely on 'n' being finite.

- Further Approach 2 only looks at a data point once:
  - Each example is an unbiased approximation of test data.
- So Approach 2 is doing stochastic gradient on test error:
  - It cannot overfit.

- Up to a constant, Approach 1 and 2 have same test error bound.
  - This is sometimes used to justify SG as the "ultimate" learning algorithm.
    - "Optimal test error by computing gradient of each example once!"
  - In practice, Approach 1 usually gives lower test error.
    - The constant factor matters!

A Practical Strategy For Choosing the Step-Size

- All these step-sizes have a constant factor in the "O" notation.
  - E.g., $\alpha^t = \frac{\gamma}{\sqrt{t}}$ ← How do we choose this constant?

- We don't know how to set step size as we go in the stochastic case.
  - And choosing wrong γ can destroy performance.

- Common practical trick:
  - Take a small amount of data (maybe 5% of the original data).
  - Do a binary search for γ that most improves objective on this subset.