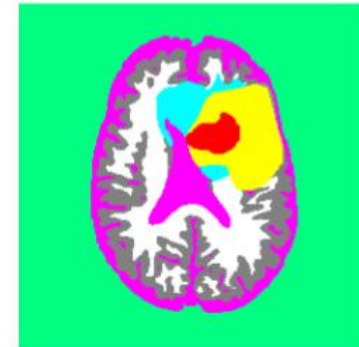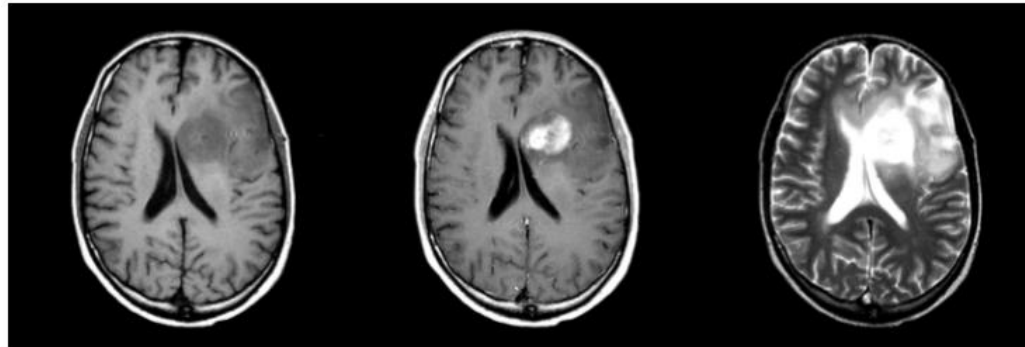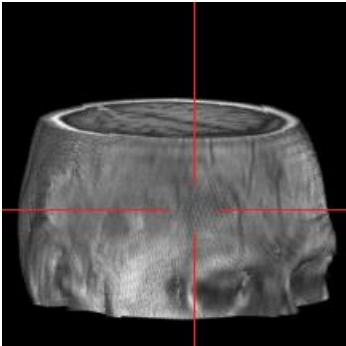# CPSC 340:
Machine Learning and Data Mining

Stochastic Gradient

Fall 2022

Motivation: Big-N Problems

- Recall the automatic brain tumour segmentation problem:

- MRI scanners at the time produced 200x200x200 volumes.
  - So one volume gives 8 million training examples.
  - And you need to train on more than one volume!

- Similar issues arise in the Gmail application:
  - If every e-mail is a training example, you have LOTS of training examples.

Motivation: Big-N Problems

- Consider fitting a least squares model:

$$f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

- Gradient methods are effective when 'd' is very large.
  - O(nd) per iteration instead of O(nd^2 + d^3) to solve as linear system.

- But what if number of training examples 'n' is very large?
  - All Gmails, all products on Amazon, all homepages, all images, and so on.

Gradient Descent vs. Stochastic Gradient

- Recall the gradient descent algorithm:

$$w^{t+1} = w^t - \alpha^t \nabla f(w^t)$$

- For least squares, our gradient has the form:

$$\nabla f(w) = \sum_{i=1}^{n} \underbrace{(w^T x_i - y_i)}_{\text{scalar}} \underbrace{x_i}_{d \times 1}$$

- So the cost of computing this gradient is linear in 'n'.
  - As 'n' gets large, gradient descent iterations become expensive.

Gradient Descent vs. Stochastic Gradient

- Common solution to this problem is stochastic gradient algorithm:

$$w^{t+1} = w^t - \alpha^t \nabla f_i(w^t)$$

- Uses the gradient of a randomly-chosen training example 'i':

$$\nabla f_i(w) = \underbrace{(w^T x_i - y_i}_{\text{scalar}} \underbrace{) x_i}_{d \times 1}$$

- Cost of computing this one gradient is independent of 'n'.
  - Iterations are 'n' times faster than gradient descent iterations.
  - With 1 billion training examples, this iteration is 1 billion times faster.

Stochastic Gradient Descent (SGD)

- **Stochastic gradient descent (SGD)** for training with large 'n':
  - We start with some initial guess, w^0.
  - Generate new guess by moving in the negative gradient direction:

  $$w^1 = w^0 - \alpha^0 \nabla f_i(w^0)$$

    - For a random training example 'i'.
  - Repeat to successively refine the guess:

  $$w^{t+1} = w^t - \alpha^t \nabla f_i(w^t) \qquad for \quad t = 1, 2, 3, \ldots$$

    - For a random training example 'i'.

Problem where we can use Stochastic Gradient

- Stochastic gradient applies when minimizing averages:

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2 \quad (\text{squared error})$$

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i w^\top x_i)) \quad (\text{logistic regression})$$

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \left[\log(1 + \exp(-y_i w^\top x_i)) + \frac{\lambda}{2} \|w\|^2 \right] \quad (L_2\text{-regularized logistic})$$

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \quad (\text{our notation for the general case})$$

- Basically, all our regression losses except "brittle" regression.
  - Recall: multiplying by positive constant doesn't change location of optimal 'w'.

Why Does Stochastic Gradient Work / Not Work?

- Main problem with stochastic gradient:
  - Gradient of random example might point in the wrong direction.

- Does this have any hope of working?
  - The expected direction is the full gradient.

$$E[\nabla f_i(w^k)] = \sum_{i=1}^{n} p^{(i)} \nabla f_i(w^k) = \sum_{i=1}^{n} \frac{1}{n} \nabla f_i(w^k) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w^t) = \nabla f(w^k)$$

expectation over choice of random example i

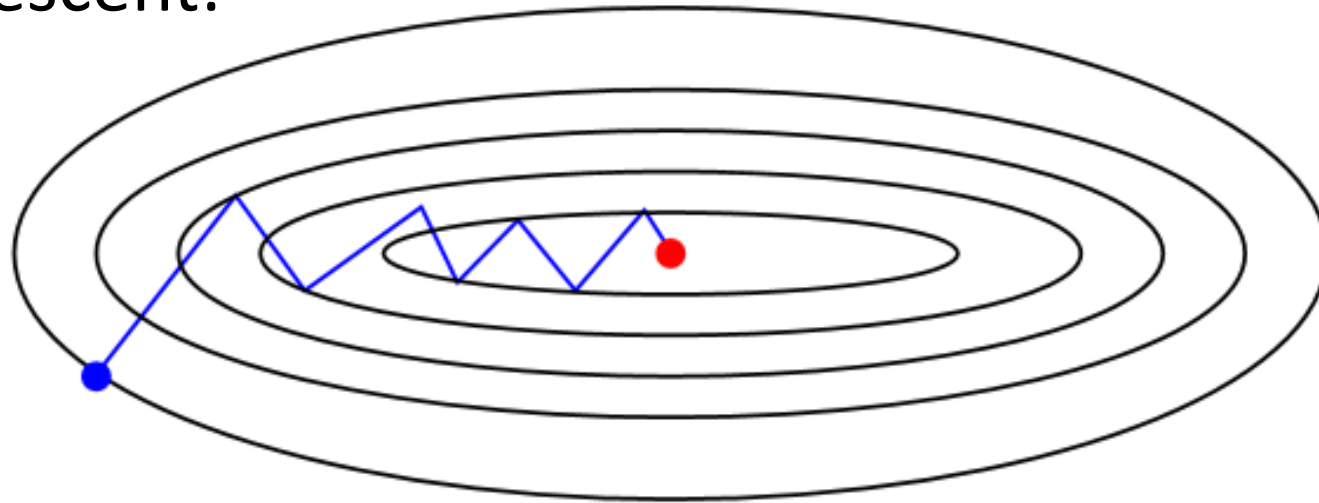definition of expectation

if each example is equally likely

gradient over all examples

  - The algorithm is going in the right direction on average.
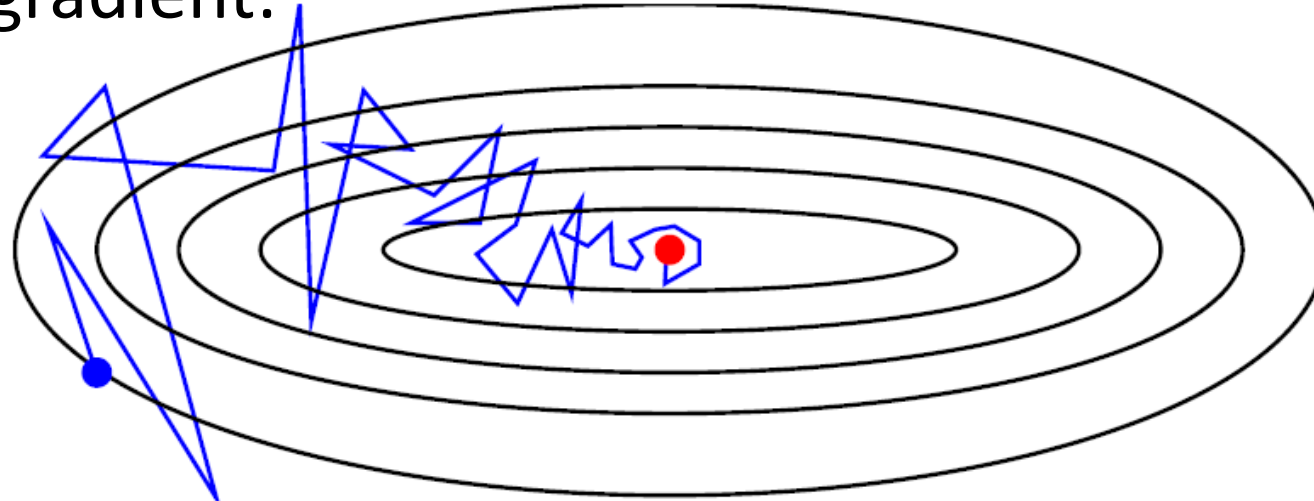
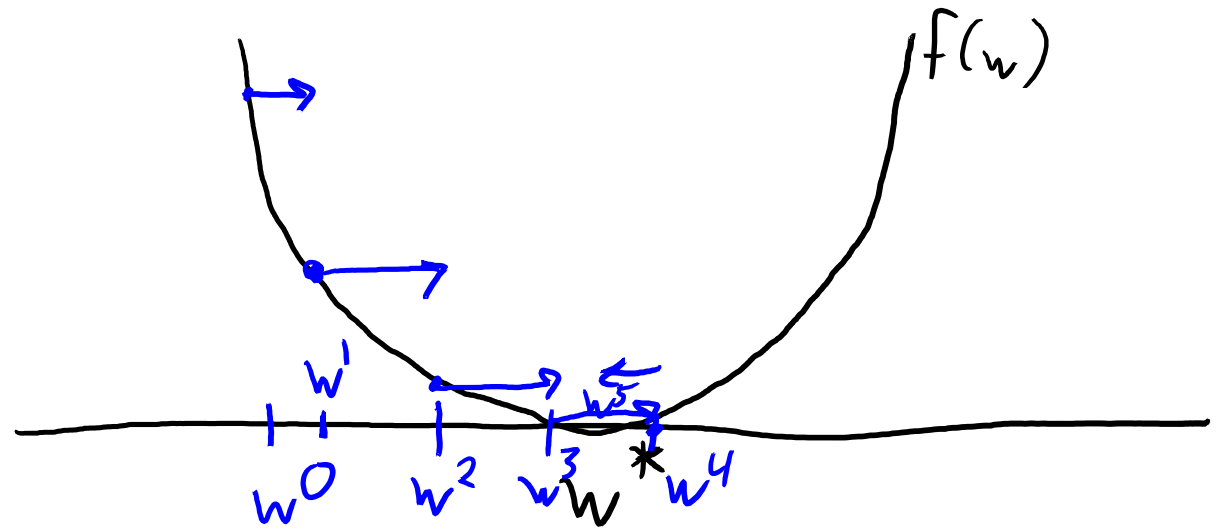# Gradient Descent vs. Stochastic Gradient (SG)

- Gradient descent:

- Stochastic gradient:

Gradient Descent in Action

$$f(w) = \frac{1}{5} \sum_{i=1}^{5} (w^7 x_i - y_i)^2$$

$f(w)$

w^1

w^0 w^2 w^3 w w^4

Stochastic Gradient in Action

$$f(w) = \frac{1}{5} \sum_{i=1}^{5} (w^T x_i - y_i)^2$$

$$f_1(w) = (w^T x_1 - y_1)^2$$

$$f_2(w) = (w^T y_2 - y_2)^2$$

$$f_3(w) = (w^T x_3 - y_3)^2$$

$$f_4(w) = (w^T x_4 - y_4)^2$$

$$f_5(w) = (w^T x_5 - y_5)^2$$

$f(w)$

w^0

w^*

$f_1(w)$ $f_2(w)$ $f_3(w)$ $f_4(w)$ $f_5(w)$

Stochastic gradient minimizes <u>average</u> value

w^0

w^*

Stochastic Gradient in Action

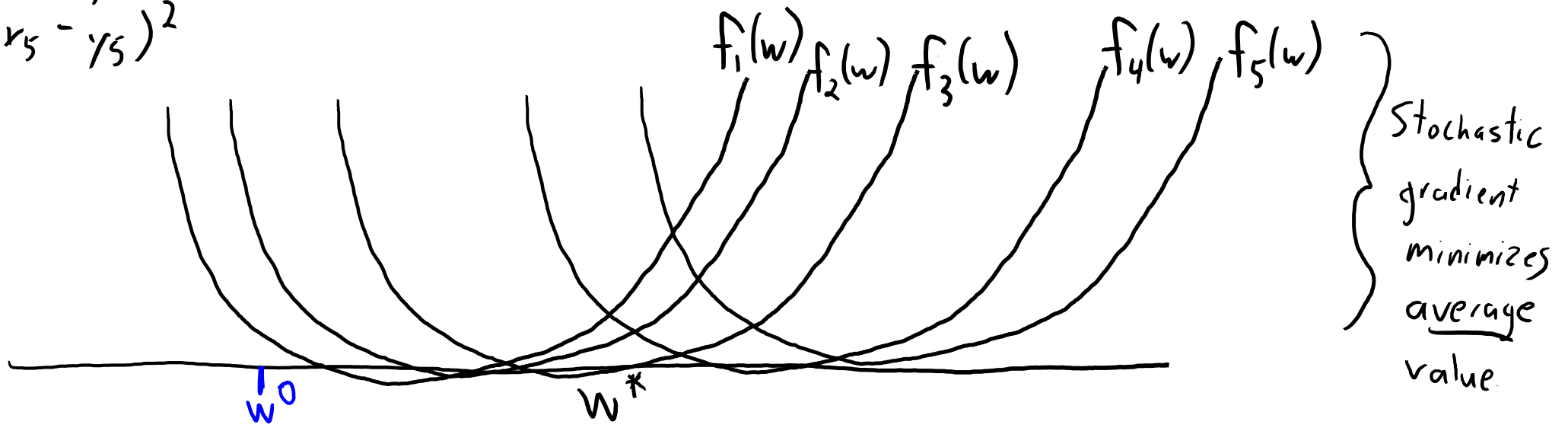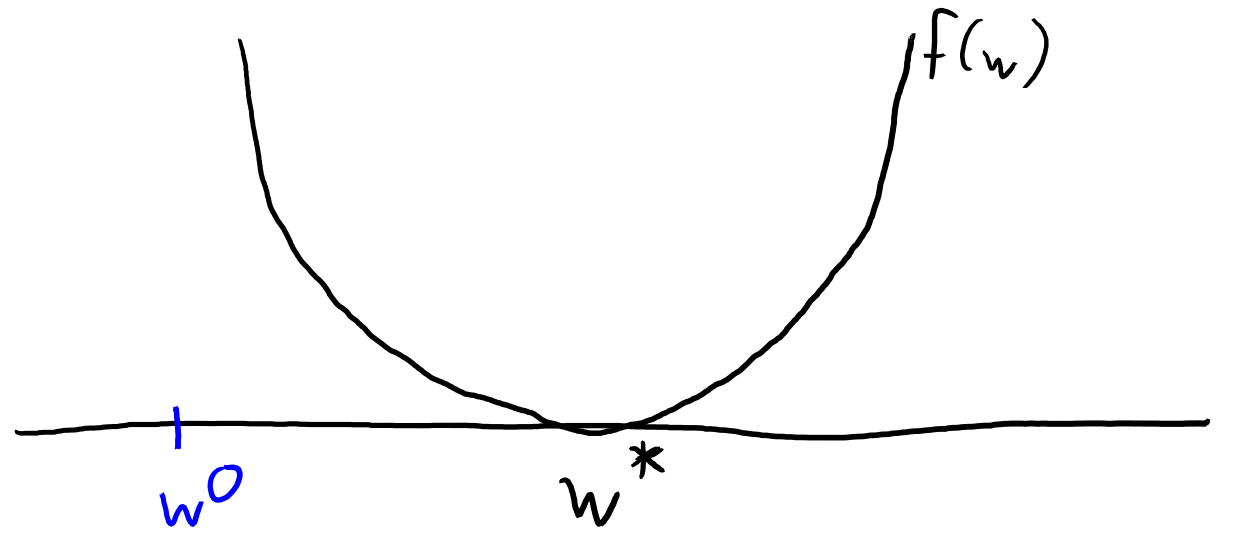$$f(w) = \frac{1}{5} \sum_{i=1}^{5} (w^T x_i - y_i)^2$$

$$f_1(w) = (w^T x_1 - y_1)^2$$

$$f_2(w) = (w^T y_2 - y_2)^2$$

$$f_3(w) = (w^T x_3 - y_3)^2$$

$$f_4(w) = (w^T x_4 - y_4)^2$$

$$f_5(w) = (w^T x_5 - y_5)^2$$

$f(w)$

w^0 w^1 w^*

$f_1(w)$ $f_2(w)$ $f_3(w)$ $f_4(w)$ $f_5(w)$

w^0 w^1 w^*

Stochastic gradient minimizes average value

Stochastic Gradient in Action

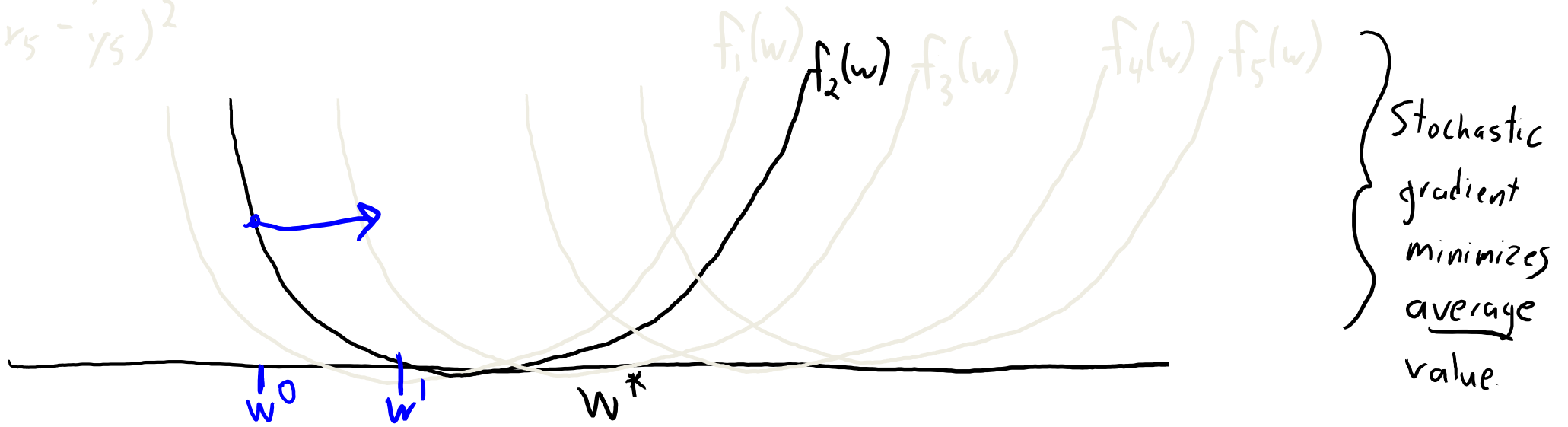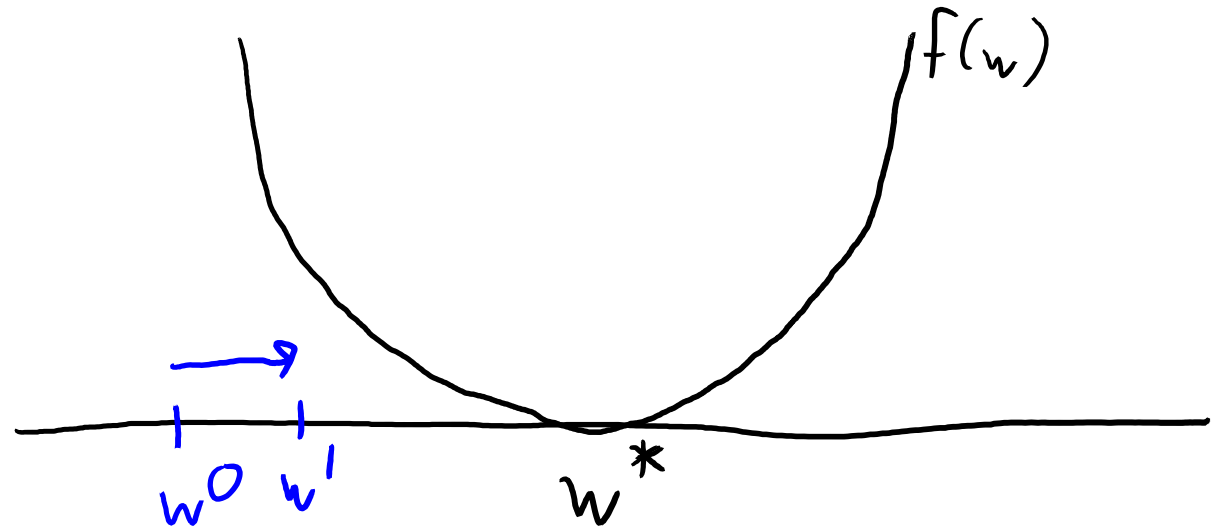$$f(w) = \frac{1}{5}\sum_{i=1}^{5}(w^T x_i - y_i)^2$$

$$f_1(w) = (w^T x_1 - y_1)^2$$

$$f_2(w) = (w^T x_2 - y_2)^2$$

$$f_3(w) = (w^T x_3 - y_3)^2$$

$$f_4(w) = (w^T x_4 - y_4)^2$$

$$f_5(w) = (w^T x_5 - y_5)^2$$

$f(w)$

w^0 w^1 w^2 w^*

$f_1(w)$ $f_2(w)$ $f_3(w)$ $f_4(w)$ $f_5(w)$

w^0 w^1 w^2 w^*

Stochastic gradient minimizes average value

Stochastic Gradient in Action

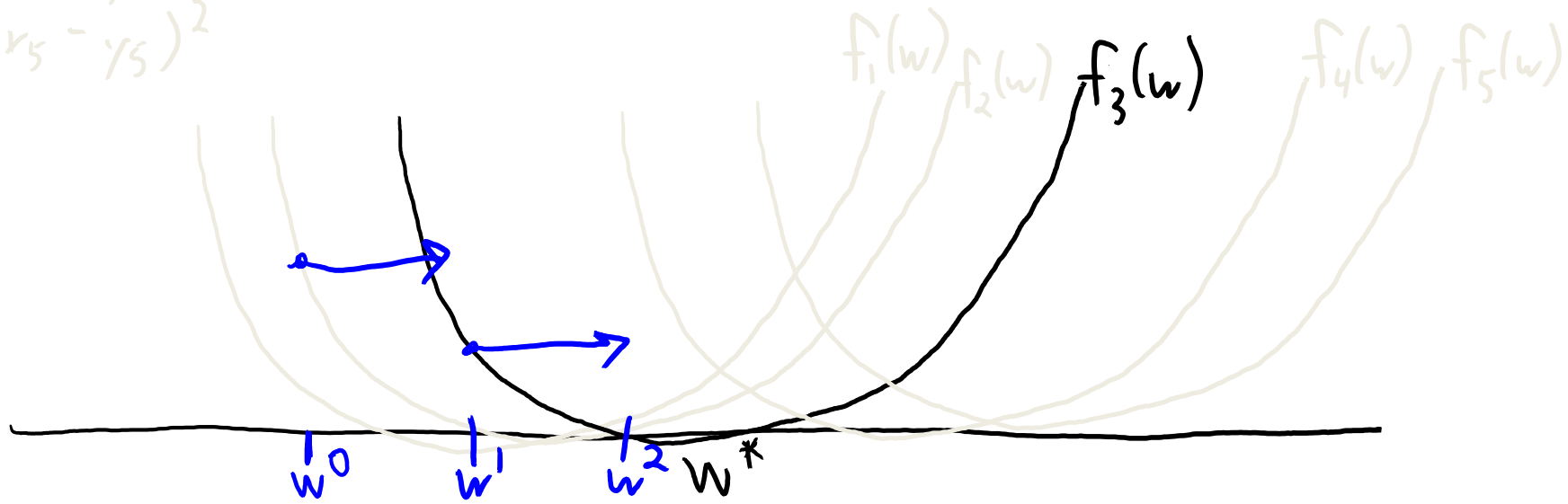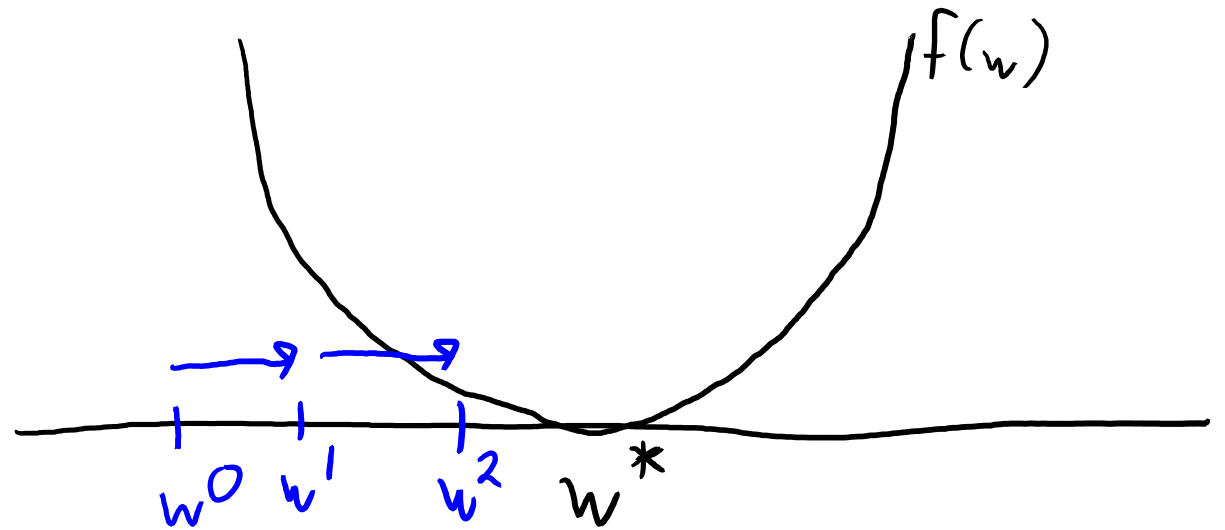$$f(w) = \frac{1}{5}\sum_{i=1}^{5}(w^T x_i - y_i)^2$$

$$f_1(w) = (w^T x_1 - y_1)^2$$

$$f_2(w) = (w^T x_2 - y_2)^2$$

$$f_3(w) = (w^T x_3 - y_3)^2$$

$$f_4(w) = (w^T x_4 - y_4)^2$$

$$f_5(w) = (w^T x_5 - y_5)^2$$

$f(w)$

$w^0 \quad w^1 w^3 \quad w^2 \quad w^*$

$f_1(w) \quad f_2(w) \quad f_3(w) \quad f_4(w) \quad f_5(w)$

$w^0 \quad w^1 w^3 \quad w^2 \quad w^*$

Stochastic gradient minimizes <u>average</u> value

Stochastic Gradient in Action

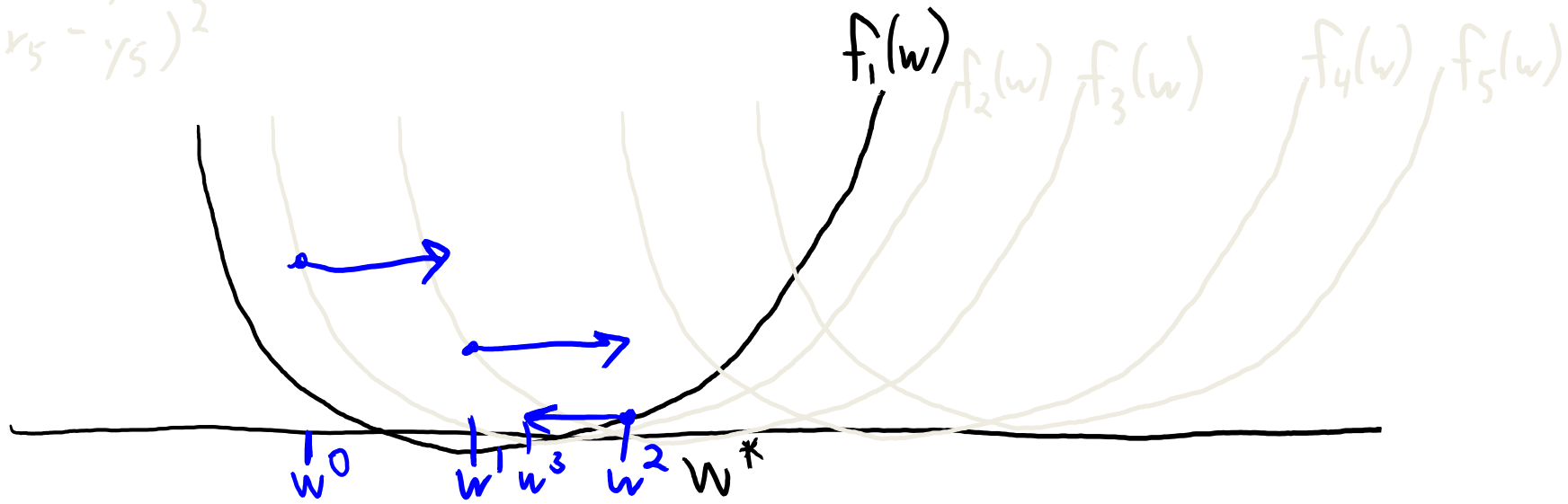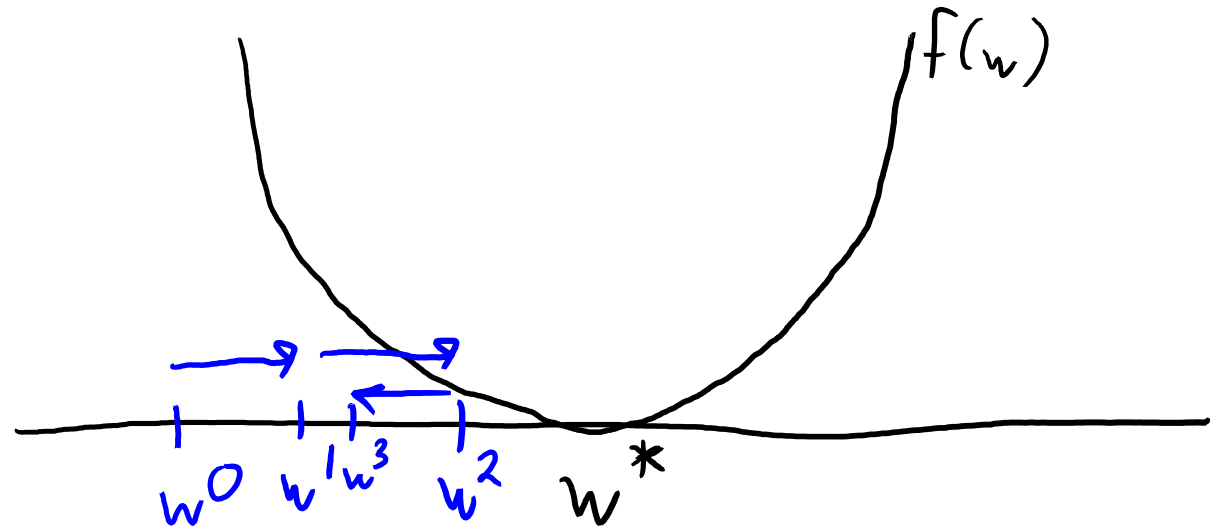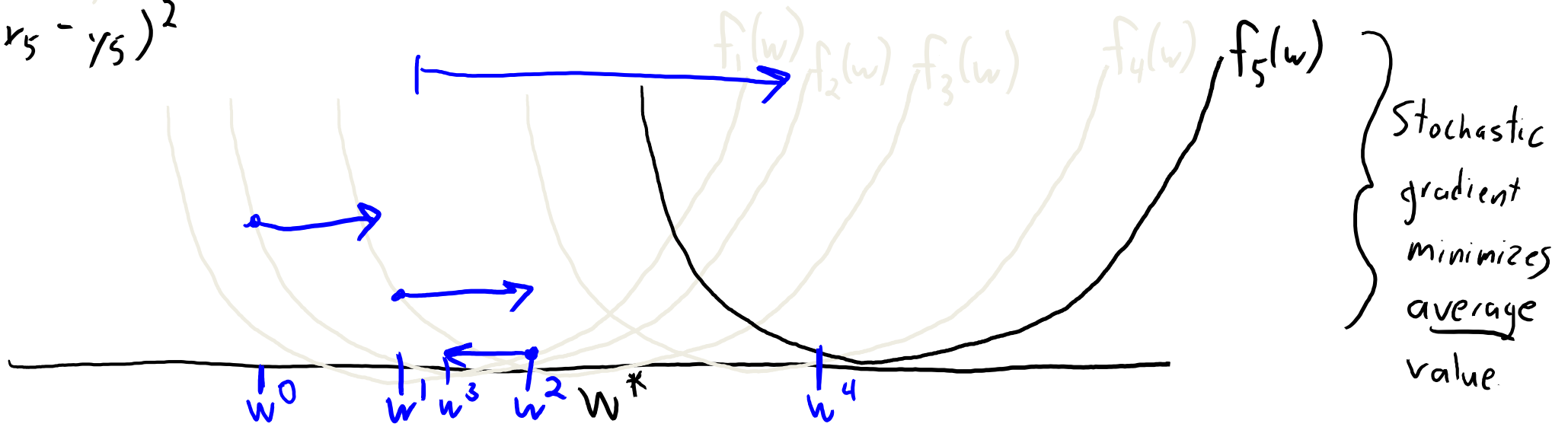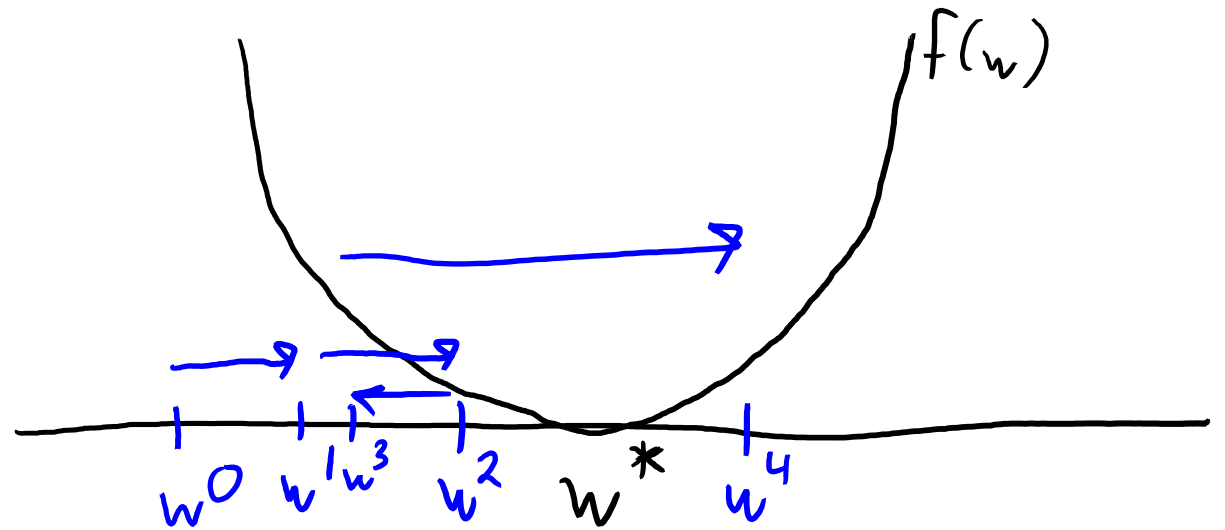$$f(w) = \frac{1}{5}\sum_{i=1}^{5}(w^T x_i - y_i)^2$$

$$f_1(w) = (w^T x_1 - y_1)^2$$

$$f_2(w) = (w^T x_2 - y_2)^2$$

$$f_3(w) = (w^T x_3 - y_3)^2$$

$$f_4(w) = (w^T x_4 - y_4)^2$$

$$f_5(w) = (w^T x_5 - y_5)^2$$

$f(w)$

w^0 w^1 w^3 w^2 w^* w^4

$f_1(w)$ $f_2(w)$ $f_3(w)$ $f_4(w)$ $f_5(w)$

w^0 w^1 w^3 w^2 w^* w^4

Stochastic gradient minimizes <u>average</u> value

Effect of 'w' Location on Progress

$f_1(w)$ $f_2(w)$ $f_3(w)$ $f_4(w)$ $f_5(w)$

w^*

Every $\nabla f_i(w)$ <u>here</u> points towards w^*

"Region of confusion": <u>some</u> $\nabla f_i(w)$ point towards w^* and <u>some</u> <u>don't</u>

Every $\nabla f_i(w)$ <u>here</u> points towards w^*
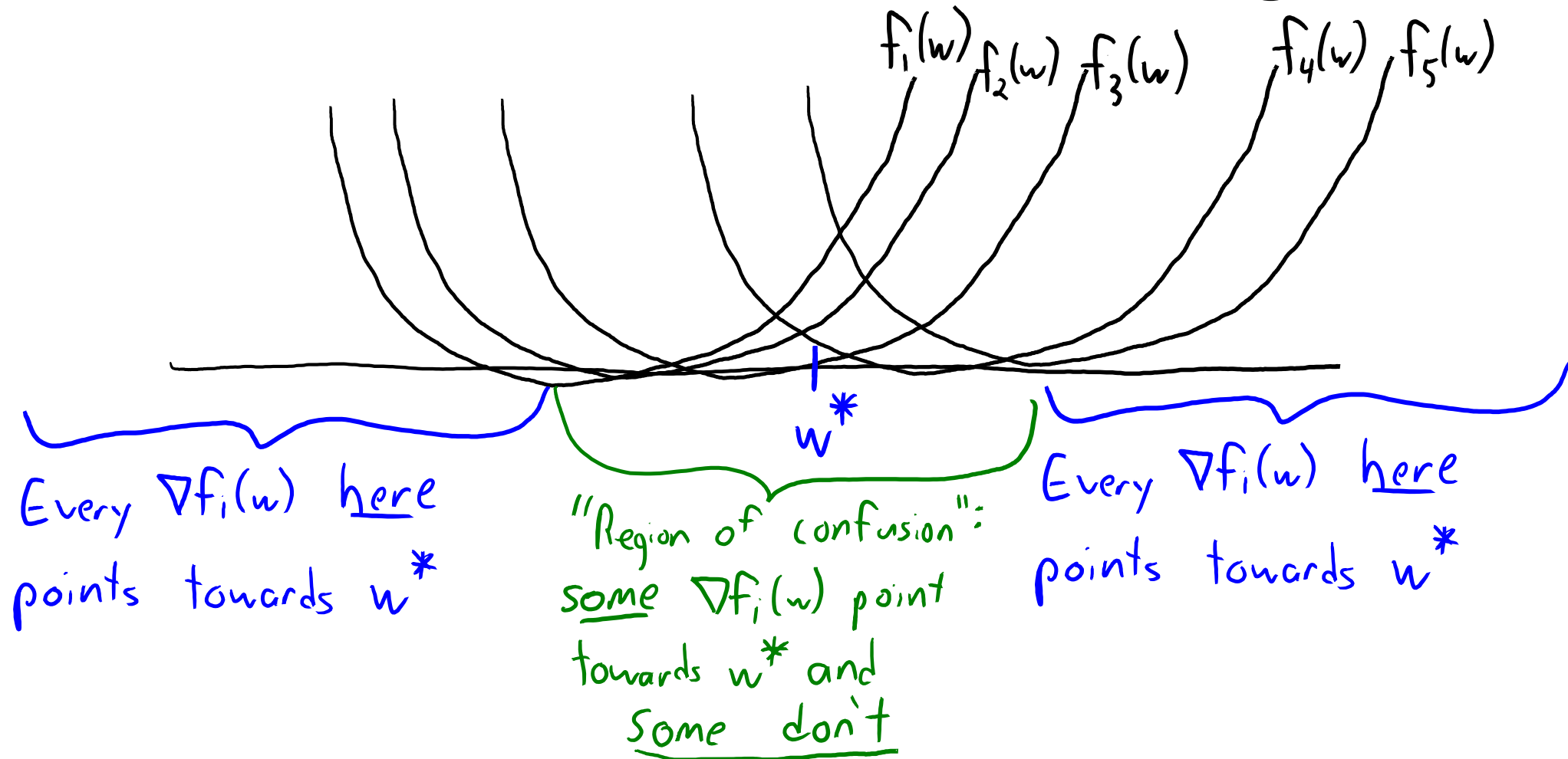
- We'll still make good progress if most gradients points in right direction.

Variance of the Random Gradients

- The "confusion" is captured by a kind of variance of the gradients:

$$\frac{1}{n} \sum_{i=1}^{n} \| \nabla f_i(w^t) - \nabla f(w^t) \|^2$$

gradient of example i: average gradient over all examples

- If the variance is 0, every step goes in the right direction.
  - We're outside of the region of confusion.
- If the variance is small, most steps point in the direction.
  - We're just inside region of confusion.
- If the variance is large, many steps will point in the wrong direction.
  - Middle of region of confusion, where w* lives.

Effect of the Step-Size

- We can reduce the effect of the variance with the step size:
  - As the step size gets smaller, the variance has less of an effect.

- For a fixed step-size, SGD makes progress until variance is too big.

- This leads to two "phases" when we use a constant step-size:
  1. Rapid progress when we are far from the solution.
  2. Erratic behaviour confined to a "ball" around solution.
     (Radius of ball is proportional to the step-size.)

Stochastic Gradient with Constant Step Size
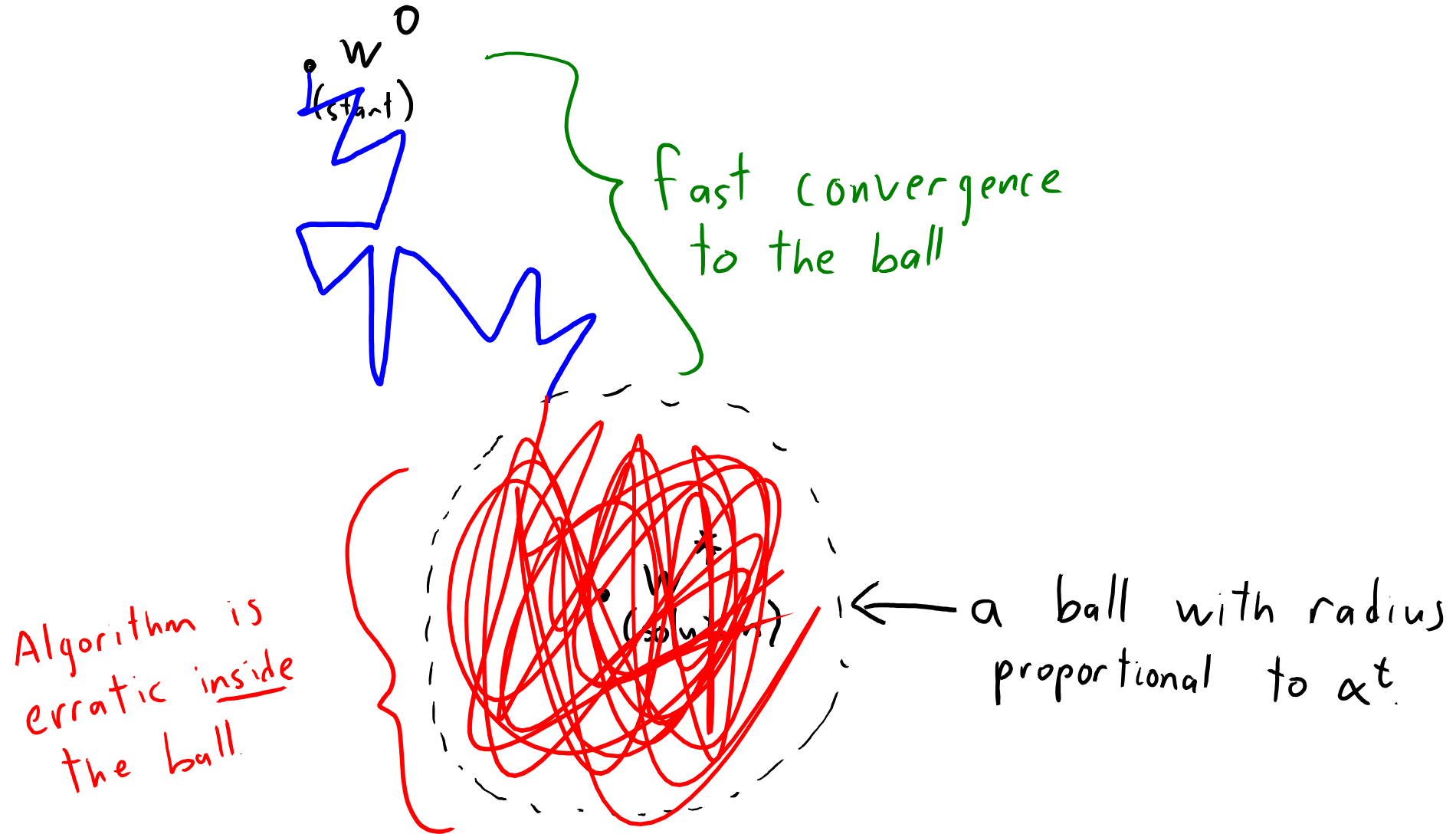
w^0
(start)

fast convergence
to the ball

Algorithm is
erratic *inside*
the ball.

w^*
(solution)

← a ball with radius
proportional to α^t

Stochastic Gradient with Constant Step Size

w^0 (start)

fast convergence to the ball

We can divide the radius of ball in 2 by dividing α^t by 2. (but takes longer to get to ball)

Algorithm is erratic <u>inside</u> the ball

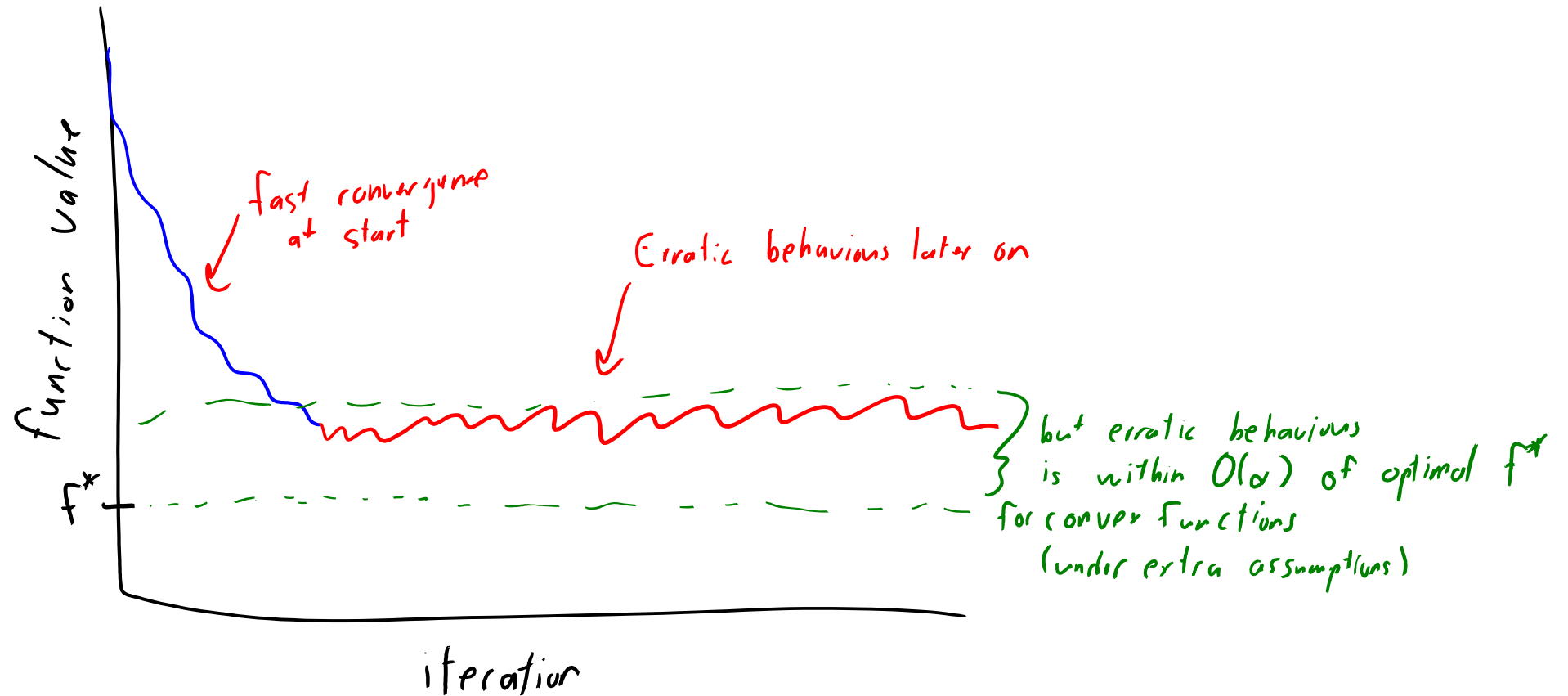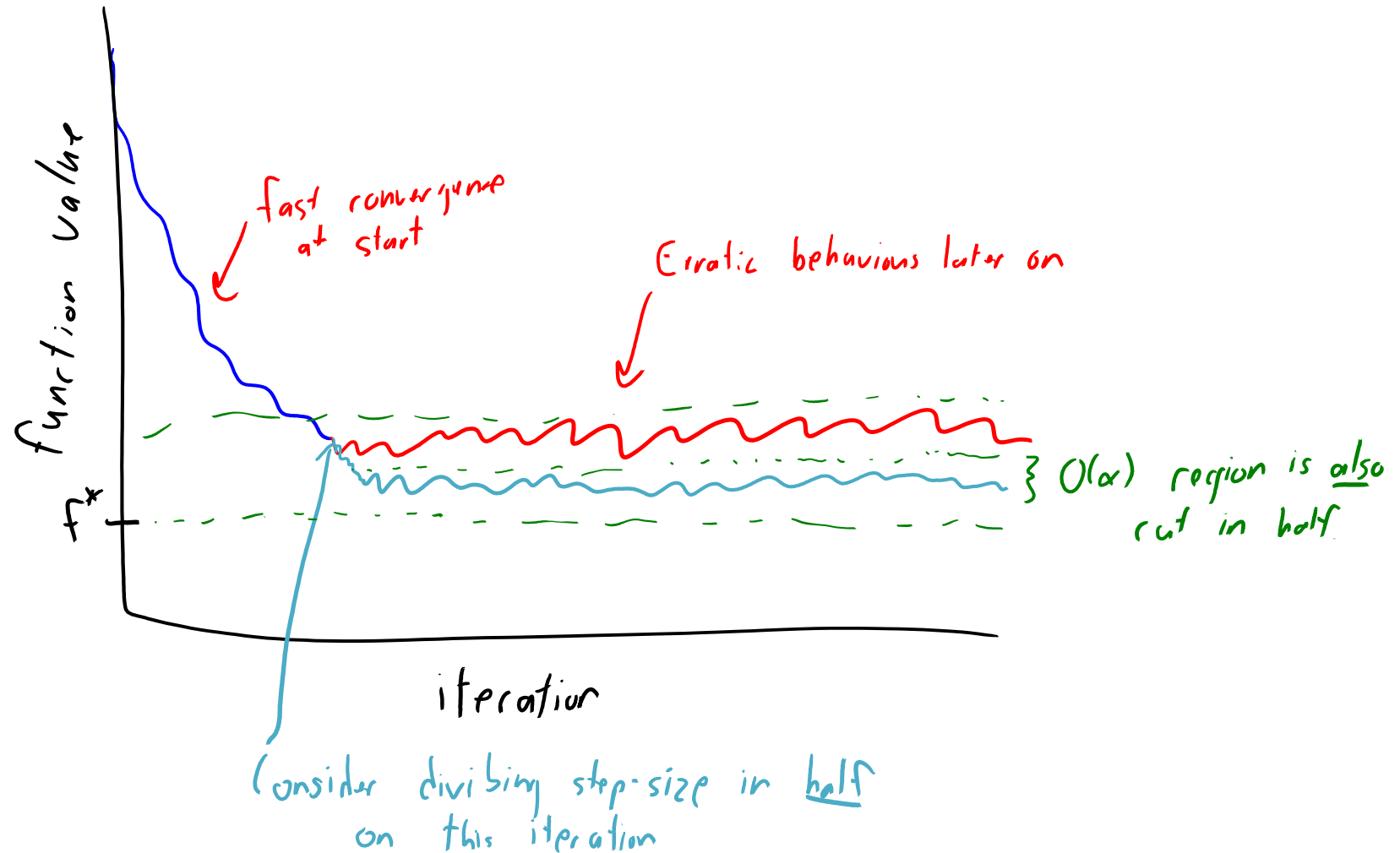w^* (solution)

a ball with radius proportional to α^t
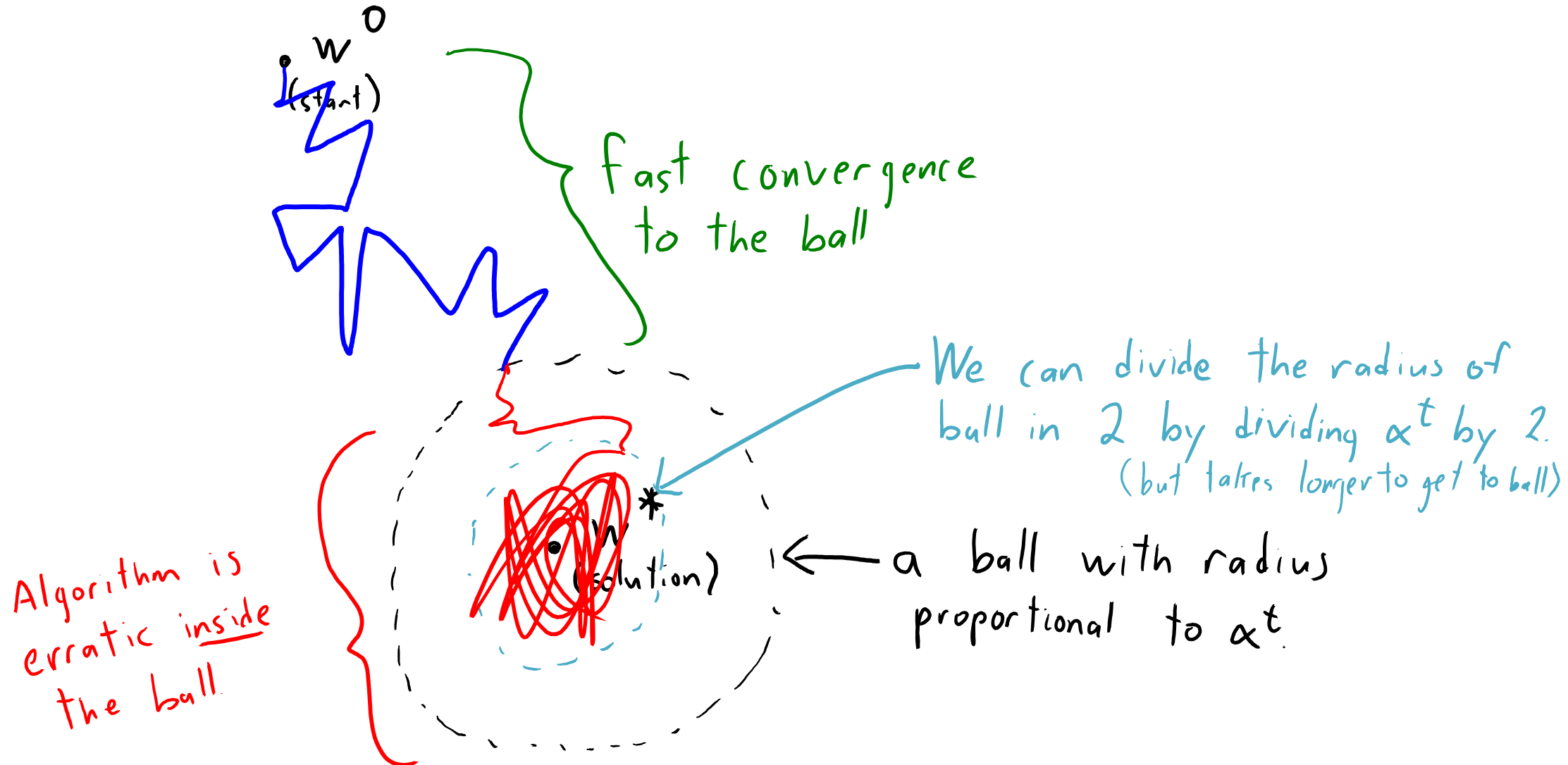
Stochastic Gradient with Constant Step Size

Stochastic Gradient with Constant Step Size

Stochastic Gradient with Decreasing Step Sizes

- To get convergence, we need a decreasing step size.
  - Shrinks size of ball to zero so we converge to w^*.
- But it can't shrink too quickly:
  - Otherwise, we don't move fast enough to reach the ball.
- Stochastic gradient converges to a stationary point if:
  - Ratio of sum of squared step-sizes over sum of step-sizes converges to 0.

$$\text{"how much noise affects you"} \rightarrow \frac{\sum_{t=1}^{\infty} (\alpha^t)^2}{\sum_{t=1}^{\infty} \alpha^t} = 0 \quad \leftarrow \text{"how far you can get"}$$

  - This choice also works for non-smooth funtions like SVMs.
    - Function must be continuous and not "too crazy" (we're still figuring it out for non-convex).

Stochastic Gradient with Decreasing Step Sizes

- For convergence step-sizes need to satisfy: $\sum_{t=1}^{\infty}(\alpha^t)^2 / \sum_{t=1}^{\alpha}\alpha^t = 0$

- Classic solution is to use a step-size sequence like $\alpha^t = O(1/t)$.
  - Like setting $\alpha^t = .001/t$.

$$\sum_{t=1}^{\infty}\alpha^t = \sum_{t=1}^{\infty}\frac{1}{t} = \infty \qquad \sum_{t=1}^{\infty}(\alpha^t)^2 = \sum_{t=1}^{\infty}\frac{1}{t^2} < \infty$$

"we can get everywhere"   "effect of variance goes to zero"

- Unfortunately, this often works badly in practice:
  - Steps get really small really fast.
  - Some authors add extra parameters like $\alpha^t = \gamma/(\beta t + \Delta)$, which helps a bit.
  - One of the only cases where this works well: binary SVMs with $\alpha^t = 1/\lambda t$.

Stochastic Gradient with Decreasing Step Sizes

- For convergence step-sizes need to satisfy: $\sum_{t=1}^{\infty}(\alpha^t)^2 \Big/ \sum_{t=1}^{\alpha}\alpha^t = 0$

- Better solution is to use a step-size sequence like $\alpha^t = O(1/\sqrt{t})$.
  - Like setting $\alpha^t = .001/\sqrt{t}$

$$\sum_{t=1}^{k}\alpha^t = \sum_{t=1}^{k}\frac{1}{\sqrt{t}} = O(\sqrt{k})$$

$$\sum_{t=1}^{k}(\alpha^t)^2 = \sum_{t=1}^{k}\frac{1}{t} = O(\log k)$$

  - Both sequences diverge, but denominator diverges faster.

- This approach (roughly) optimizes rate that ratio goes to zero.
  - Better worst-case theoretical properties (and more robust to step-size).
  - Often better in practice too.

Stochastic Gradient with Constant Step Sizes?

- Alternately, could we just use a constant step-size?
  - Like setting $\alpha^t = .001$ for all 't'.

- This will not converge to a stationary point in general.
  - However, do we need it to converge?

- What if you only care about the first 2-3 digits of the test error?
  - Who cares if you are not able to get 10 digits of optimization accuracy?

- There is a step-size small enough to achieve any fixed accuracy.
  - Just need radius of "ball" to be small enough.

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).
    - Do not need decreasing step sizes if batch size is increasing ("batching").
  - Useful for vectorizing/parallelizing code.
    - Evaluate one gradient on each core.

Gradient Descent vs. Stochastic Gradient vs. Hybrid
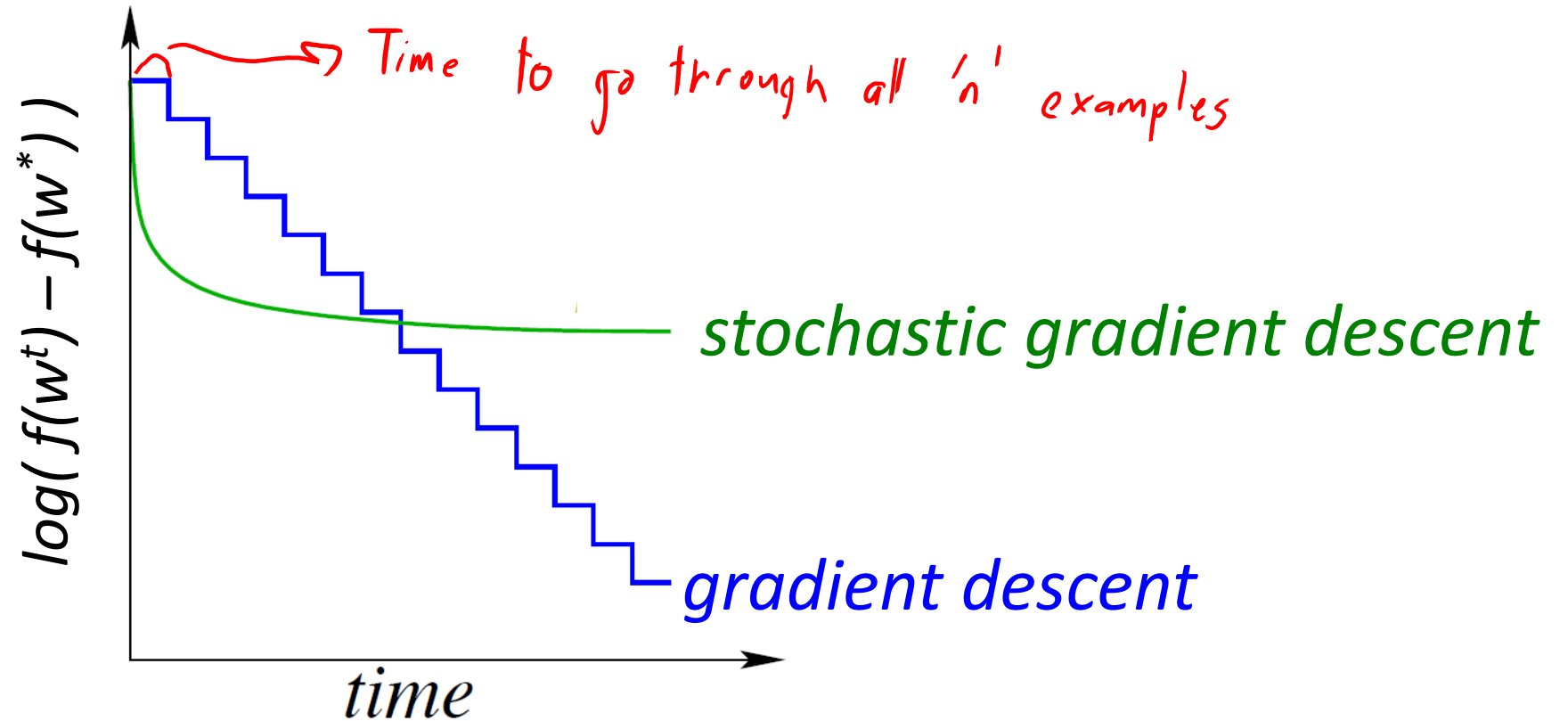

- Gradient descent makes consistent progress with slow iterations.
- Stochastic gradient descent has fast iterations but progress slows.
  - Algorithm to choose depends on available time and desired accuracy.

Gradient Descent vs. Stochastic Gradient vs. Hybrid
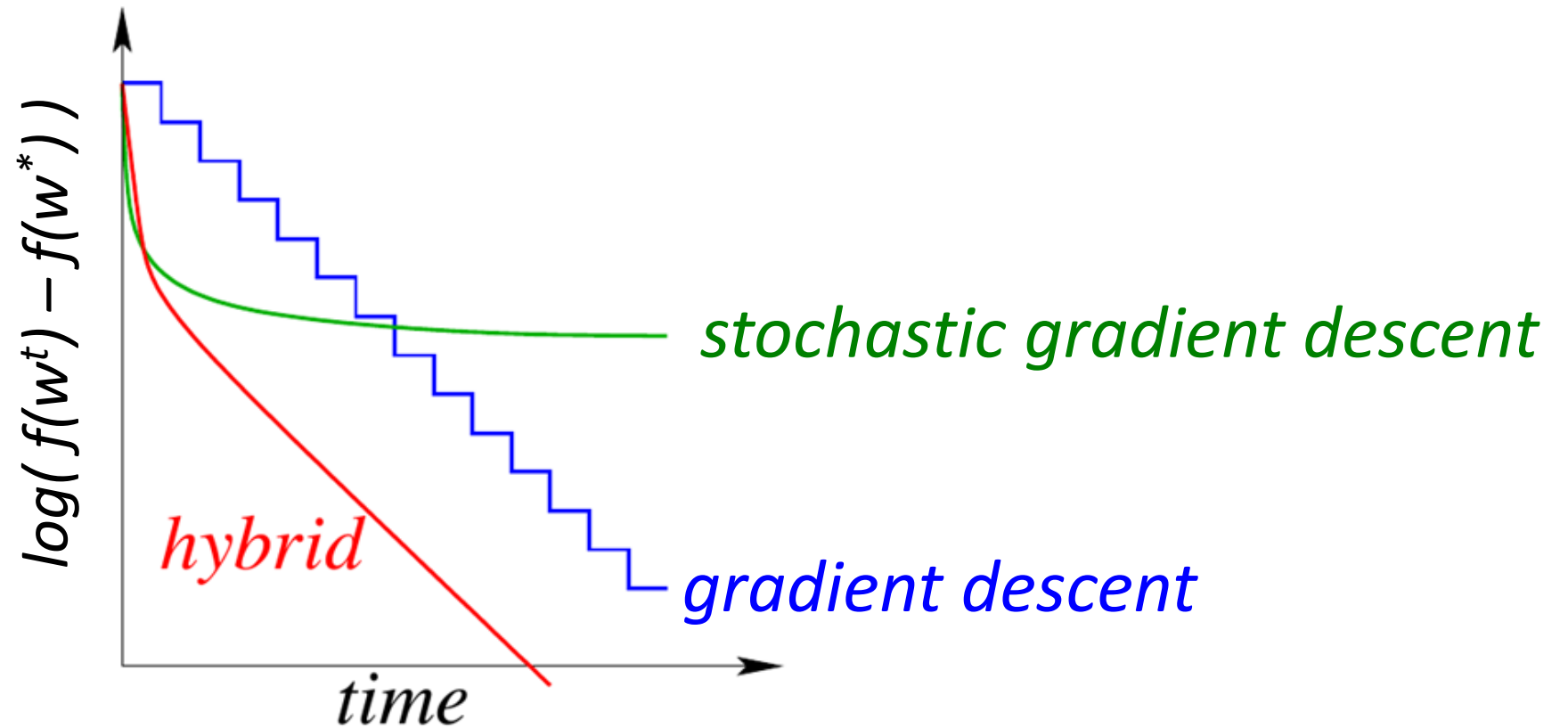

- Hybrid deterministic-stochastic methods can outperform both.
  - For example, batching methods that grow the batch size.
  - Make consistent progress but eventually have high iteration costs.

# Gradient Descent vs. Stochastic Gradient vs. Hybrid
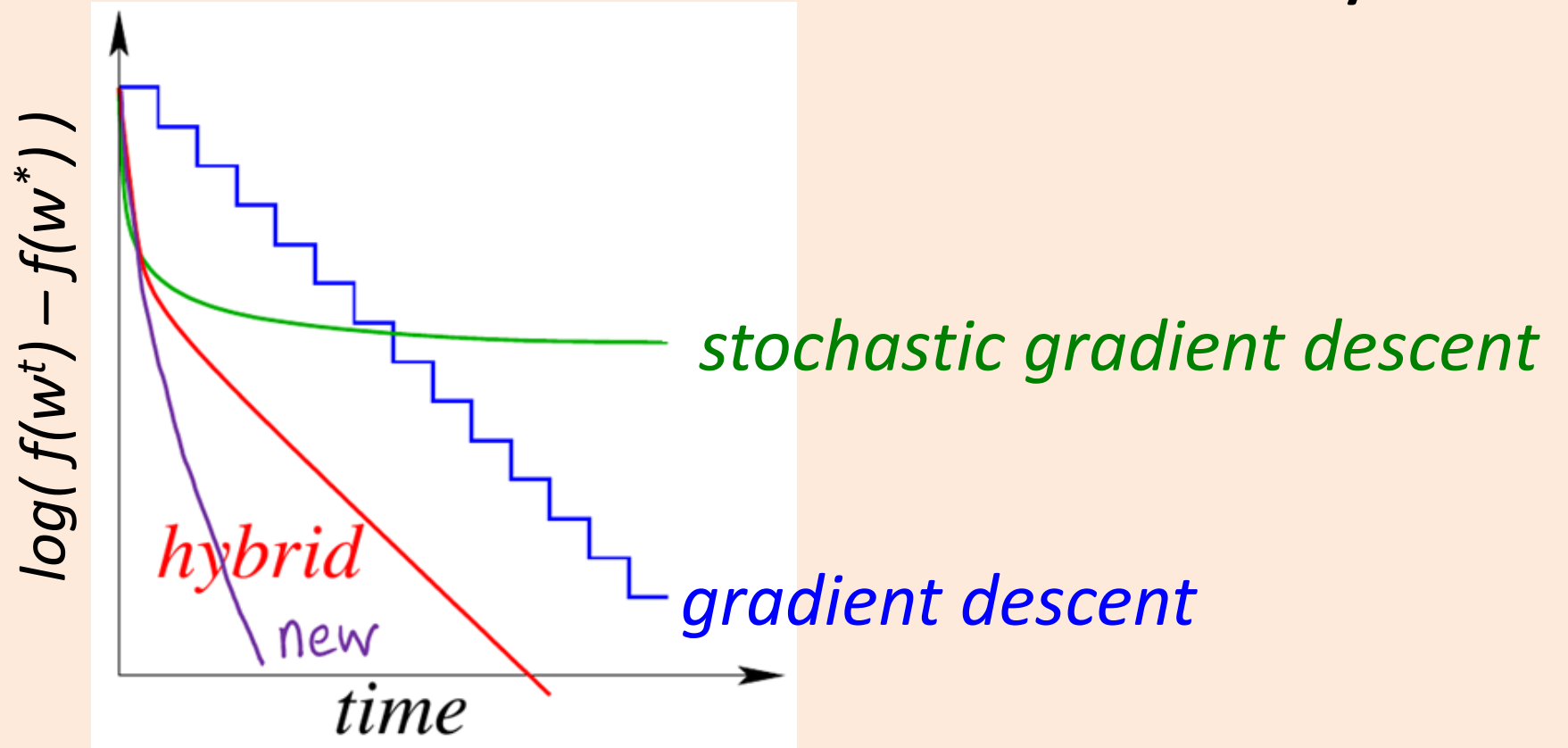
- 2012: methods with cost of stochastic gradient, progress of full gradient.
  - Key idea: if 'n' is finite, you can use a memory instead of having α_t go to zero.
  - First was stochastic average gradient (SAG), "low storage" version is SVRG.
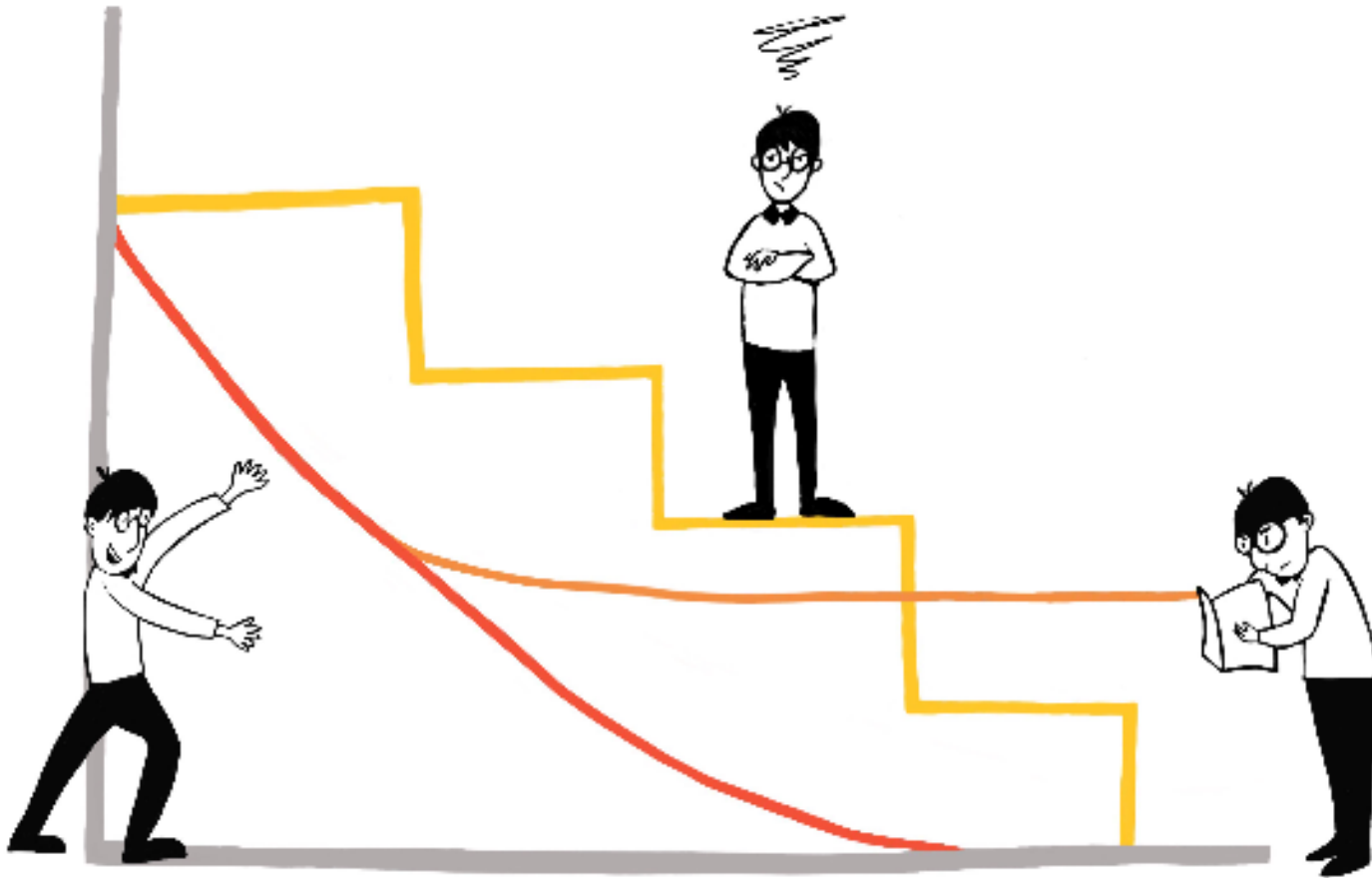
This graph shows how algorithms have become fast and more efficient over time. The horizontal axis represents time and the vertical axis represents error. Older algorithms (yellow) were very slow but had very little error. Faster algorithms were created by only analyzing some of the data (orange). The method was faster but had an accuracy limit. Schmidt's algorithm is faster and has no accuracy limit. *Aiken Lao / The Ubyssey*
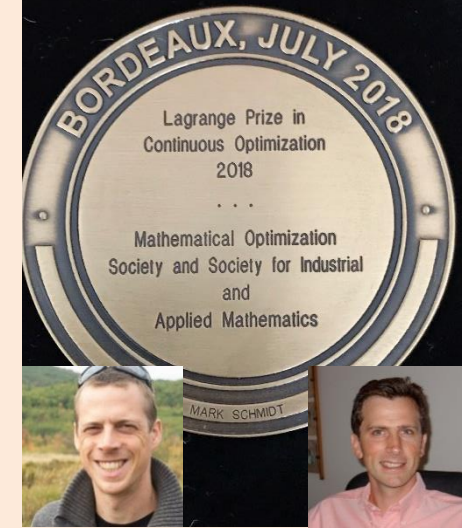
Random vs. Re-shuffle SGD

- Classic SGD: i_k sampled randomly "with replacement".
- SGD with "random shuffling":
  - Generate a random ordering of the example 1,2,…,n.
  - Do an SGD step on each example, following the random order.
    - Sampling "without replacement".
  - Go back to step 1.
- We used to view this as an "engineering trick".
  - It seemed to work better than random SGD but we did not know why.
- We now know random shuffling converges faster in many cases.
 - This was only proven recently.

Early Stopping of SGD

- In gradient descent, we can stop when gradient is close to zero.

- In stochastic gradient:
  - Individual gradients do not necessarily go to zero.
  - We cannot see full gradient, so we do not know when to stop.

- Practical trick:
  - Every 'k' iterations (for some large 'k'), measure validation set error.
  - Stop if the validation set error "is not improving".
    - This "early stopping" can also reduce overfitting.

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.
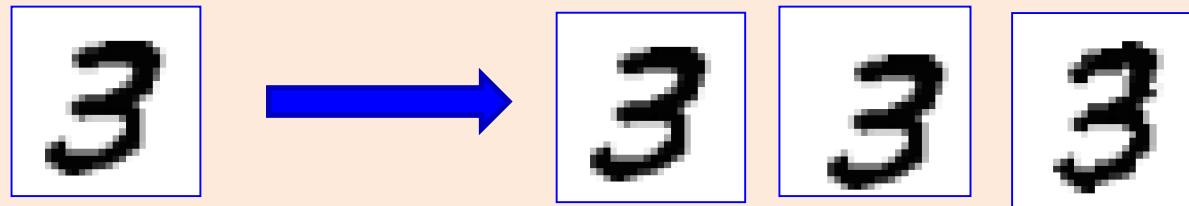
  - Learning from simulators with random numbers (physics/chem/bio):
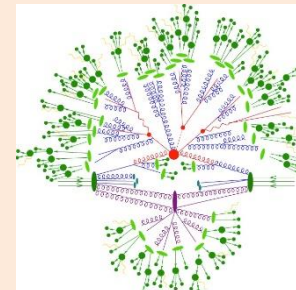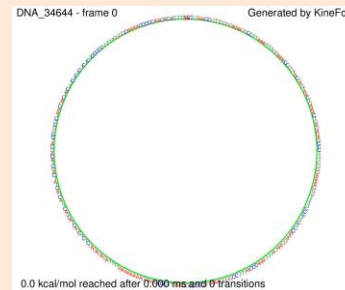
Stochastic Gradient with Infinite Data

- 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 SGD 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!

Summary

- Stochastic gradient methods let us use huge datasets.
- Step-size in stochastic gradient is a huge pain:
  - Needs to go to zero to get convergence, but classic $O(1/t)$ steps are bad.
  - $O(1/\sqrt{t})$ works better, but still pretty slow.
  - Constant step-size is fast, but only up to a certain point.
- SGD practical issues: mini-batching, early stopping.
- SAG and other methods fix SGD convergence for finite datasets.
- Infinite datasets can be used with SGD and do not overfit.

- Next time:
  - A model that has been dominating Kaggle ML competitions.

A Practical Strategy For Choosing the Step-Size

- All these step-sizes have a constant factor in the "O" notation.
  - E.g., $\alpha^t = \dfrac{\gamma}{\sqrt{t}}$ ← 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.

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.

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

Here, v^k is a scalar

"weight" of iteration 'k'

Uniform average:

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

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

- 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

w^0
(start)

fast convergence
to the ball

Often, you average the second half of the iterations.

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

Average of erratic behaviour can converge to w^*

w^*
(solution)

a ball with radius proportional to α^t.