CPSC 340:
Machine Learning and Data Mining

Stochastic Gradient

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Motivation: Big-n Problems

- Consider fitting a least squares model:

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

- 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, etc.

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:

$$\nabla f_i(w) = (w^T x_i - y_i) x_i$$

- 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 (SG)

- Stochastic gradient is an iterative optimization algorithm:
  - 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:

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

    - For a random training example 'i'.

Stochastic Gradient (SG)

- Stochastic gradient applies when minimizing averages:

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

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

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

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

- Basically, all our regression losses except “brittle” regression.
  - Multiplying be positive constant doesn’t change location of optimal ‘w’.

Intuition: per-example gradients

- The gradient (derivative) and summation are both linear operators
  - This means we can switch the order of the gradient and the summation
- The losses we use are an average of per-example losses:

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

- That means the gradient is an average of per-example gradients:

$$\nabla_w F(w) = \nabla_w \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla_w f_i(w)$$

- With SG we are randomly sampling one of these gradients instead of averaging all of them
  - This is an estimate of the average that is faster to compute

Why Does Stochastic Gradient Work / Not Work?

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

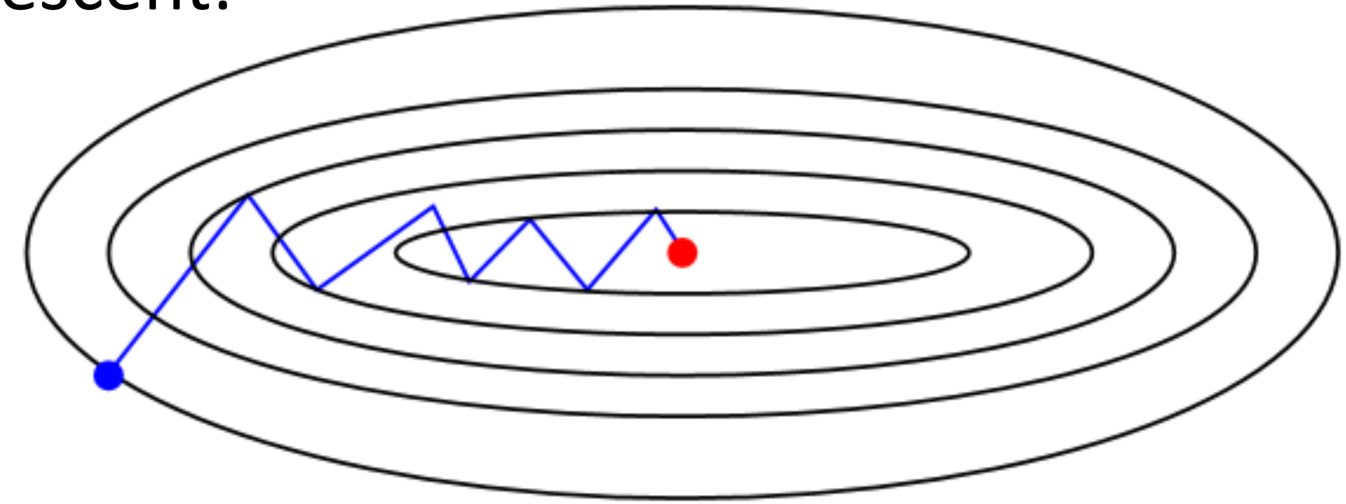
- Does this have any hope of working?
  - The average of the random gradients is the full gradient.

Mean over $\nabla f_i(w^t)$ is $\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^t)$ which is $\nabla f(w^t)$

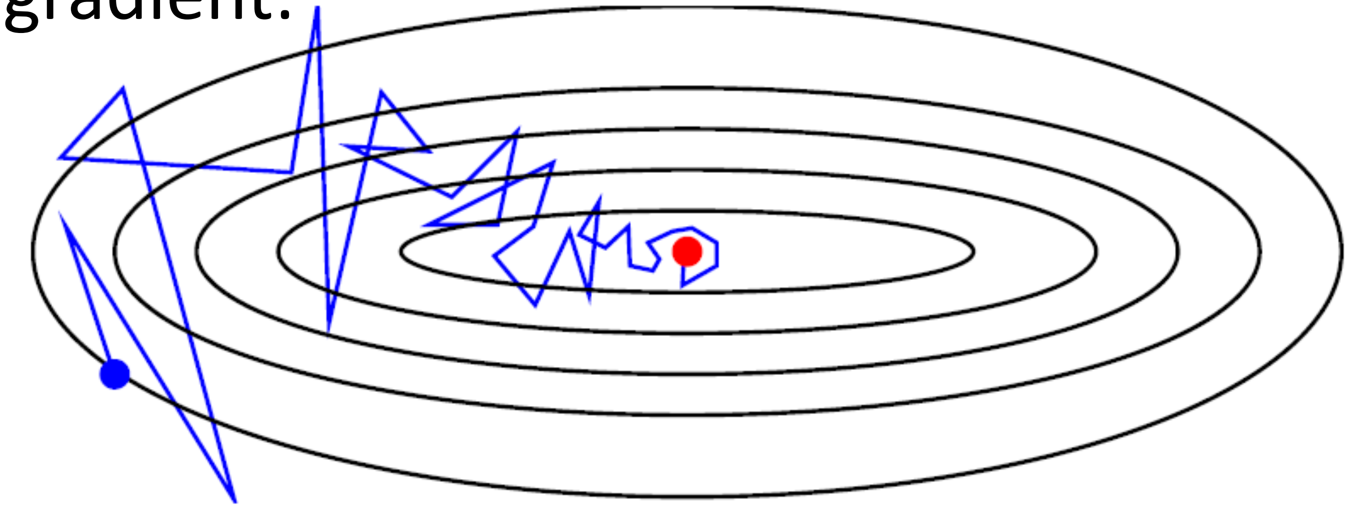
  - The algorithm is going in the right direction on average.

Gradient Descent vs. Stochastic Gradient (SG)

- Gradient descent:

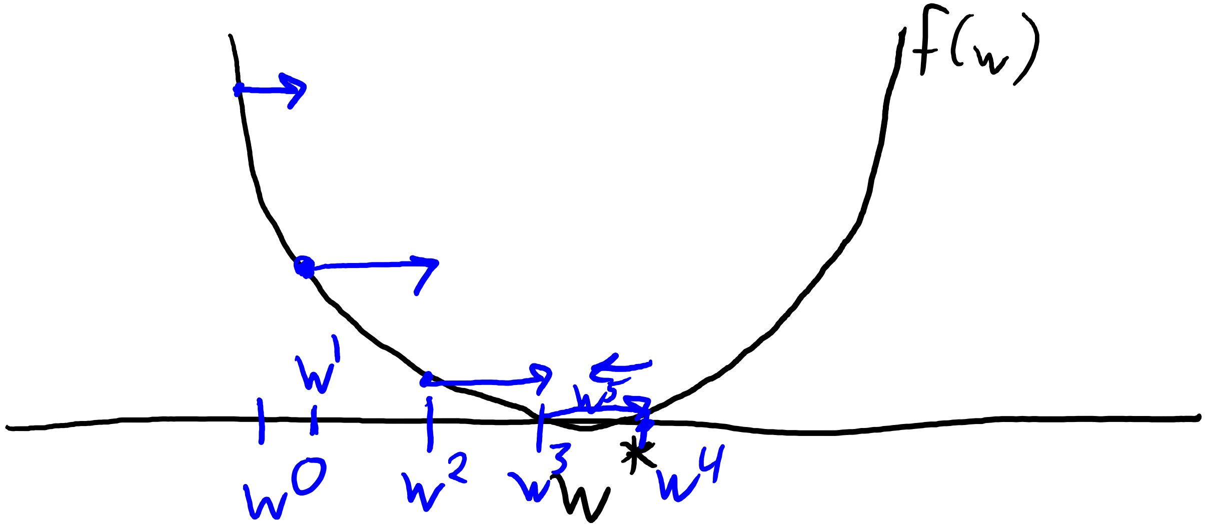


- Stochastic gradient:



Gradient Descent in Action

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



Stochastic Gradient in Action

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

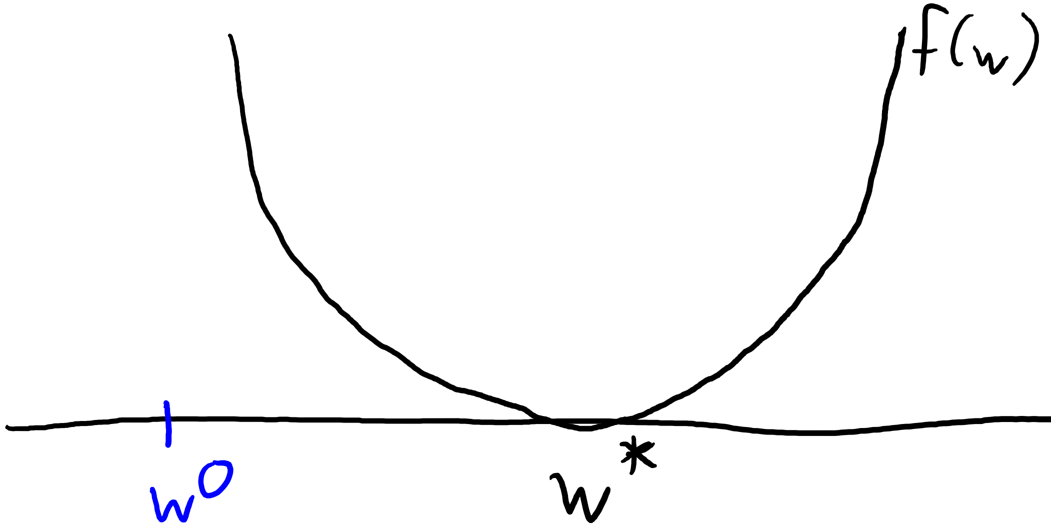
$$f_1(w) = (w^T x_1 - y_1)^2$$

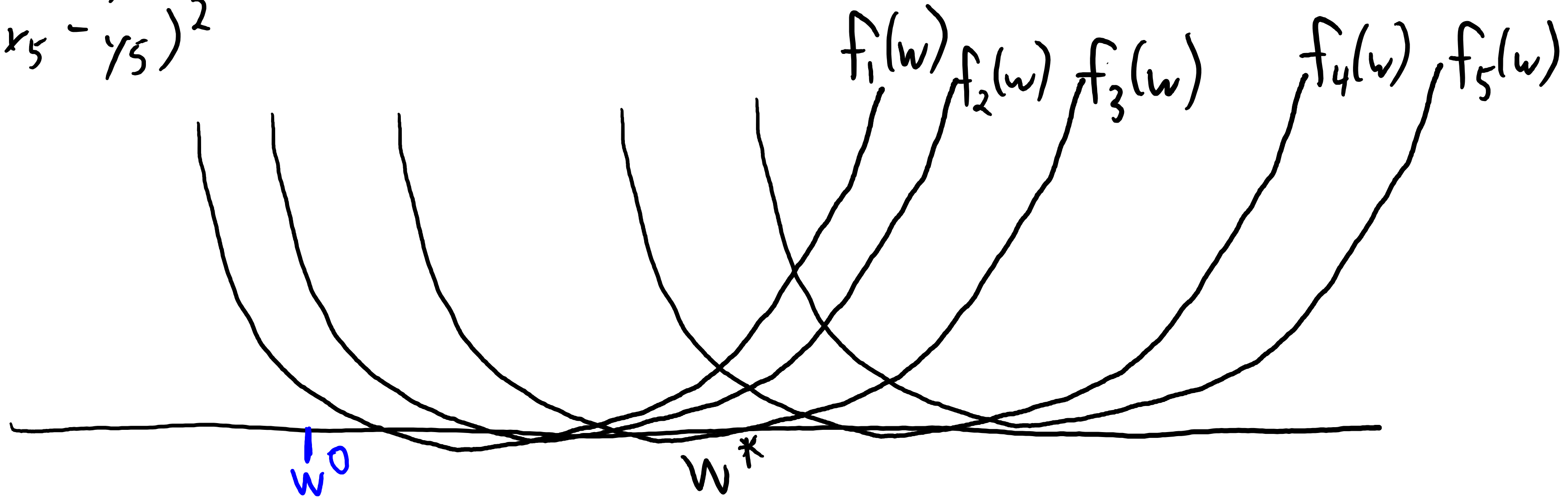
$$f_2(w) = (w^T x_2 - y_2)^2$$

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

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

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





Stochastic gradient minimizes average value.

Stochastic Gradient in Action

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

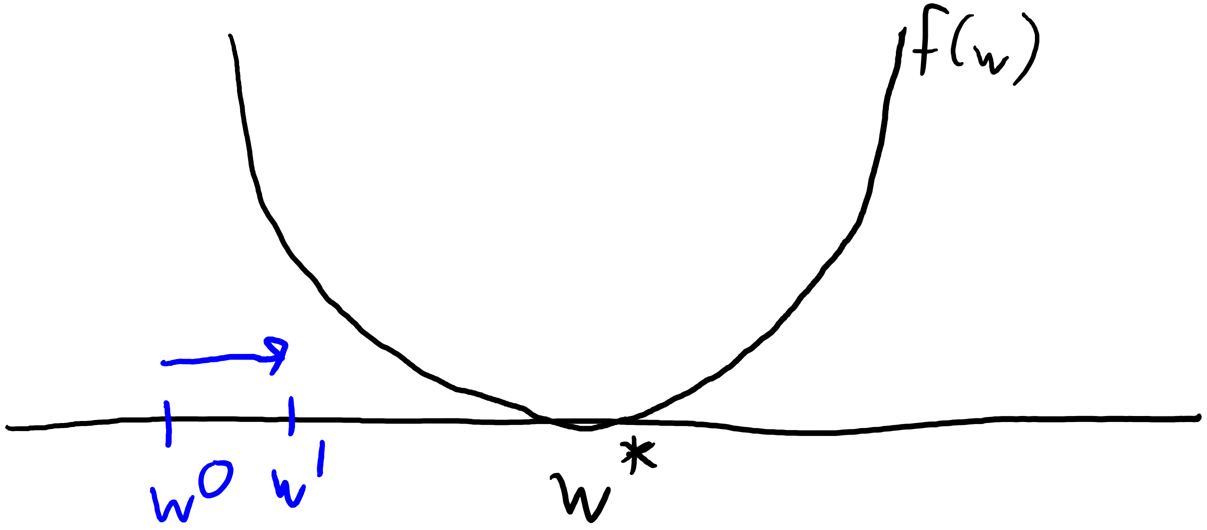
$$f_1(w) = (w^Tx_1 - y_1)^2$$

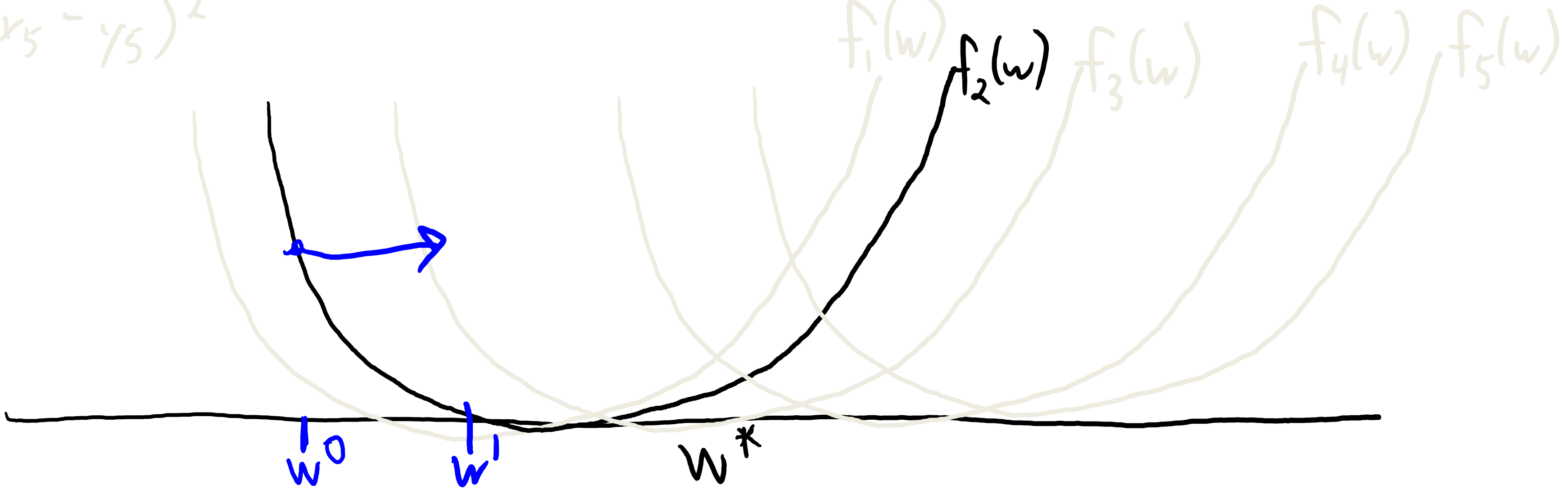
$$f_2(w) = (w^Tx_2 - y_2)^2$$

$$f_3(w) = (w^Tx_3 - y_3)^2$$

$$f_4(w) = (w^Tx_4 - y_4)^2$$

$$f_5(w) = (w^Tx_5 - y_5)^2$$





Stochastic gradient minimizes average value.

Stochastic Gradient in Action

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

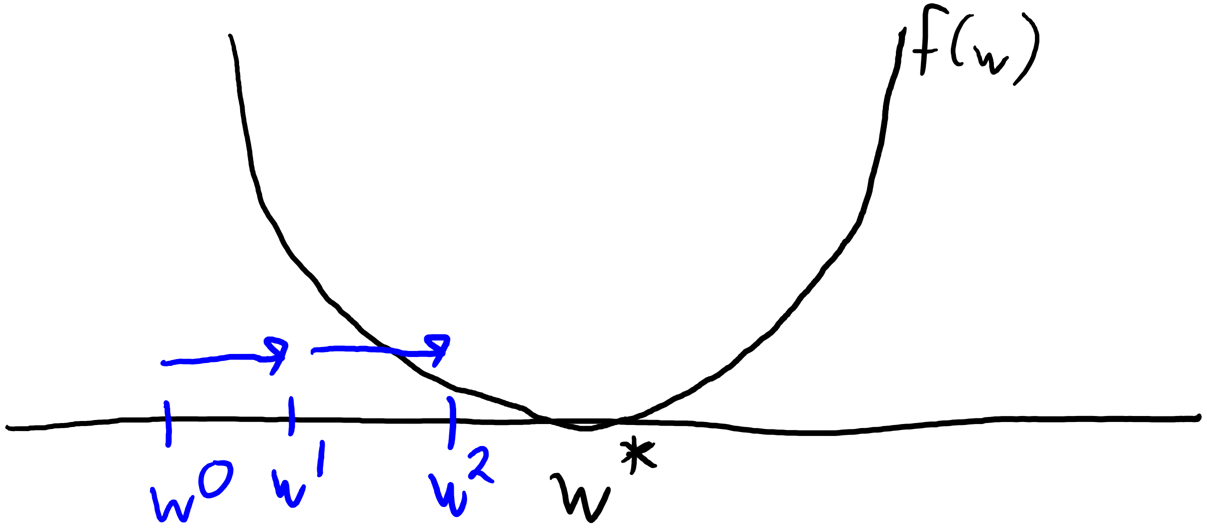
$$f_1(w) = (w^T x_1 - y_1)^2$$

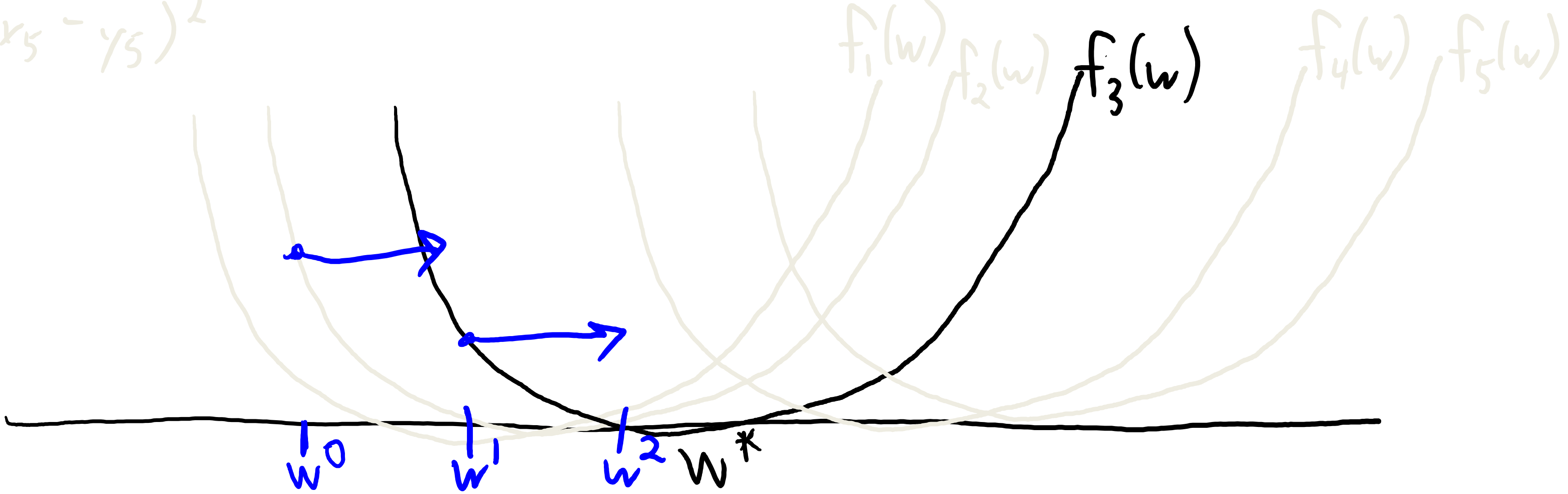
$$f_2(w) = (w^T x_2 - y_2)^2$$

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

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

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





Stochastic gradient minimizes average value.

Stochastic Gradient in Action

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

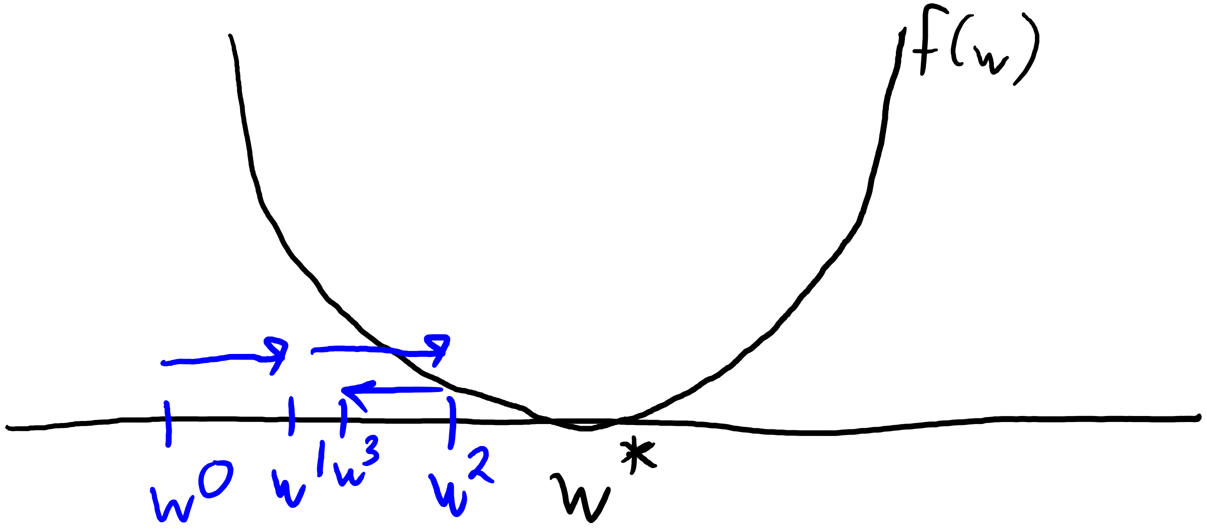
$$f_1(w) = (w^Tx_1 - y_1)^2$$

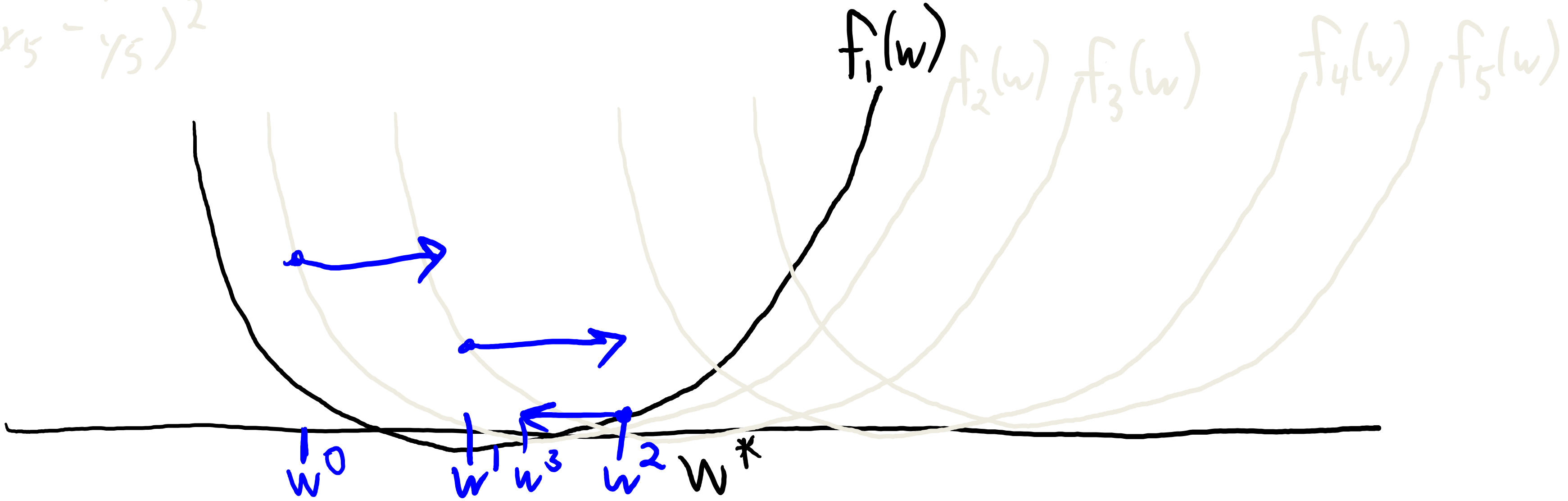
$$f_2(w) = (w^Tx_2 - y_2)^2$$

$$f_3(w) = (w^Tx_3 - y_3)^2$$

$$f_4(w) = (w^Tx_4 - y_4)^2$$

$$f_5(w) = (w^Tx_5 - y_5)^2$$





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

Stochastic Gradient in Action

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

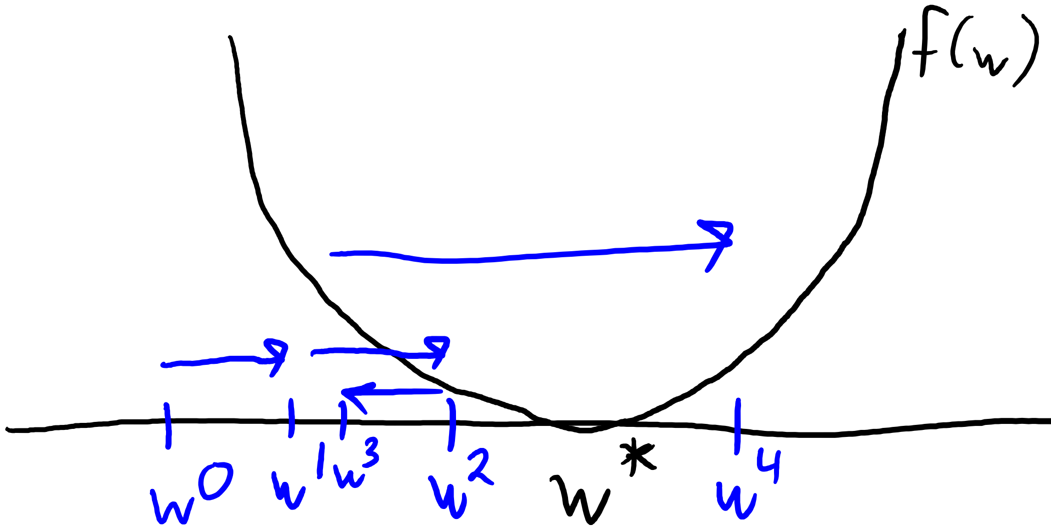
$$f_1(w) = (w^T x_1 - y_1)^2$$

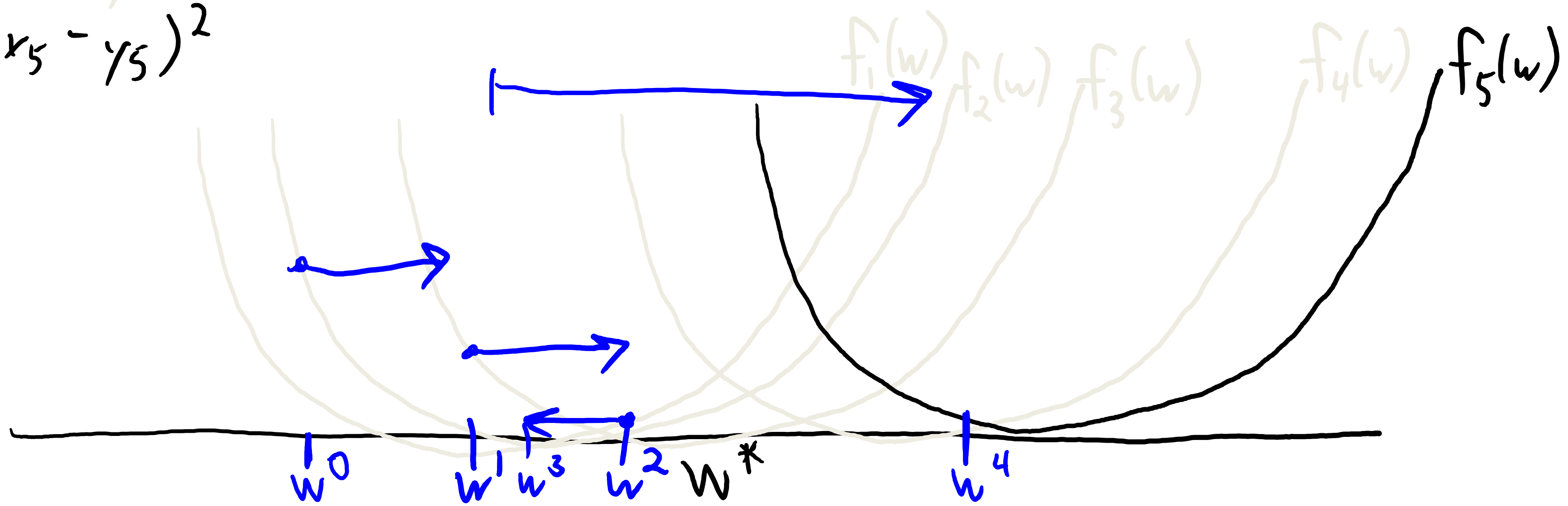
$$f_2(w) = (w^T x_2 - y_2)^2$$

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

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

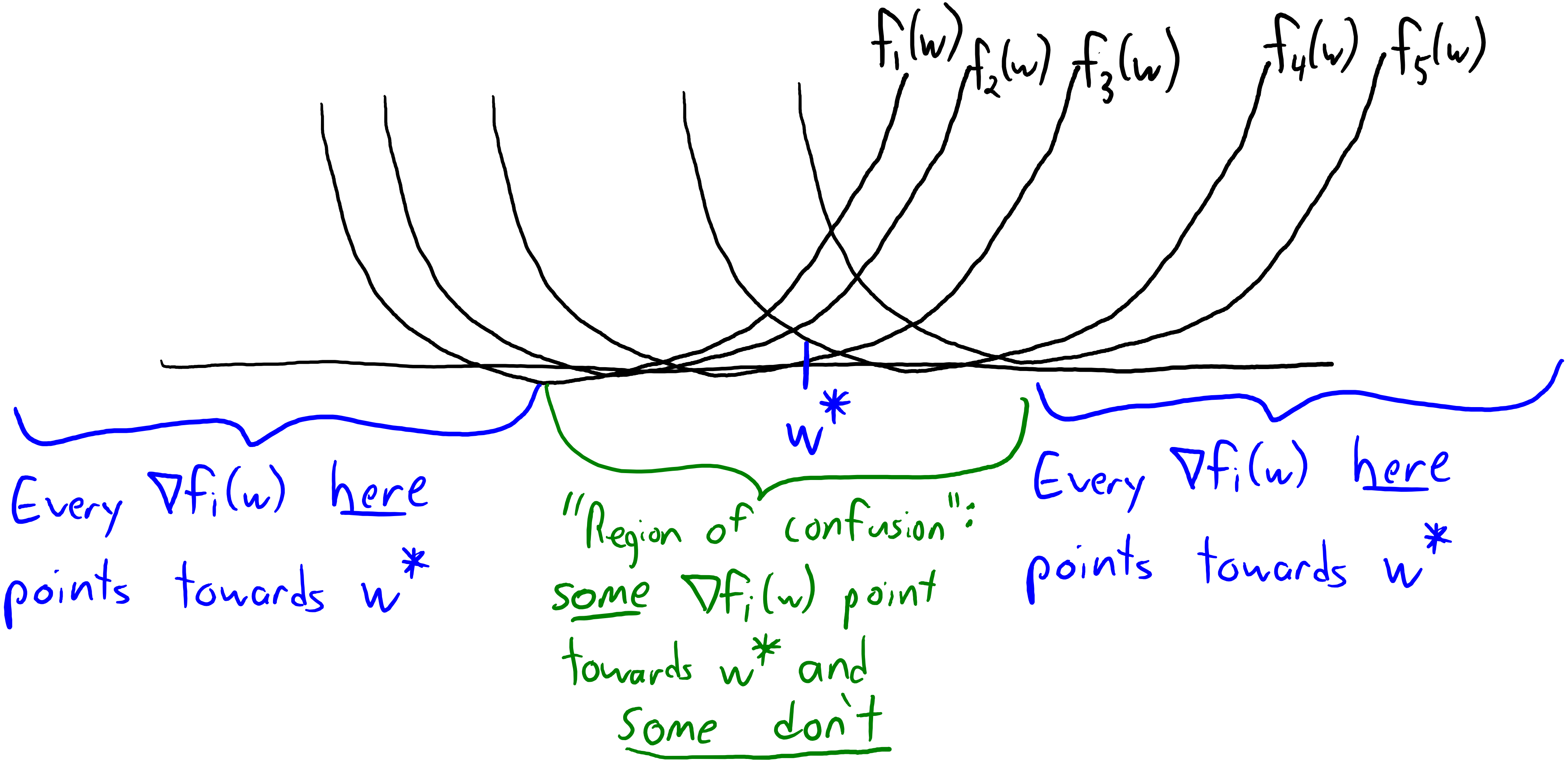
$$f_5(w) = (w^T x_5 - y_5)^2$$





Stochastic gradient minimizes average value.

Effect of ‘w’ Location on Progress


$f_1(w)$
$f_2(w)$
$f_3(w)$
$f_4(w)$
$f_5(w)$
w^*
Every $\nabla f_i(w)$ here points towards w^*
"Region of confusion": some $\nabla f_i(w)$ point towards w^* and some don't
Every $\nabla f_i(w)$ here 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:

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

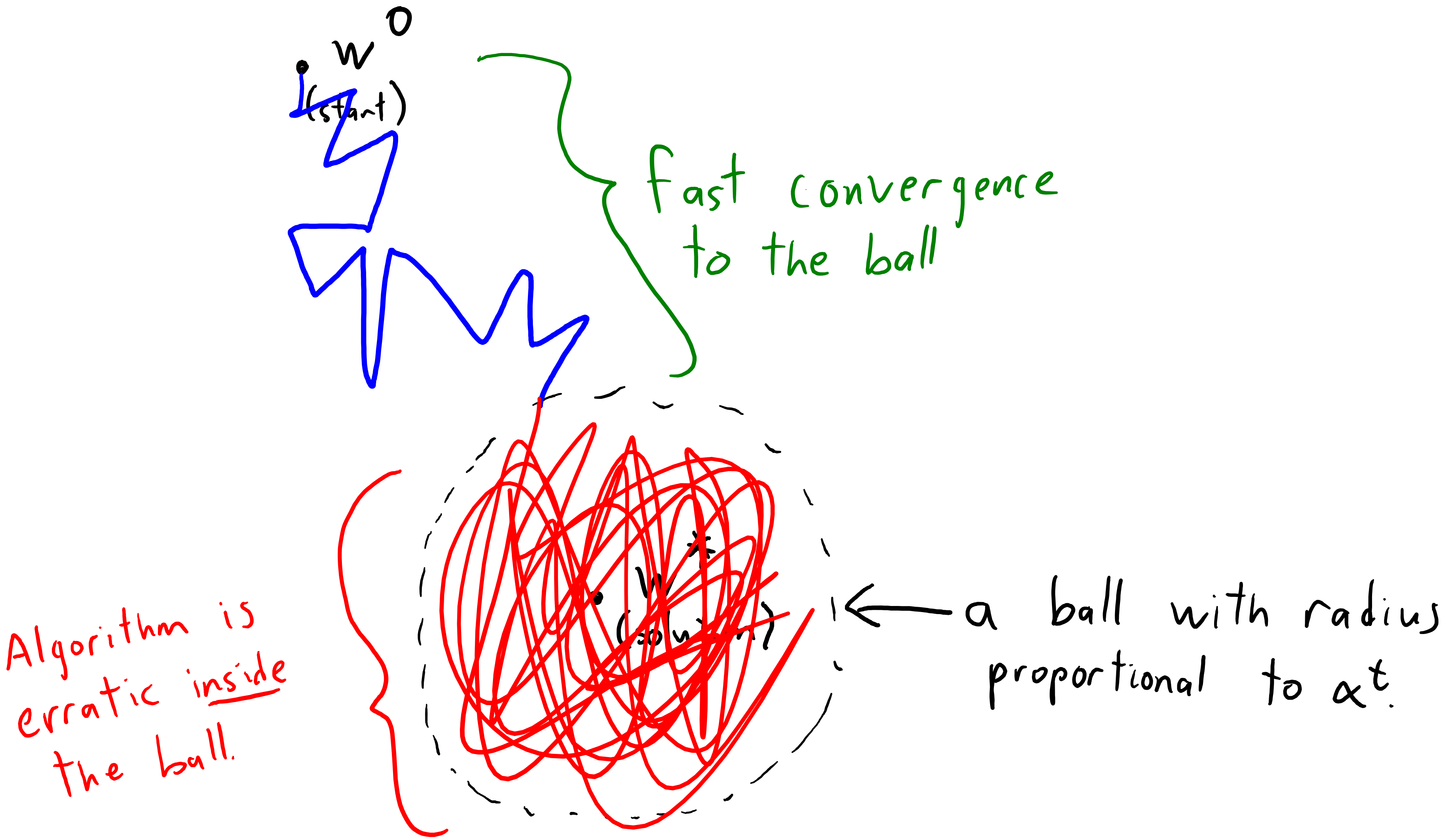
- If the variance is 0, every step goes in the right direction.
  - We're outside of region of confusion.
- If the variance is small, most steps point in the direction.
  - We're just inside region of confusion.
- If the variance is large, many steps will point in the wrong direction.
  - Middle of region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

Effect of the Step-Size

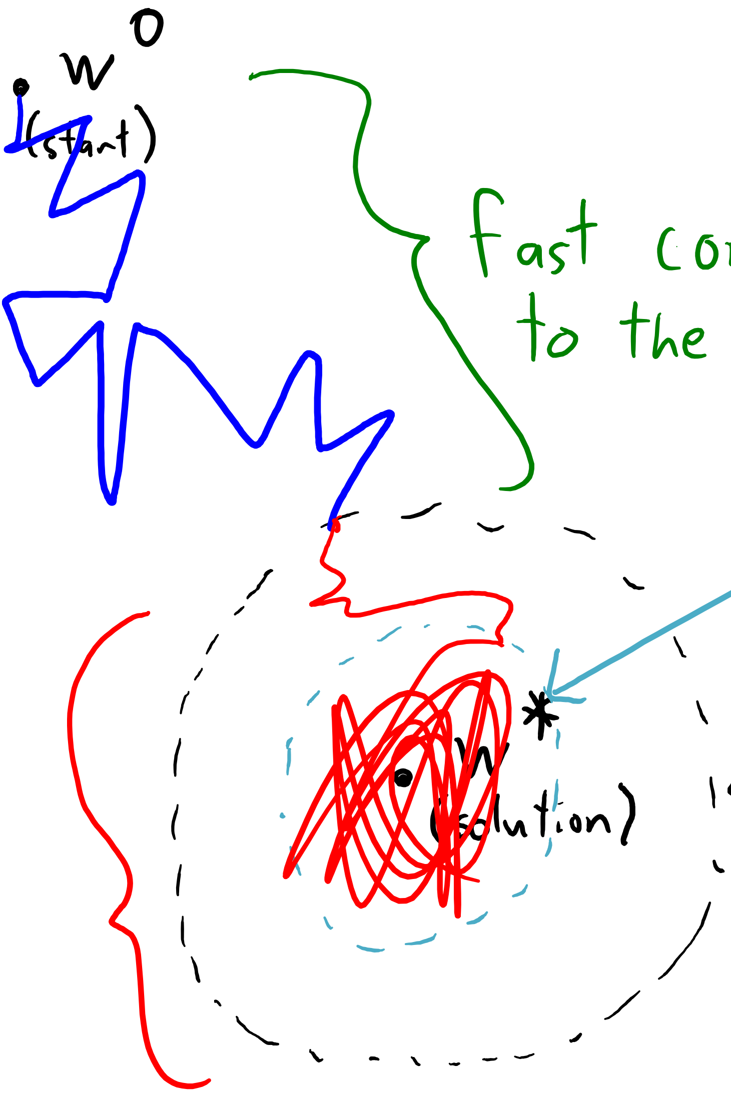
- We can reduce the effect of the variance with the step size.
  - Variance slows progress by amount proportional to square of step-size.
  - So as the step size gets smaller, the variance has less of an effect.

- For a fixed step-size, SG 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
w^*
(solution)
Algorithm is erratic inside the ball
a ball with radius proportional to α^t.

Stochastic Gradient with Constant Step Size



fast convergence to the ball

We can divide the radius of ball in 2 by dividing α^t by 2.

a ball with radius proportional to α^t.

Algorithm is erratic inside the ball

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.
- Classic solution to this problem is step-sizes α^t satisfying:

$$\underbrace{\sum_{t=1}^{\infty} \alpha^t = \infty}_{\text{"we can get everywhere"}} \qquad \underbrace{\sum_{t=1}^{\infty} (\alpha^t)^2 < \infty}_{\text{"effect of variance goes to zero"}}$$

- We can achieve this by using a step-size sequence like $\alpha^t = O(1/t)$.
  - E.g., $\alpha^t = .001/t$.

Stochastic Gradient Methods in Practice

- Unfortunately, setting $\alpha^t = O(1/t)$ works badly in practice:
  - Initial steps can be very large.
  - Later steps get very tiny.
- Practical tricks:
  - Some authors add extra parameters like $\alpha^t = \gamma/(t + \Delta)$.
  - Theory and practice support using steps that go to zero more slowly:

$$\alpha^t = O(1/\sqrt{t}) \quad \text{or} \quad \alpha^t = O(1) \quad (\text{constant})$$

    - But return a weighted average of the iterations:

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

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

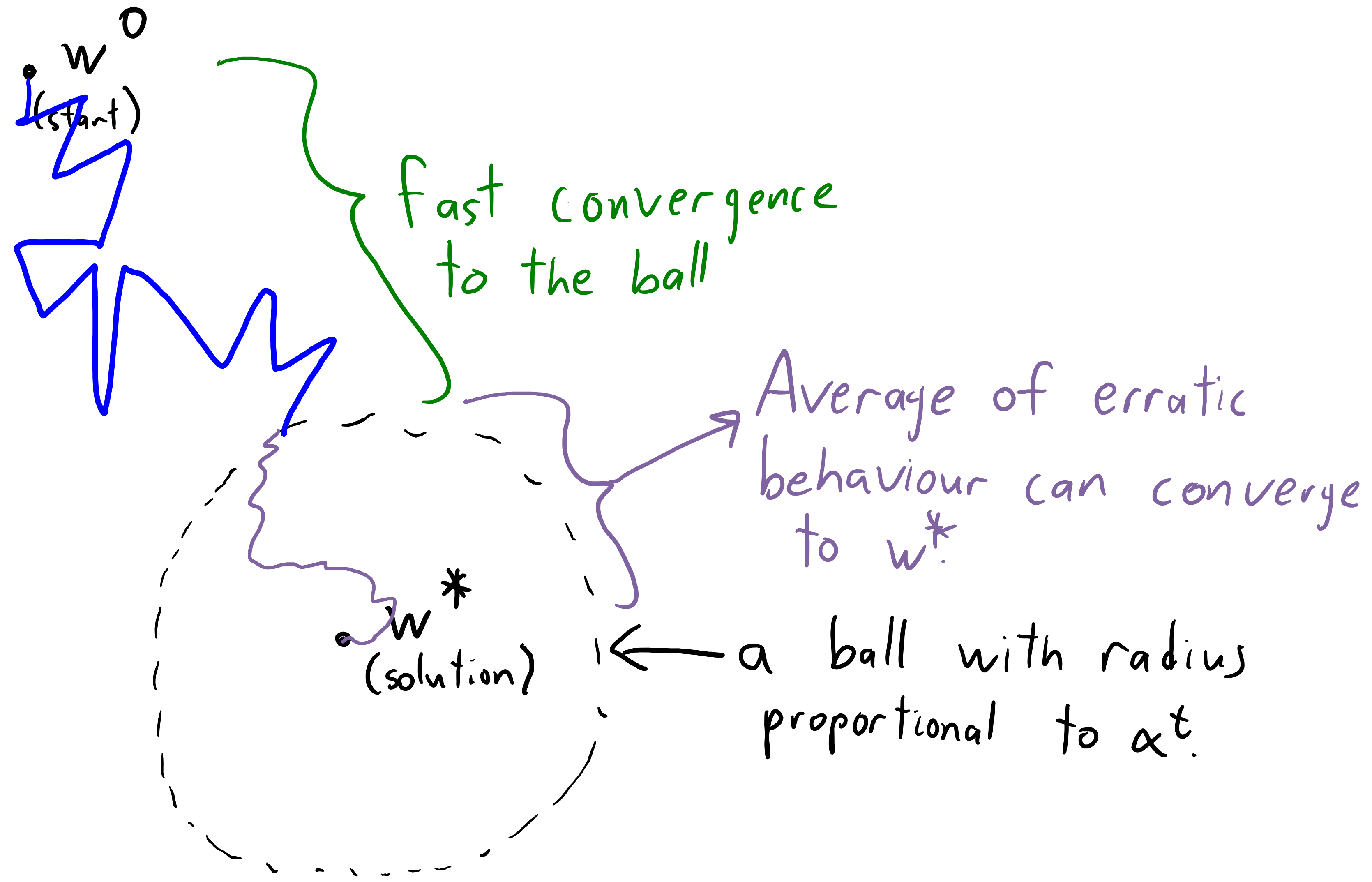
Uniform average: $\bar{w}^t = \frac{1}{n}\sum_{k=1}^{t} w^k$

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

Stochastic Gradient with Averaging

Often, you average the second half of the iterations.

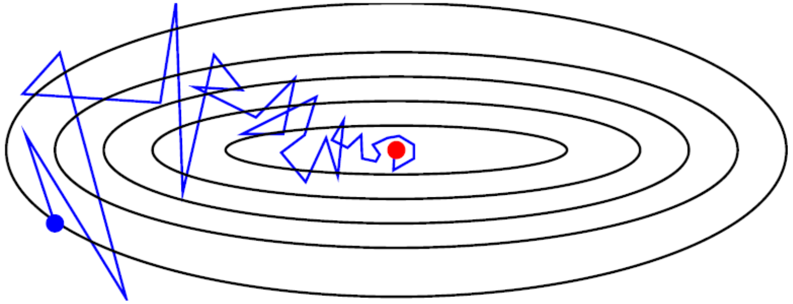
Set $\bar{w}^t = \frac{1}{t/2} \sum_{k=t/2}^{t} w^k$


w^0
(start)
fast convergence to the ball
Average of erratic behaviour can converge to w^*.
w^*
(solution)
a ball with radius proportional to α^t.

Summry: Stochastic Gradient

- Stochastic gradient minimizes average of smooth functions:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$



  - Function $f_i(w)$ is error for example 'i'.
- Iterations perform gradient descent on one random example 'i':

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

  - Cheap iterations even when 'n' is large, but doesn't always decrease 'f'.
  - But solves problem if α^t goes to 0 at an appropriate rate.
    - Theory says use $\alpha^t = O(1/t)$, in practice you need to experiment.

Summary: Stochastic Gradient

- Stochastic gradient converges very slowly:
  - But if your dataset is too big, there may not be much you can do.

- Practical tricks to improve performance:
  - Constant or slowly-decreasing step-sizes and/or average the w^t.
  - Binary search for step size, stop using validation error (bonus slides).

- You can also improve performance by reducing the variance:
  - Using "mini-batches" or random samples rather than 1 random sample.
  - New "variance-reduced" methods (SAG, SVRG) for finite training sets.

Summary

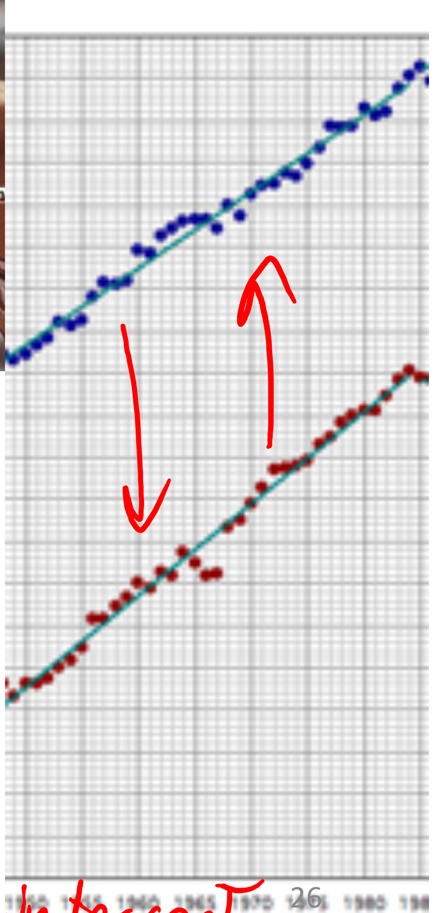
- Global vs. local features allow “personalized” predictions.
- Stochastic gradient methods let us use huge datasets.
- Step-size in stochastic gradient is a huge pain:
  - Needs to go to zero to get convergence, but this works badly.
  - Constant step-size works well, but only up to a certain point.
- SAG and other newer methods fix convergence for finite datasets.

Linear Models with Binary Features

- What is the effect of a binary feature on linear regression?

Year	Gender
1975	1
1975	0
1980	1
1980	0

Height
1.85
2.25
1.95
2.30



- Adding a bias β, our linear model is:

$$height = \beta + w_1 * year + w_2 * gender$$

- The ‘gender’ variable causes a change in y-intercept:

$$\text{If } gender == 0 \text{ then } height = \beta + w_1 * year$$

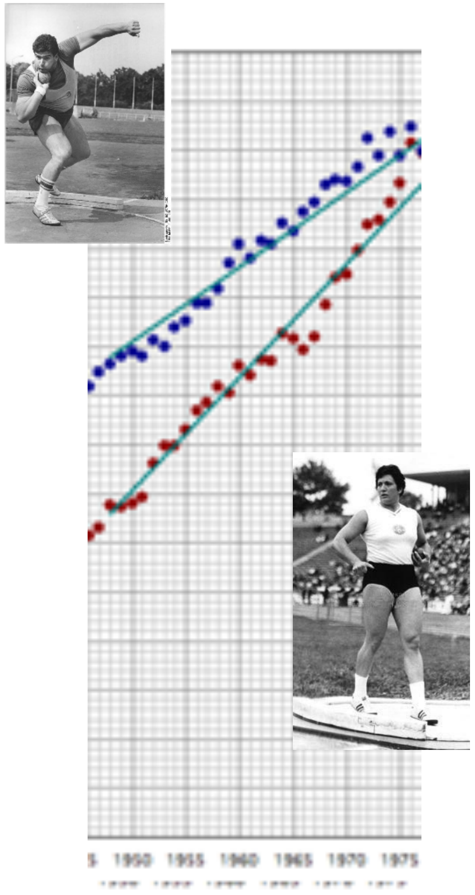
$$\text{If } gender == 1 \text{ then } height = \beta + w_1 * year + w_2$$

new y-intercept

http://www.medalinframe.com/athletes/sara-simeoni/
http://www.at-a-lanta.nl/weia/Progressie.html

Linear Models with Binary Features

- What if different genders have different slopes?
  - You can use gender-specific features.



http://www.at-a-lanta.nl/weia/Progressie.html
http://www.wikiwand.com/it/Udo_Beyer
http://women-s-rights.blogspot.ca/

Year	Gender
1975	1
1975	0
1980	1
1980	0

⇒

Bias (gender = 1)	Year (gender = 1)	Bias (gender = 0)	Year (gender = 0)
1	1975	0	0
0	0	1	1975
1	1980	0	0
0	0	1	1980

$$\left.\begin{array}{l} \text{distance} = \beta_1 + w_1 * \text{year} \quad (\text{if gender} = 1) \\ \text{distance} = \underset{\text{separate bias}}{\beta_2} + \underset{\text{separate slope}}{w_2} * \text{year} \quad (\text{if gender} = 0) \end{array}\right\} \text{Fitting a separate model for each gender.}$$

Linear Models with Binary Features

- To share information across genders, include a “global” version.

Year	Gender
1975	1
1975	0
1980	1
1980	0

⟹

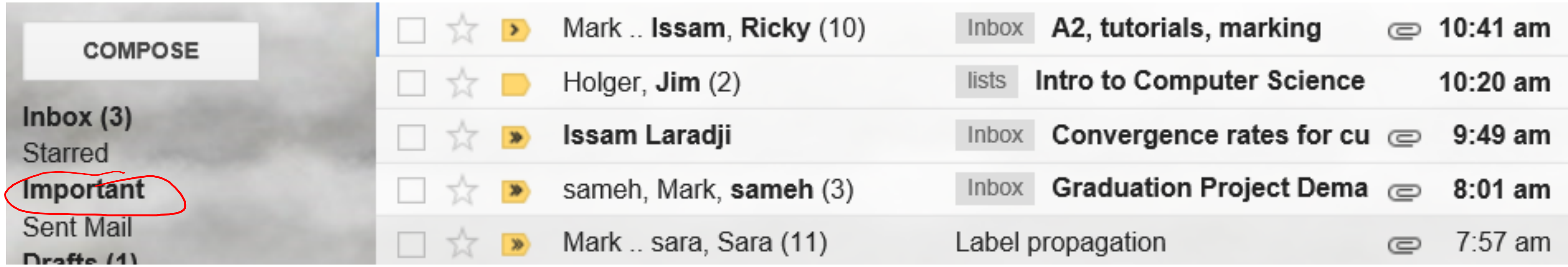
Year (any gender)	Year (if gender = 1)	Year (if gender = 0)
1975	1975	0
1975	0	1975
1980	1980	0
1980	0	1980

- “Global” year feature: influence of time on both genders.
  - E.g., improvements in technique.
- “Local” year feature: gender-specific deviation from global trend.
  - E.g., different effects of performance-enhancing drugs.

$\hat{y}_i = w_1 * year + \underline{w_2} * gender$ or $\hat{y}_i = w_1 * year + \underline{w_3} * year$

Motivation: “Personalized” Important E-mails

- Recall that we discussed identifying ‘important’ e-mails?



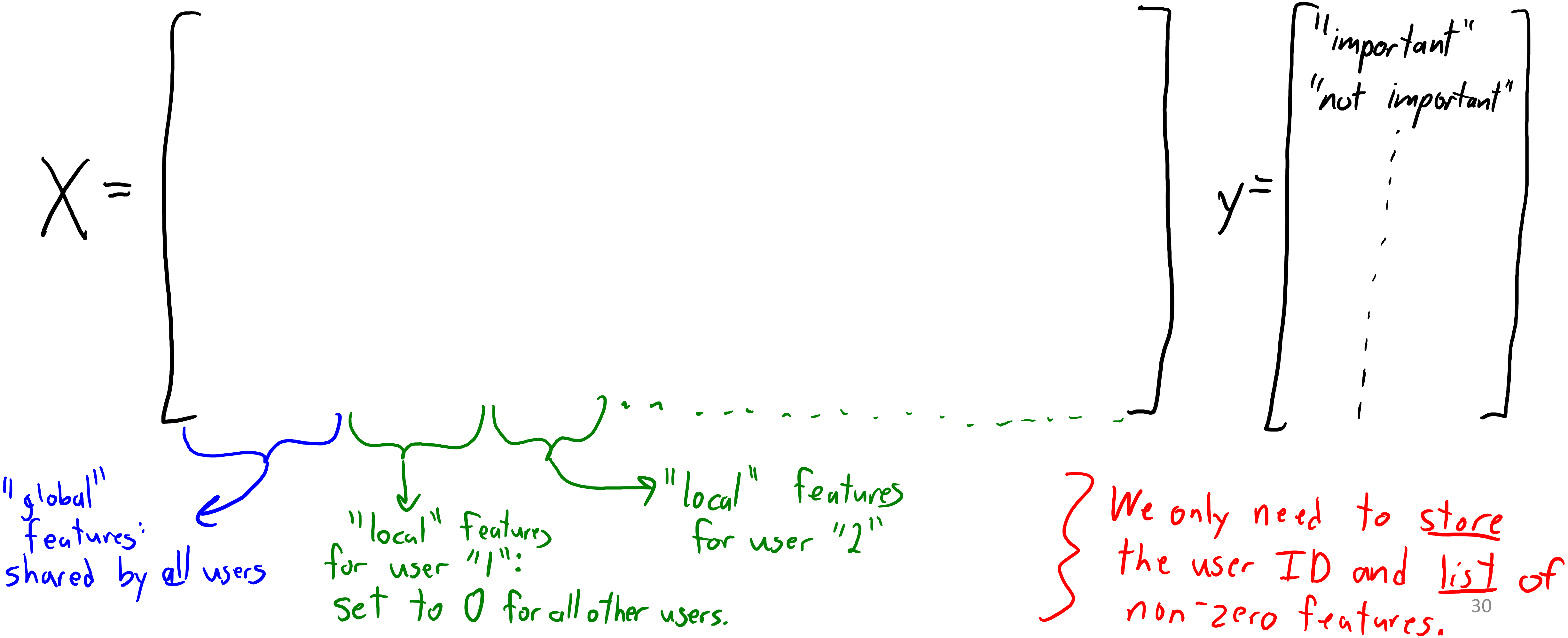
Mark .. Issam, Ricky (10)	Inbox	A2, tutorials, marking	10:41 am
Holger, Jim (2)	lists	Intro to Computer Science	10:20 am
Issam Laradji	Inbox	Convergence rates for cu	9:49 am
sameh, Mark, sameh (3)	Inbox	Graduation Project Dema	8:01 am
Mark .. sara, Sara (11)		Label propagation	7:57 am

- There might be some “globally” important messages:
  - “This is your mother, something terrible happened, give me a call ASAP.”

- But your “important” message may be unimportant to others.
  - Similar for spam: “spam” for one user could be “not spam” for another.

The Big Global/Local Feature Table for E-mails

- Each row is one e-mail (there are lots of rows):



"global" features: shared by all users

"local" features for user "1": set to 0 for all other users.

"local" features for user "2"

We only need to store the user ID and list of non-zero features.

Predicting Importance of E-mail For New User

- Consider a new user:
  - We start out with no information about them.
  - So we use global features to predict what is important to a generic user.

$$y_i = \text{sign}(\underbrace{w_g^T x_{ig}}_{}) \longrightarrow \text{features/weights } \underline{\text{shared}} \text{ across users.}$$

- With more data, update global features and user’s local features:
  - Local features make prediction *personalized*.

$$y_i = \text{sign}(w_g^T x_{ig} + \underbrace{w_u^T x_{iu}}_{}) \longrightarrow \text{features/weights } \underline{\text{specific}} \text{ to user.}$$

  - What is important to *this* user?
- G-mail system: classification with logistic regression.
  - Trained with a variant of stochastic gradient.

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:

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

- Notice that it's cheaper than O(nd) if the x_i are very sparse:
  - Each e-mail has a limited number of non-zero features,
  - Each e-mail only has "global" features and "local" features for one user.

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

Stochastic Gradient with Infinite Data

- Amazing property of stochastic gradient:
  - The classic convergence analysis does not rely on ‘n’ being finite.
- Consider an infinite sequence of IID samples.
  - Or any dataset that is so large we cannot even go through it once.
- Approach 1 (gradient descent):
  - Stop collecting data once you have a very large ‘n’.
  - Fit a regularized model on this fixed dataset.
- Approach 2 (stochastic gradient):
  - Perform a stochastic gradient iteration on each example as we see it.
  - Never re-visit any example, always take a new one.

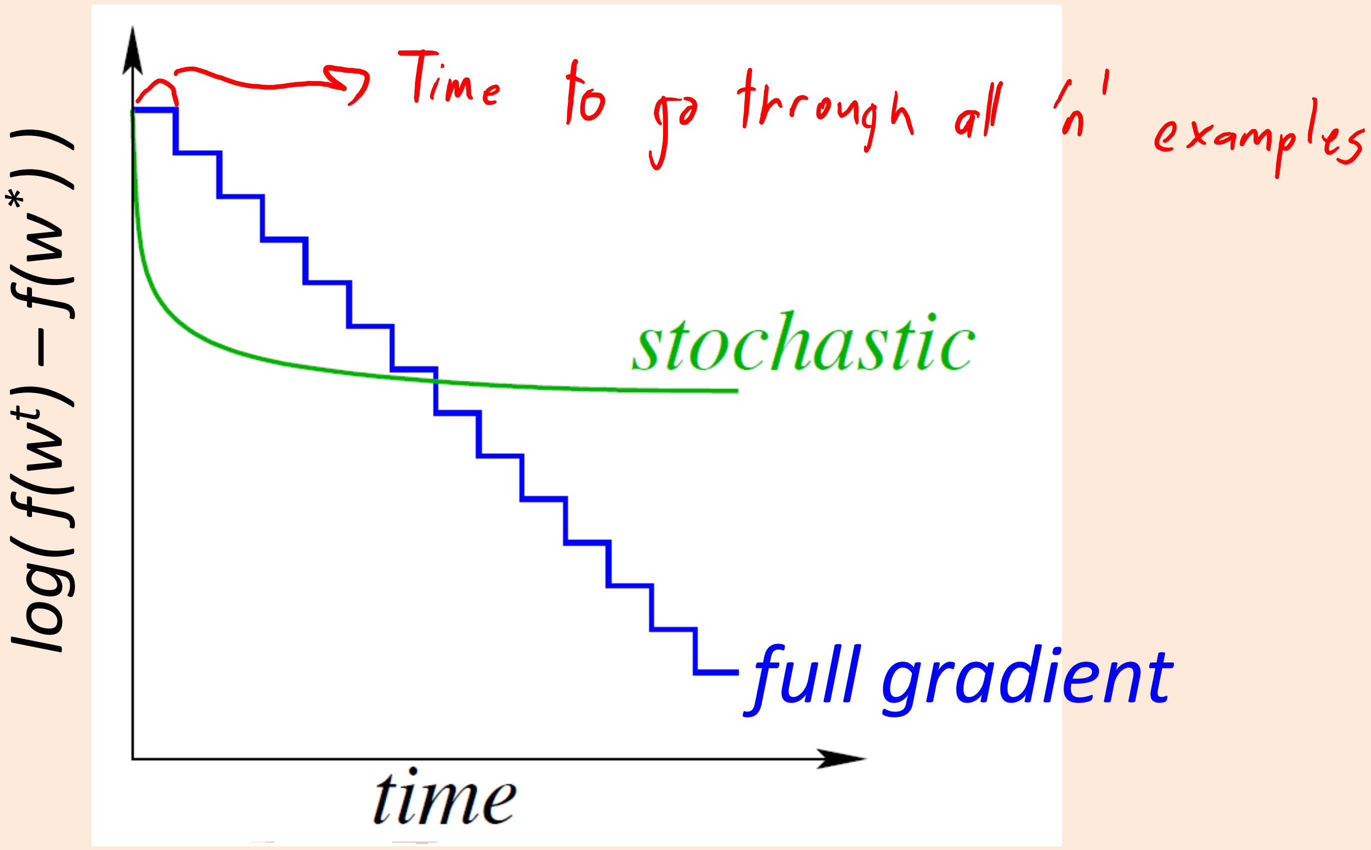
Stochastic Gradient with Infinite Data

- 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.

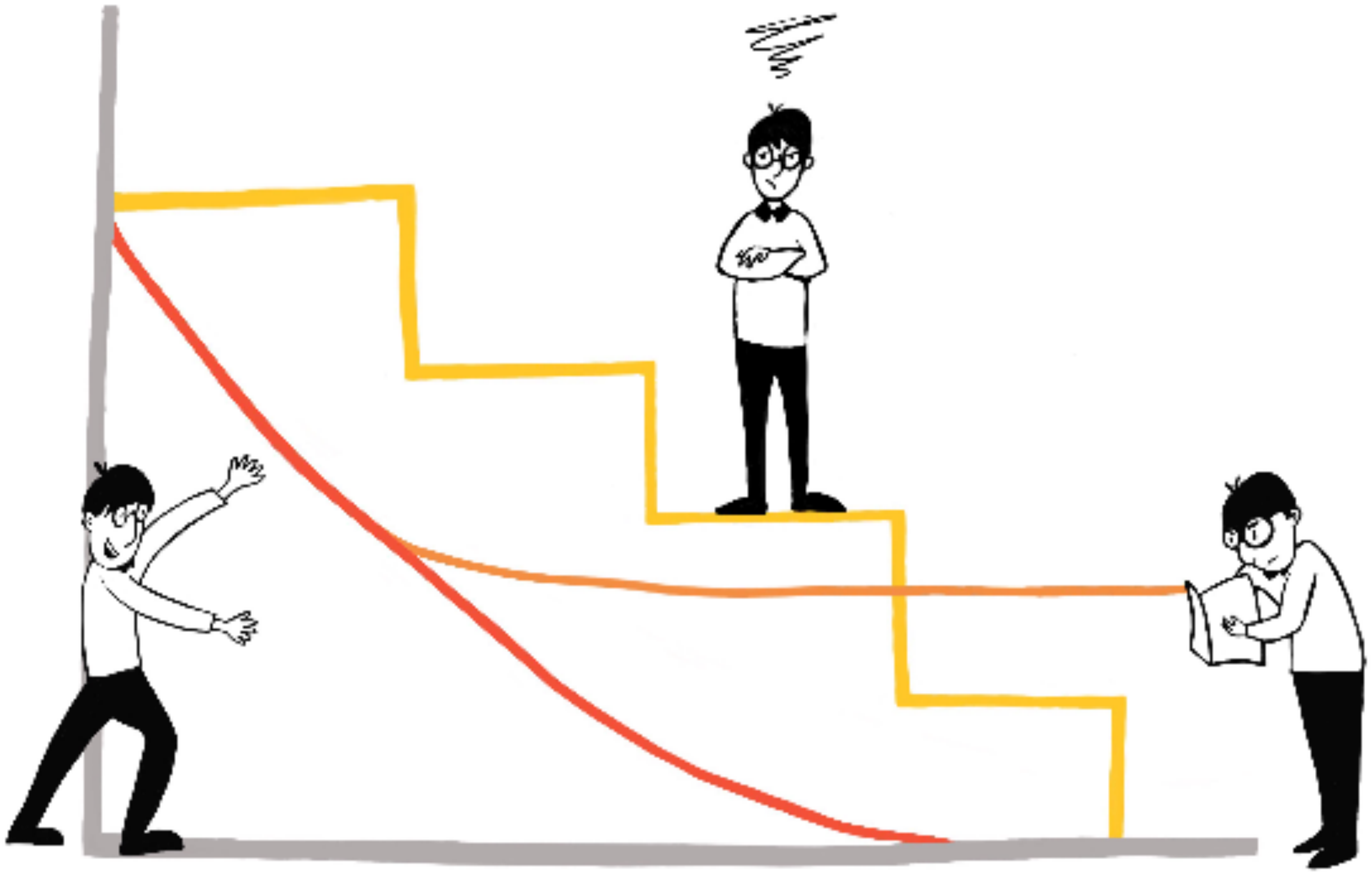
- Up to a constant, Approach 2 achieves test error of Approach 1.
  - This is sometimes used to justify SG as the “ultimate” learning algorithm.
    - “Optimal test error by computing gradient of each example once!”
  - In practice, Approach 1 usually gives lower test error.
    - The constant factor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

Gradient Descent vs. Stochastic Gradient



Time to go through all 'n' examples

- 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/

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.

A Practical Strategy for Deciding When to Stop

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

- In stochastic gradient:
  - Individual gradients don’t necessarily go to zero.
  - We can’t see full gradient, so we don’t know when to stop.

- Practical trick:
  - Every ‘k’ iterations (for some large ‘k’), measure validation set error.
  - Stop if the validation set error isn’t improving.

More Practical Issues

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

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

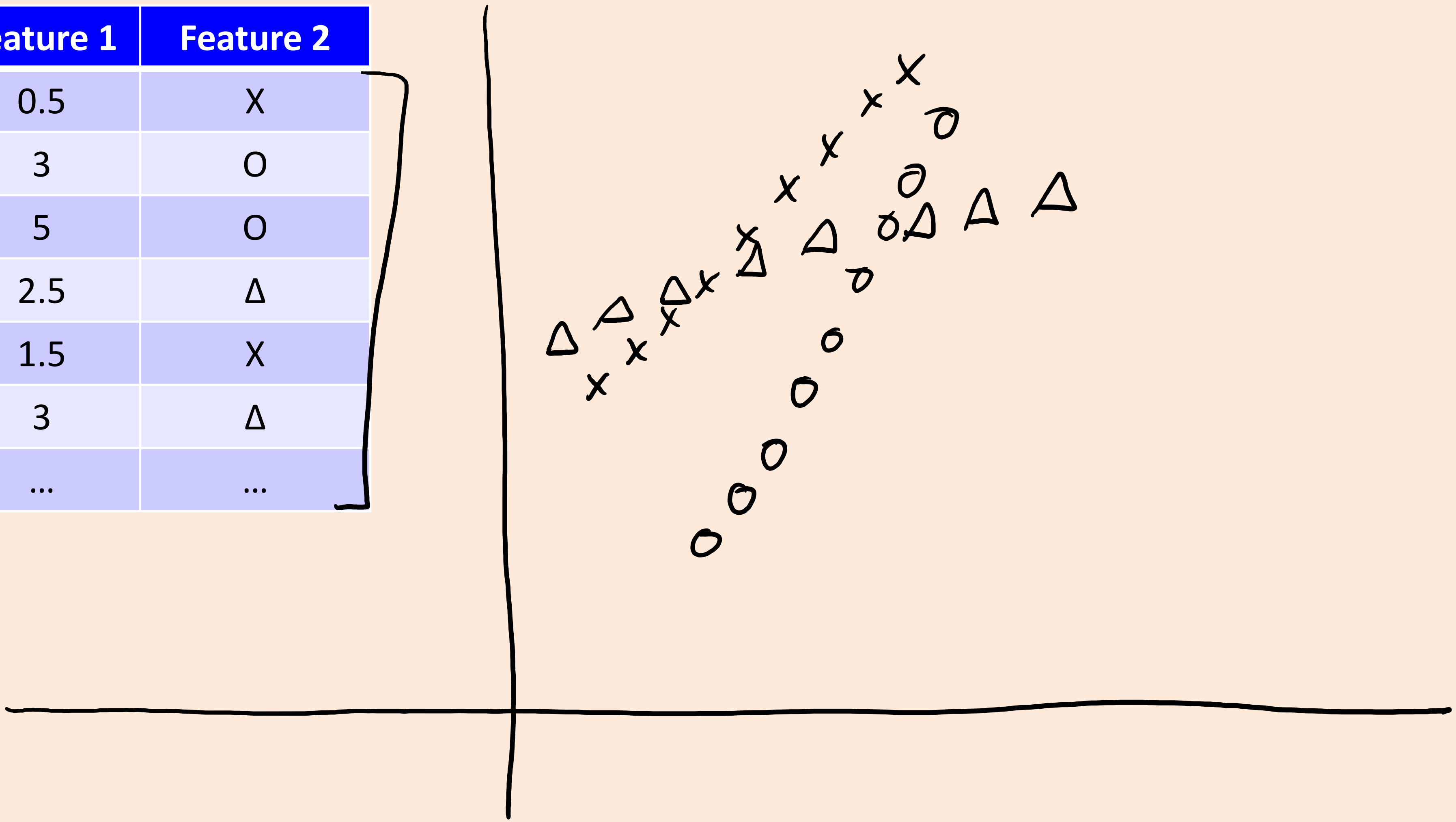
Random "batch" of examples.

  - The variance is inversely proportional to the mini-batch size.
    - You can use bigger step size as the batch size increases.
    - Big gains for going from 1 to 2, less big gains from going from 100 to 101.
  - Useful for vectorizing/parallelizing code.
    - Evaluate one gradient on each core.

Linear Models with Binary Features

X =

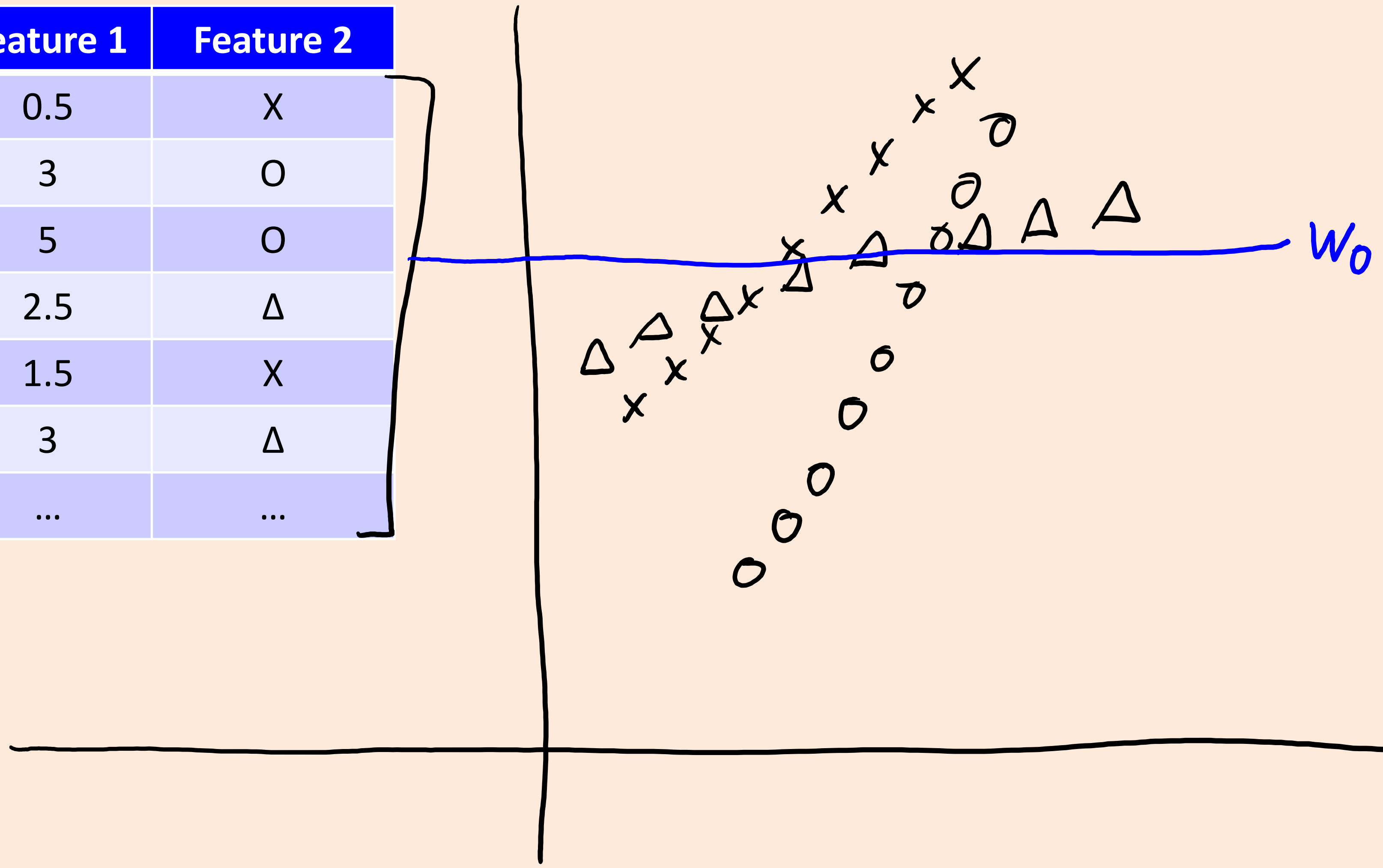
Feature 1	Feature 2
0.5	X
3	O
5	O
2.5	Δ
1.5	X
3	Δ
…	…



Linear Models with Binary Features

$X =$

Feature 1	Feature 2
0.5	X
3	O
5	O
2.5	Δ
1.5	X
3	Δ
…	…



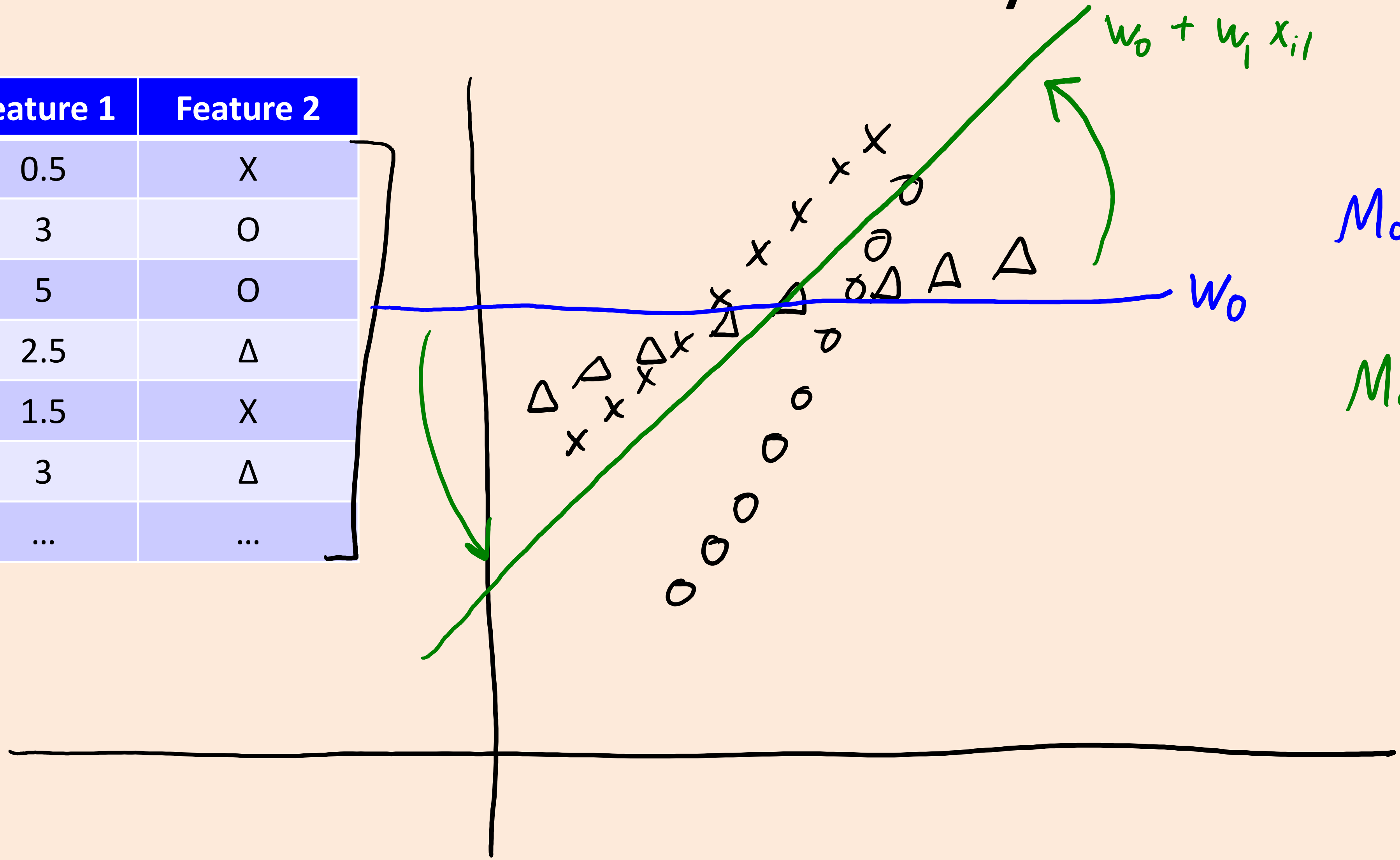
Model 1: only bias

$y_i = w_0$

Linear Models with Binary Features

$X =$

Feature 1	Feature 2
0.5	X
3	O
5	O
2.5	Δ
1.5	X
3	Δ
…	…



Model 1: only bias

$y_i = w_0$

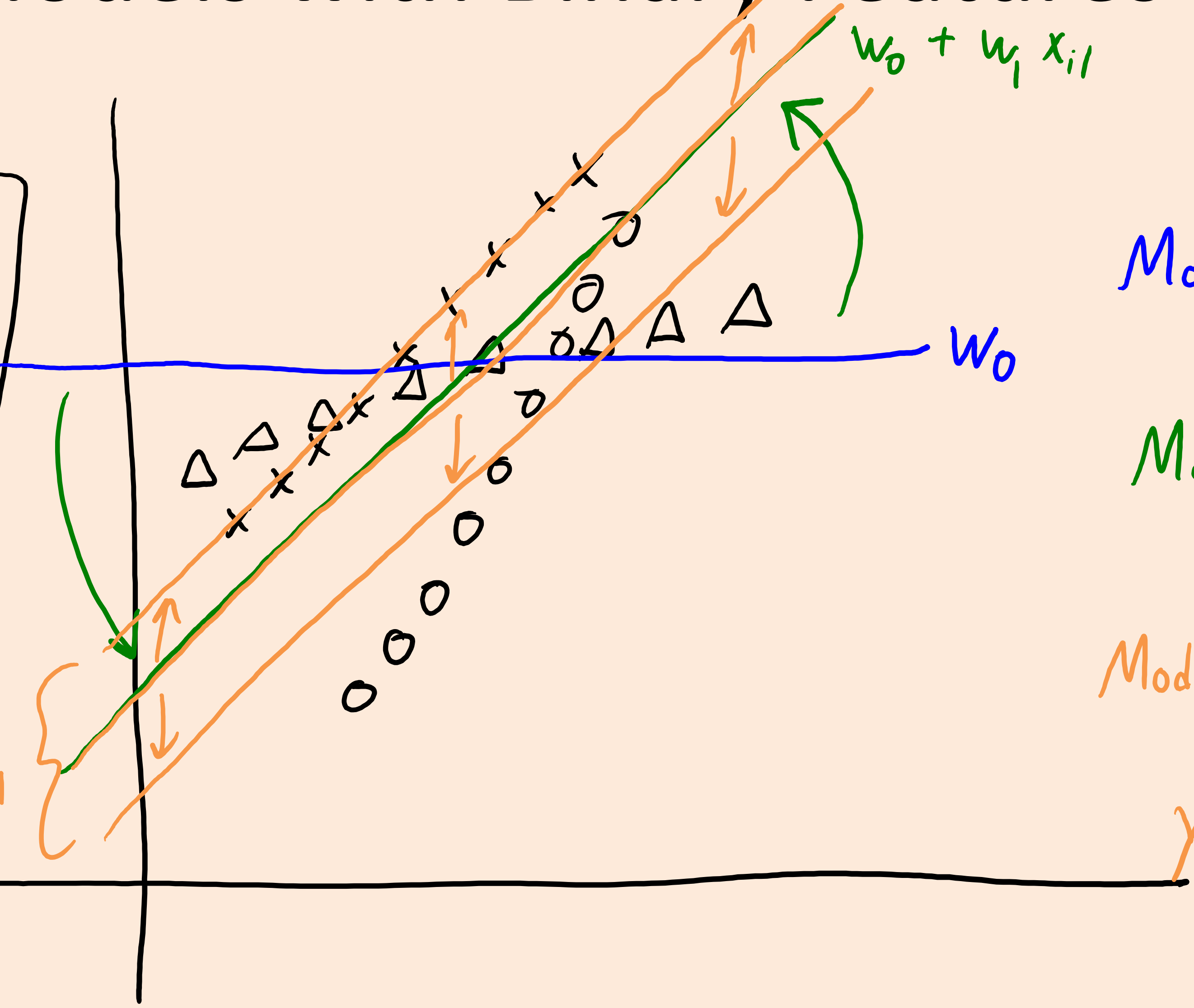
Model 2: bias + feature 1

$y_i = w_0 + w_1 x_{i1}$

Linear Models with Binary Features

$X =$

Feature 1	Feature 2
0.5	X
3	O
5	O
2.5	Δ
1.5	X
3	Δ
…	…



Model 1: only bias

$y_i = w_0$

Model 2: bias + feature 1

$y_i = w_0 + w_1 x_{i1}$

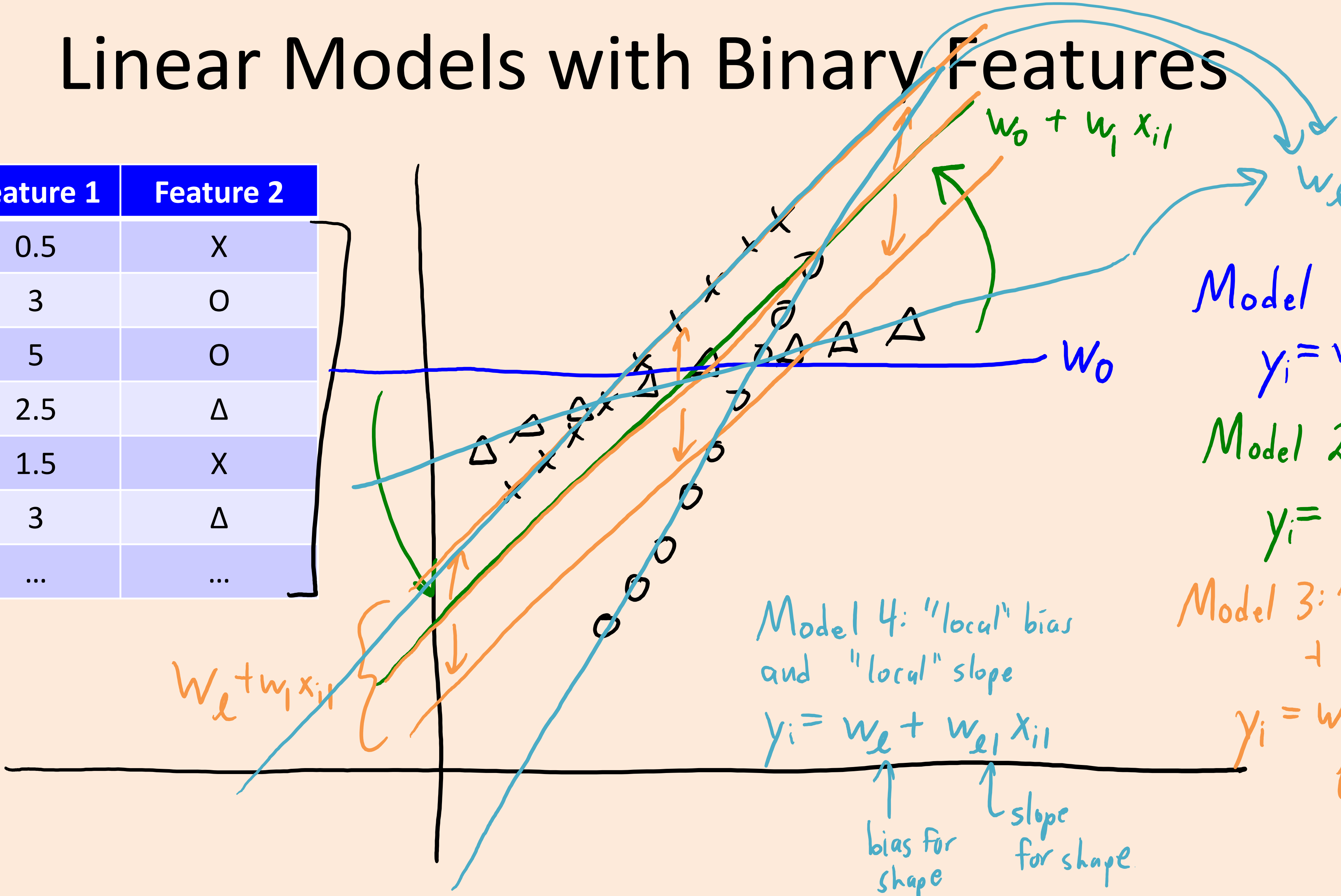
Model 3: "local" bias + feature 1

$y_i = \underset{\uparrow \text{shape}}{w_\ell} + w_1 x_{i1}$

Linear Models with Binary Features

$X =$

Feature 1	Feature 2
0.5	X
3	O
5	O
2.5	Δ
1.5	X
3	Δ
…	…



Model 1: only bias

$y_i = w_0$

Model 2: bias + feature 1

$y_i = w_0 + w_1 x_{i1}$

Model 3: "local" bias + feature 1

$y_i = \underset{\uparrow \text{shape}}{w_\ell} + w_1 x_{i1}$

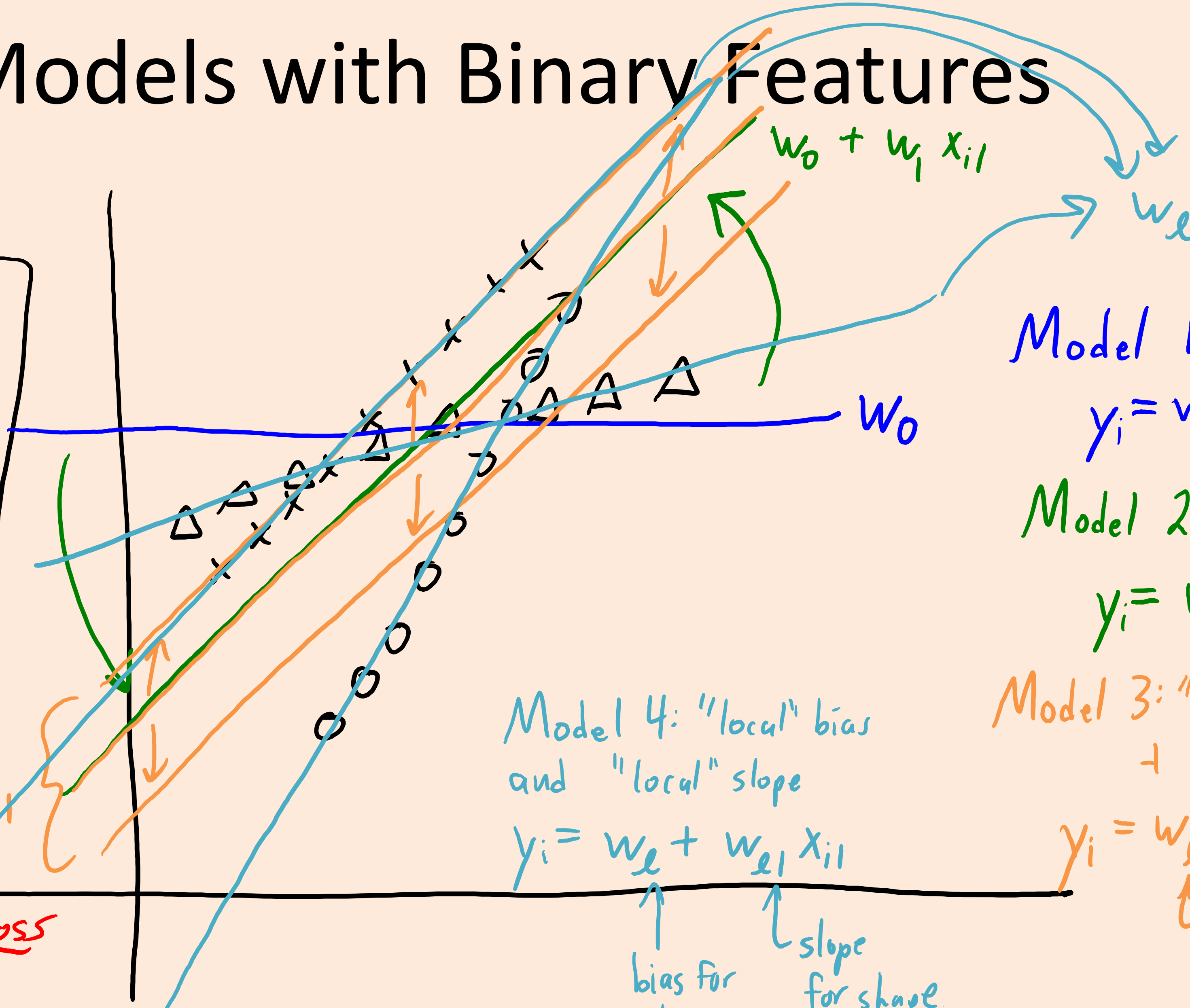
Model 4: "local" bias and "local" slope

$y_i = \underset{\uparrow \text{bias for shape}}{w_\ell} + \underset{\uparrow \text{slope for shape}}{w_{\ell 1}} x_{i1}$

Linear Models with Binary Features

$X =$

Feature 1	Feature 2
0.5	X
3	O
5	O
2.5	Δ
1.5	X
3	Δ
…	…



Model 1: only bias

$y_i = w_0$

Model 2: bias + feature 1

$y_i = w_0 + w_1 x_{i1}$

Model 3: "local" bias + feature 1

$y_i = \underset{\text{shape}}{w_\ell} + w_1 x_{i1}$

Model 4: "local" bias and "local" slope

$y_i = \underset{\text{bias for shape}}{w_\ell} + \underset{\text{slope for shape}}{w_{\ell 1}} x_{i1}$

Could also share information across categories with global bias slope:

$y_i = w_0 + w_1 x_{i1} + w_\ell + w_{\ell 1} x_{i\ell}$