

# CPSC 340: Machine Learning and Data Mining

Regularization

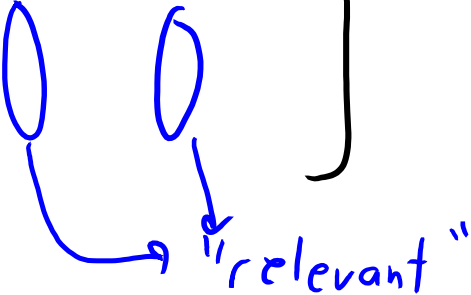
Fall 2019

# Admin

- **Midterm** is Thursday.
  - October 17<sup>th</sup> at 6:30pm.
  - Last names starting with A-L: West Mall Swing Space Room 121.
  - Last names starting with M-Z: West Mall Swing Space Room 221.
  - 80 minutes.
  - Closed-book.
  - One doubled-sided 'cheat sheet'.
  - Auditors do not take the midterm.
- There will be **two types of questions on the midterm**:
  - 'Technical' questions requiring things like pseudo-code or derivations.
    - Similar to assignment questions, and will only be on topics related to those in assignments.
  - 'Conceptual' questions testing understanding of key concepts.
    - All lecture slide material except "bonus slides" is fair game here.

# Last Time: Feature Selection

- Last time we discussed **feature selection**:
  - Choosing set of “relevant” features.

$$X = \begin{bmatrix} \text{ } & \text{ } \end{bmatrix} \quad y = \begin{bmatrix} \text{ } \end{bmatrix}$$


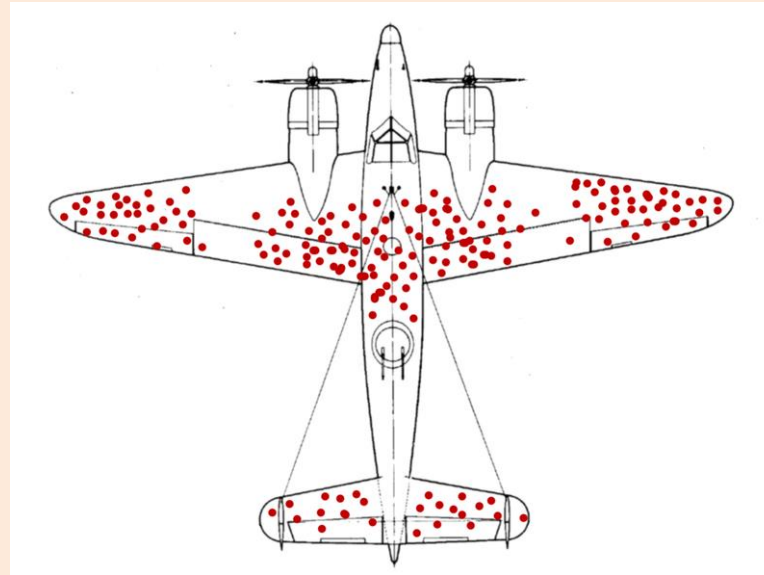
- Most common approach is **search and score**:
  - Define “score” and “search” for features with best score.
- But it's **hard to define the “score” and it's hard to “search”**.
  - So we often use greedy methods like **forward selection**.
- Methods work ok on “toy” data, but are **frustrating on real data**.
  - Different methods may return very different results.
  - Defining whether a feature is “relevant” is complicated and ambiguous.

# My advice if you want the “relevant” variables.

- Try the **association approach**.
- Try **forward selection with different values of  $\lambda$** .
- Try out a few other feature selection methods too.
- **Discuss the results** with the domain expert.
  - They probably have an idea of why some variables might be relevant.
- **Don't be overconfident:**
  - These methods are probably not discovering how the world truly works.
  - “The algorithm has found that these variables are helpful in predicting  $y_i$ .”
    - Then a warning that these models are not perfect at finding relevant variables.

# Related: Survivorship Bias

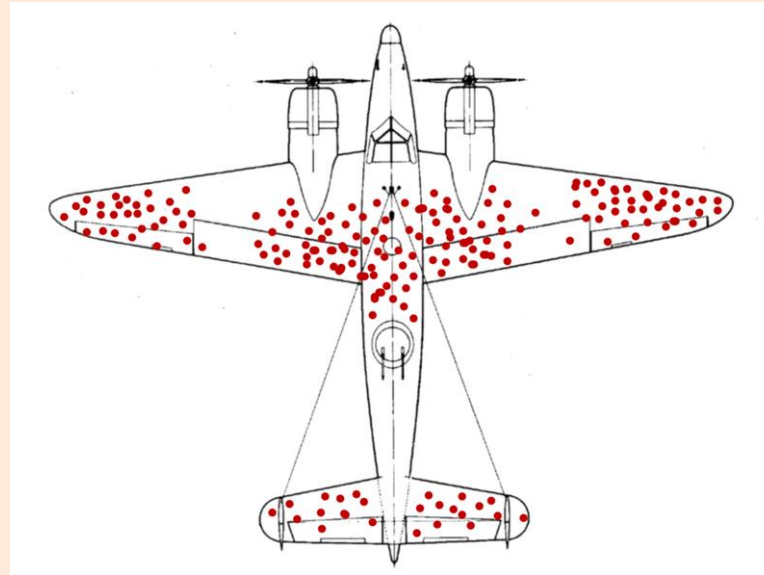
- Plotting location of bullet holes on planes returning from WW2:



- Where are the “relevant” parts of the plane to protect?
  - “Relevant” parts are actually **where there are no bullets.**
  - **Planes shot in other places did not come back** (armor was needed).

# Related: Survivorship Bias

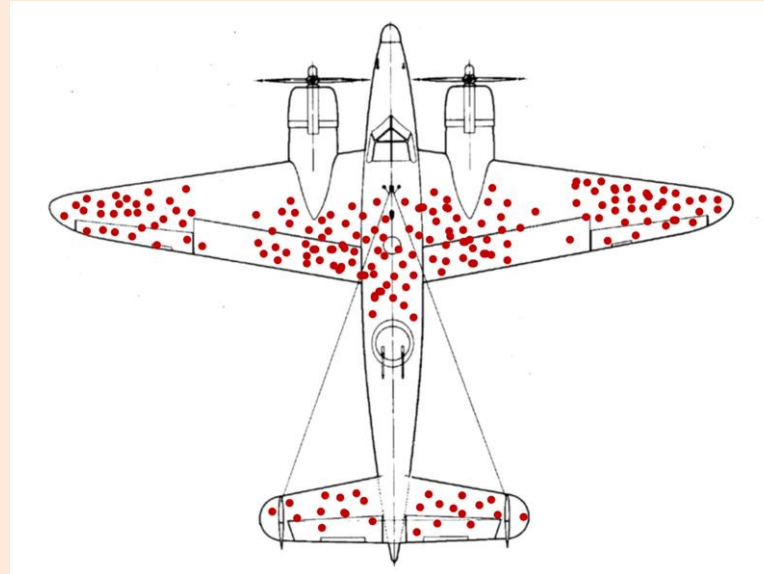
- Plotting location of bullet holes on planes returning from WW2:



- This is an example of “survivorship bias”:
  - Data is not IID because you only sample the “survivors”.
  - Causes havoc for feature selection, and ML methods in general.

# Related: Survivorship Bias

- Plotting location of bullet holes on planes returning from WW2:



- People come to **wrong conclusions due to survivor bias** all the time.
  - Article on “secrets of success”, focusing on traits of successful people.
  - But ignoring the number of non-super-successful with the same traits.
  - [Article](#) hypothesizing about various topics (allergies, mental illness, etc.).

# “Feature” Selection vs. “Model” Selection?

- **Model selection**: “which model should I use?”
  - KNN vs. decision tree, depth of decision tree, **degree of polynomial basis**.
- **Feature selection**: “which features should I use?”
  - Using feature 10 or not, **using  $x_i^2$  as part of basis**.
- These two tasks are **highly-related**:
  - It’s a different “model” if we add  $x_i^2$  to linear regression.
  - But the  $x_i^2$  term is just a “feature” that could be “selected” or not.
  - Usually, “feature selection” means choosing from some “original” features.
    - You could say that “feature” selection is a special case of “model” selection.



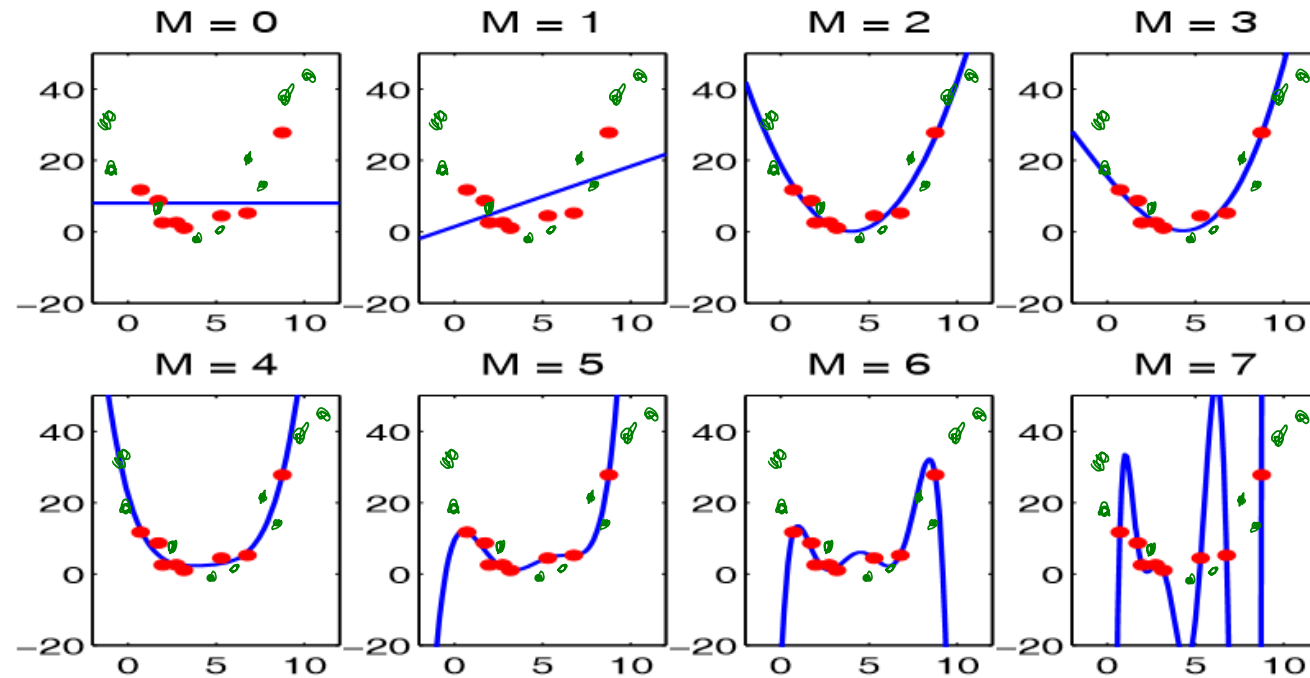
# Can it help prediction to throw features away?

- First, note that **linear regression can overfit** with large 'd'.
  - Even though it's "just" a hyper-plane.
- Consider using  $d=n$ , with completely random features.
  - With high probability, you will be able to **get a training error of 0**.
  - But the features were random, this is **completely overfitting**.
- You could view "**number of features**" as a **hyper-parameter**.
  - Model gets more complex as you add more features.

(pause)

# Recall: Polynomial Degree and Training vs. Testing

- We've said that **complicated models tend to overfit more.**



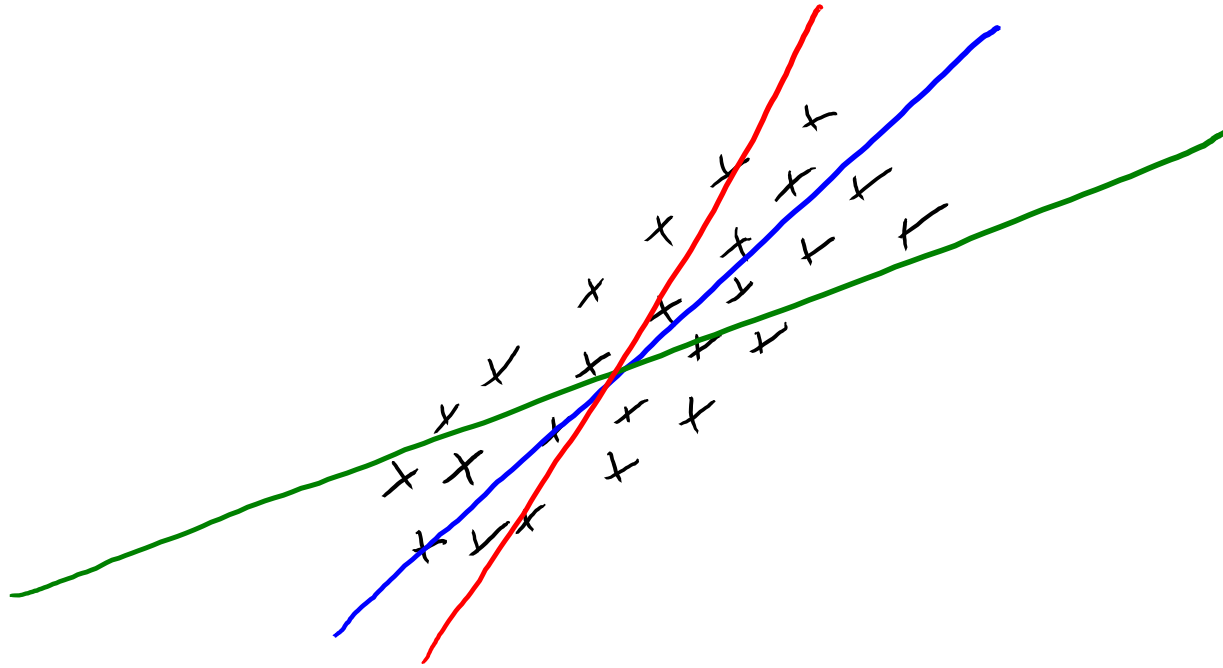
- But what if we **need a complicated model?**

# Controlling Complexity

- Usually “true” mapping from  $x_i$  to  $y_i$  is complex.
  - Might need high-degree polynomial.
  - Might need to combine many features, and don’t know “relevant” ones.
- But complex models can overfit.
- So what do we do???
- Our main tools:
  - Model averaging: average over multiple models to decrease variance.
  - Regularization: add a penalty on the complexity of the model.

# Would you rather?

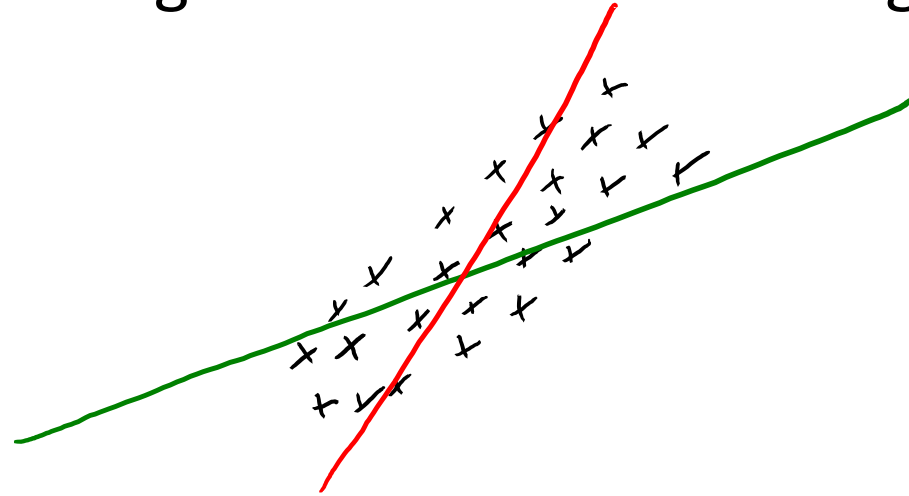
- Consider the following dataset and 3 linear regression models:



- Which line should we choose?

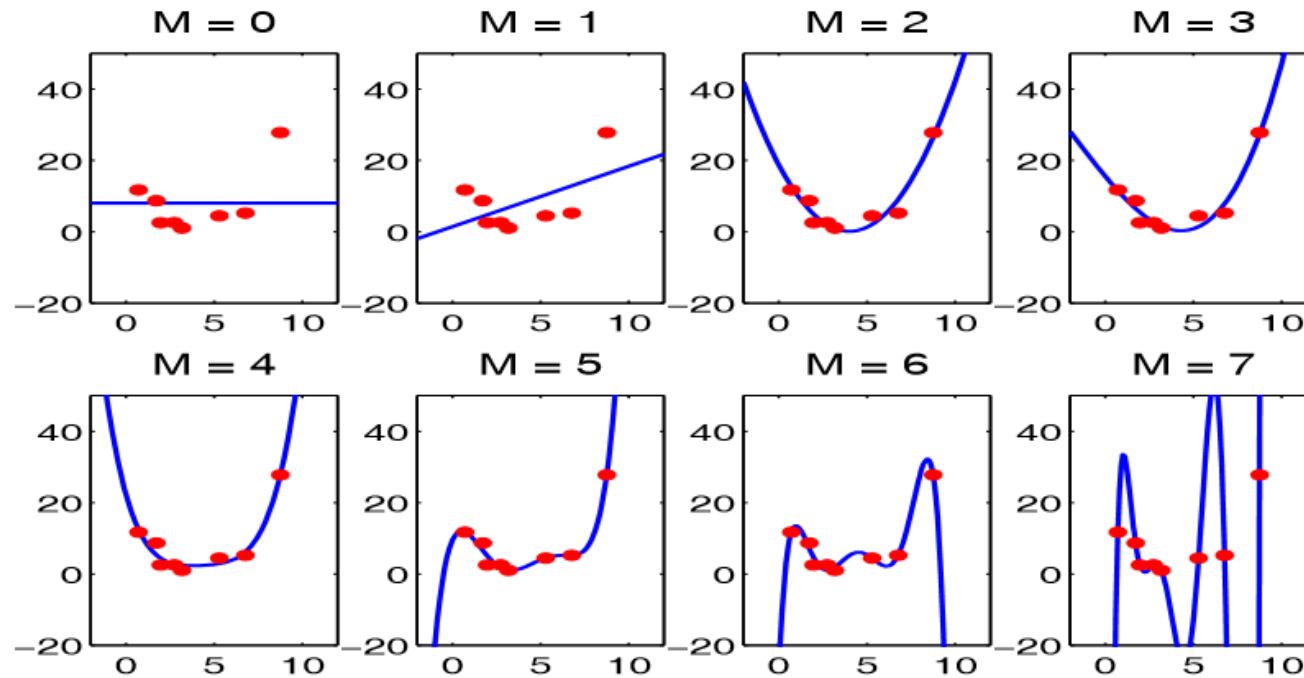
# Would you rather?

- Consider the following dataset and 3 linear regression models:



- What if you are forced to choose between **red** and **green**?
  - And assume they have the same training error.
- You should **pick green**.
  - The slope is smaller, so predictions are **less sensitive to having 'w' exactly right**.
  - Since green 'w' is less sensitive to data, test error might be lower.

# Size of Regression Weights are Overfitting



- The regression weights  $w_j$  with degree-7 are huge in this example.
- The degree-7 polynomial would be less sensitive to the data, if we “regularized” the  $w_j$  so that they are small.

$$\hat{y}_i = 0.0001(x_i)^7 + 0.03(x_i)^3 + 3 \quad \text{vs.} \quad \hat{y}_i = 1000(x_i)^7 - 500(x_i)^6 + 890x_i$$

# L2-Regularization

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2} \sum_{i=1}^n (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^d w_j^2 \quad \text{or} \quad f(w) = \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2} \|w\|^2$$

- Intuition: large slopes  $w_j$  tend to lead to overfitting.
- Objective balances getting low error vs. having small slopes ' $w_j$ '.
  - “You can increase the training error if it makes ‘w’ much smaller.”
  - Nearly-always reduces overfitting.
  - Regularization parameter  $\lambda > 0$  controls “strength” of regularization.
    - Large  $\lambda$  puts large penalty on slopes.



# L2-Regularization

- Standard **regularization** strategy is **L2-regularization**:

$$f(w) = \frac{1}{2} \sum_{i=1}^n (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^d w_j^2 \quad \text{or} \quad f(w) = \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2} \|w\|^2$$

- In terms of fundamental trade-off:
  - Regularization **increases training error**.
  - Regularization **decreases approximation error**.
- How should you choose  $\lambda$ ?
  - Theory: as 'n' grows  $\lambda$  should be in the range  $O(1)$  to  $(\sqrt{n})$ .
  - Practice: optimize **validation set** or **cross-validation** error.
    - This **almost always decreases the test error**.

# L2-Regularization “Shrinking” Example

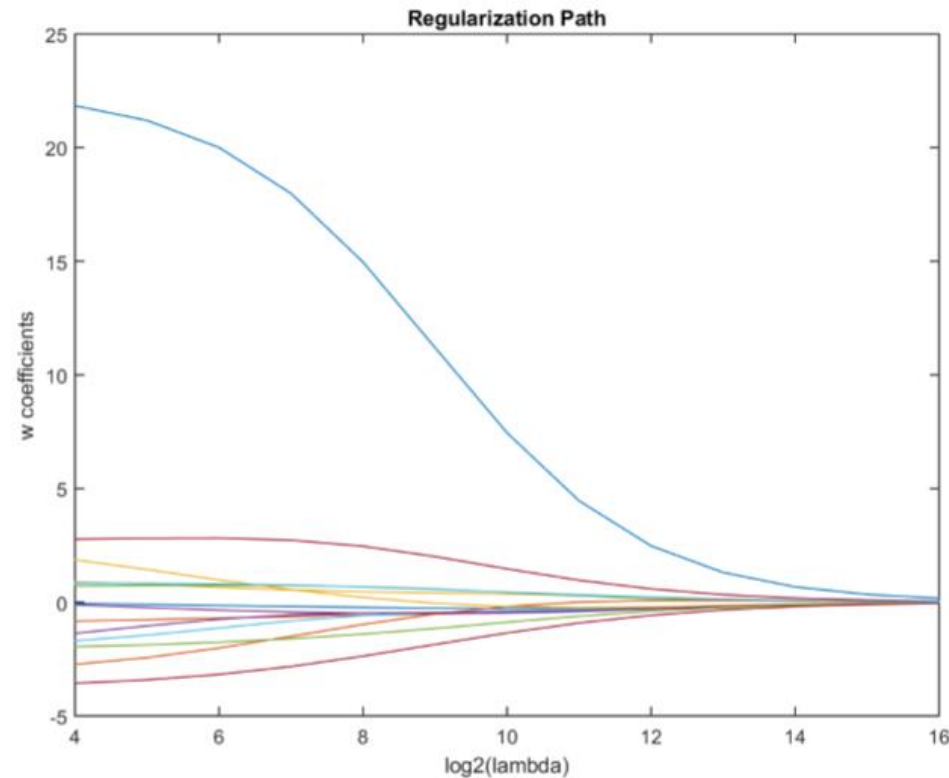
- Solution to a “least squares with L2-regularization” for different  $\lambda$ :

$\lambda$	$w_1$	$w_2$	$w_3$	$w_4$	$w_5$	$  Xw - y  ^2$	$  w  ^2$
0	-1.88	1.29	-2.63	1.78	-0.63	285.64	15.68
1	-1.88	1.28	-2.62	1.78	-0.64	285.64	15.62
4	-1.87	1.28	-2.59	1.77	-0.66	285.64	15.43
16	-1.84	1.27	-2.50	1.73	-0.73	285.71	14.76
64	-1.74	1.23	-2.22	1.59	-0.90	286.47	12.77
256	-1.43	1.08	-1.70	1.18	-1.05	292.60	8.60
1024	-0.87	0.73	-1.03	0.57	-0.81	321.29	3.33
4096	-0.35	0.31	-0.42	0.18	-0.36	374.27	0.56

- We get least squares with  $\lambda = 0$ .
  - But we can achieve similar training error with smaller  $||w||$ .
- $||Xw - y||$  increases with  $\lambda$ , and  $||w||$  decreases with  $\lambda$ .
  - Though individual  $w_j$  can increase or decrease with lambda.
  - Because we use the L2-norm, the large ones decrease the most.

# Regularization Path

- **Regularization path** is a plot of the optimal weights ' $w_j$ ' as ' $\lambda$ ' varies:



- Starts with least squares with  $\lambda=0$ , and  $w_j$  converge to 0 as  $\lambda$  grows.

# L2-regularization and the normal equations

- When using L2-regularized squared error, we can solve for  $\nabla f(w) = 0$ .
- Loss before:  $f(w) = \|Xw - y\|_2^2$
- Loss after:  $f(w) = \|Xw - y\|_2^2 + \lambda \|w\|_2^2$
- Gradient before:  $\nabla f(w) = X^T Xw - X^T y$
- Gradient after:  $\nabla f(w) = X^T Xw - X^T y + \lambda w$
- Linear system before:  $X^T Xw = X^T y$
- Linear system after:  $(X^T X + \lambda I)w = X^T y$
- But unlike  $X^T X$ , the matrix  $(X^T X + \lambda I)$  is always invertible:
  - Multiply by its inverse for unique solution:  $w = (X^T X + \lambda I)^{-1} (X^T y)$

# Gradient Descent for L2-Regularized Least Squares

- The L2-regularized least squares objective and gradient:

$$f(w) = \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2} \|w\|^2 \quad \nabla f(w) = X^T(Xw - y) + \lambda w$$

- Gradient descent iterations for L2-regularized least squares:

$$w^{t+1} = w^t - \alpha^t \left[ \underbrace{X^T(Xw^t - y) + \lambda w^t}_{\nabla f(w^t)} \right]$$

- Cost of gradient descent iteration is still  $O(nd)$ .
  - Can show **number of iterations decrease as  $\lambda$  increases** (not obvious).

# Why use L2-Regularization?

- It's a weird thing to do, but Mark says “always use regularization”.
  - “Almost always decreases test error” should already convince you.
- But here are 6 more reasons:
  1. Solution ‘w’ is **unique**.
  2.  $X^T X$  does **not need to be invertible** (no collinearity issues).
  3. **Less sensitive** to changes in X or y.
  4. Gradient descent **converges faster** (bigger  $\lambda$  means fewer iterations).
  5. Stein's paradox: if  $d \geq 3$ , ‘shrinking’ **moves us closer to ‘true’ w**.
  6. Worst case: just set  $\lambda$  small and get the same performance.

(pause)

# Features with Different Scales

- Consider continuous features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard 'unit'?
  - It **doesn't matter for decision trees or naïve Bayes**.
    - They only look at one feature at a time.
  - It **doesn't matter for least squares**:
    - $w_j \cdot (100 \text{ mL})$  gives the same model as  $w_j \cdot (0.1 \text{ L})$  with a different  $w_j$ .



# Features with Different Scales

- Consider continuous features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard ‘unit’?
  - It **matters for k-nearest neighbours**:
    - “Distance” will be affected more by large features than small features.
  - It **matters for regularized least squares**:
    - Penalizing  $(w_j)^2$  means different things if features ‘j’ are on different scales.

# Standardizing Features

$$X = \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

average of column 'j'

- It is common to **standardize continuous features**:

- For each feature:

1. Compute mean and standard deviation:

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij} \quad \sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \mu_j)^2}$$

2. Subtract mean and divide by standard deviation ("z-score")

Replace  $x_{ij}$  with  $\frac{x_{ij} - \mu_j}{\sigma_j}$

- Now **changes in 'w<sub>j</sub>' have similar effect** for any feature 'j'.
- How should we **standardize test data**?
  - **Wrong approach**: use mean and standard deviation of test data.
  - Training and test mean and standard deviation might be very different.
  - Right approach: **use mean and standard deviation of training data**.

# Standardizing Features

$$X = \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

average of column 'j'

- It is common to **standardize continuous features**:

- For each feature:

1. Compute mean and standard deviation:

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij} \quad \sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \mu_j)^2}$$

2. Subtract mean and divide by standard deviation ("z-score")

Replace  $x_{ij}$  with  $\frac{x_{ij} - \mu_j}{\sigma_j}$

- Now **changes in 'w<sub>j</sub>' have similar effect** for any feature 'j'.
- If we're doing 10-fold cross-validation:
  - Compute  $\mu_j$  and  $\sigma_j$  based on the 9 training folds (e.g., average over 9/10s of data).
  - Standardize the remaining ("validation") fold with this "training"  $\mu_j$  and  $\sigma_j$ .
  - Re-standardize for different folds.

# Standardizing Target

- In regression, we sometimes **standardize the targets  $y_i$** .
  - Puts targets on the same standard scale as standardized features:

$$\text{Replace } y_i \text{ with } \frac{y_i - \mu_y}{\sigma_y}$$

- With standardized target, setting  $w = 0$  **predicts average  $y_i$** :
  - High **regularization makes us predict closer to the average** value.
- Again, make sure you **standardize test data with the training stats**.
- Other common transformations of  $y_i$  are logarithm/exponent:

$$\text{Use } \log(y_i) \text{ or } \exp(\gamma y_i)$$

- Makes sense for geometric/exponential processes.

# Regularizing the y-Intercept?

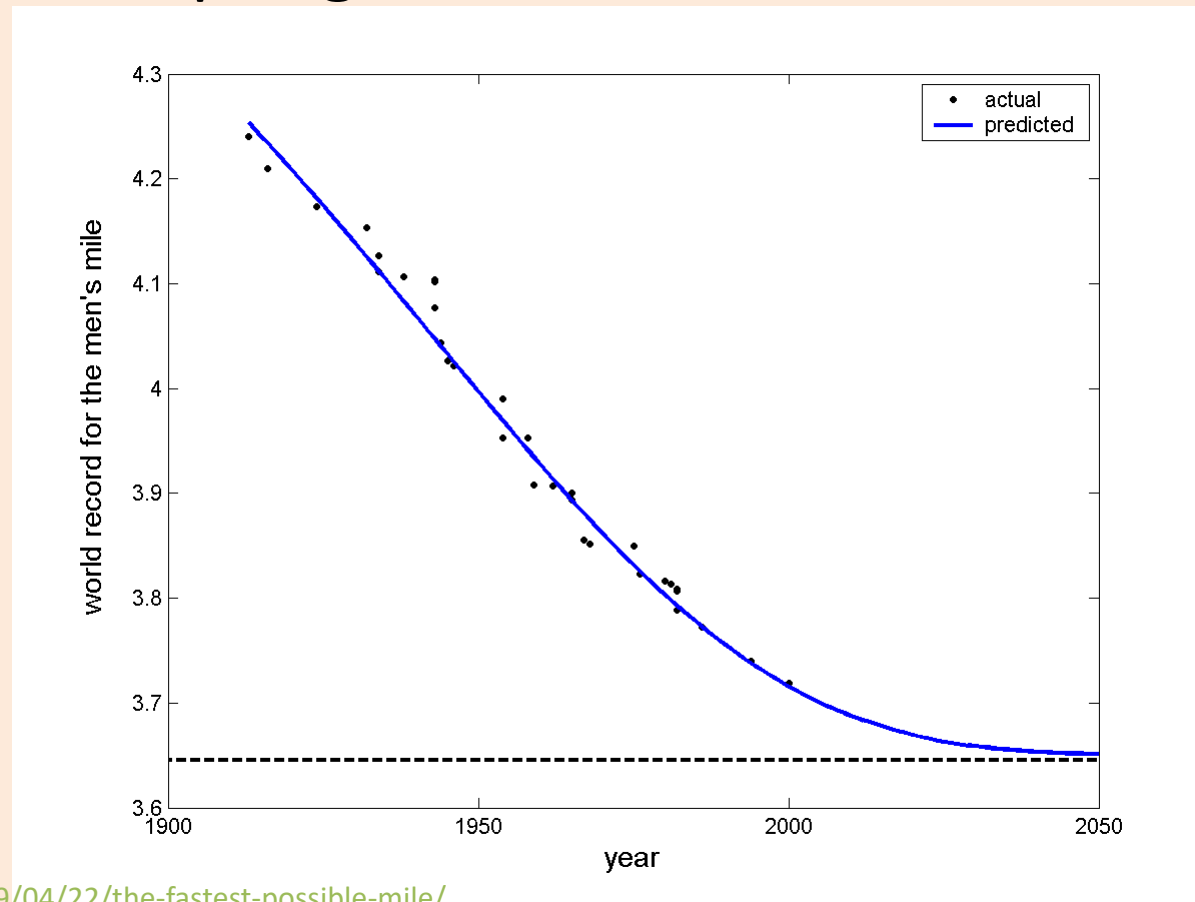
- Should we **regularize the y-intercept**?
- No! Why encourage it to be closer to zero? (It could be anywhere.)
  - You should be allowed to shift function up/down globally.
- Yes! It makes the solution unique and it easier to compute 'w'.
- Compromise: regularize by a **smaller amount** than other variables.

$$f(w, w_0) = \frac{1}{2} \|Xw + w_0 - y\|^2 + \frac{\lambda}{2} \|w\|^2 + \frac{\lambda_0}{2} w_0^2$$

(pause)

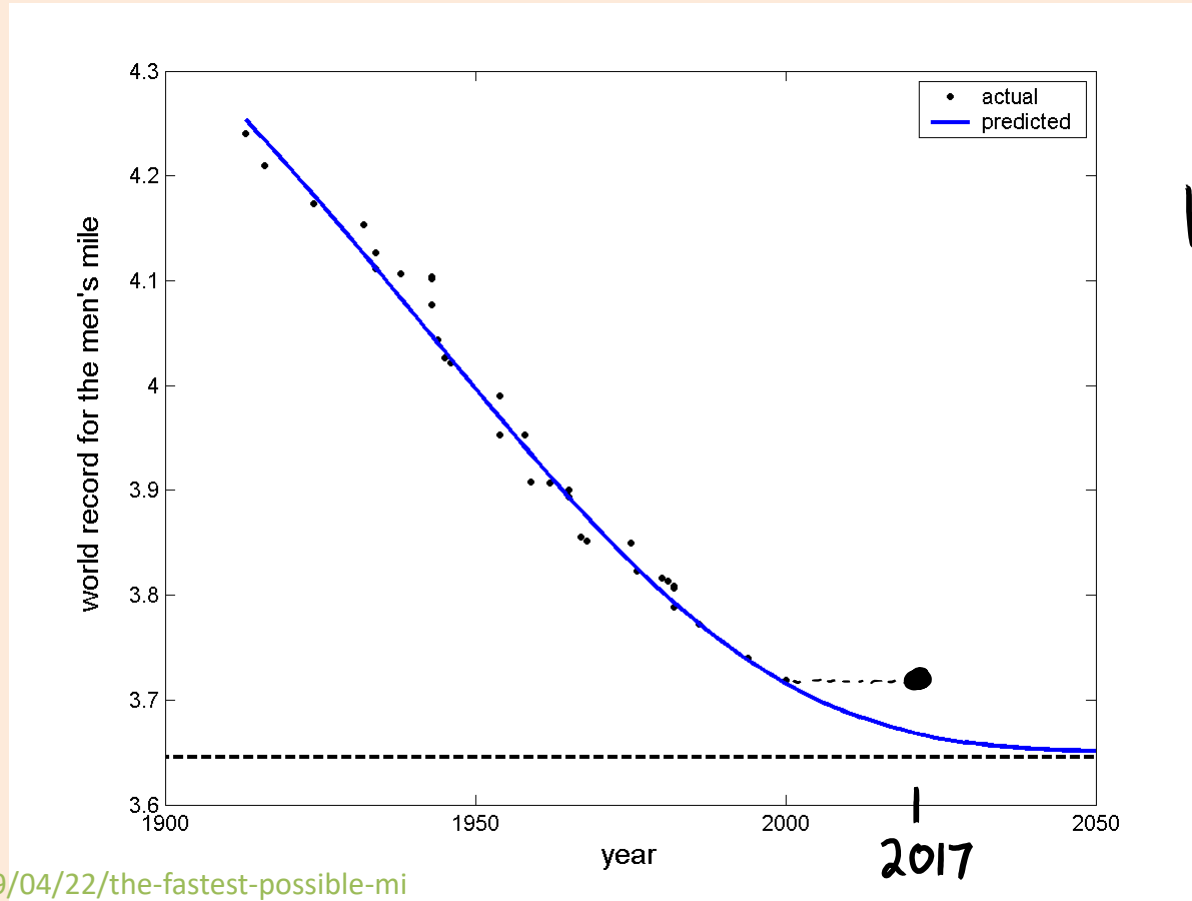
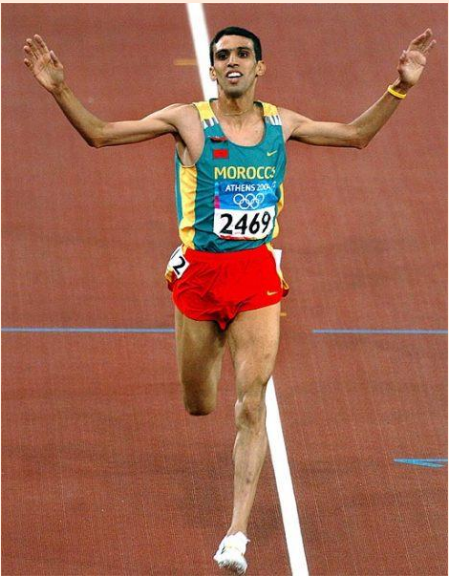
# Predicting the Future

- In principle, we can use any features  $x_i$  that we think are relevant.
- This makes it tempting to use **time** as a feature, and predict future.



# Predicting the Future

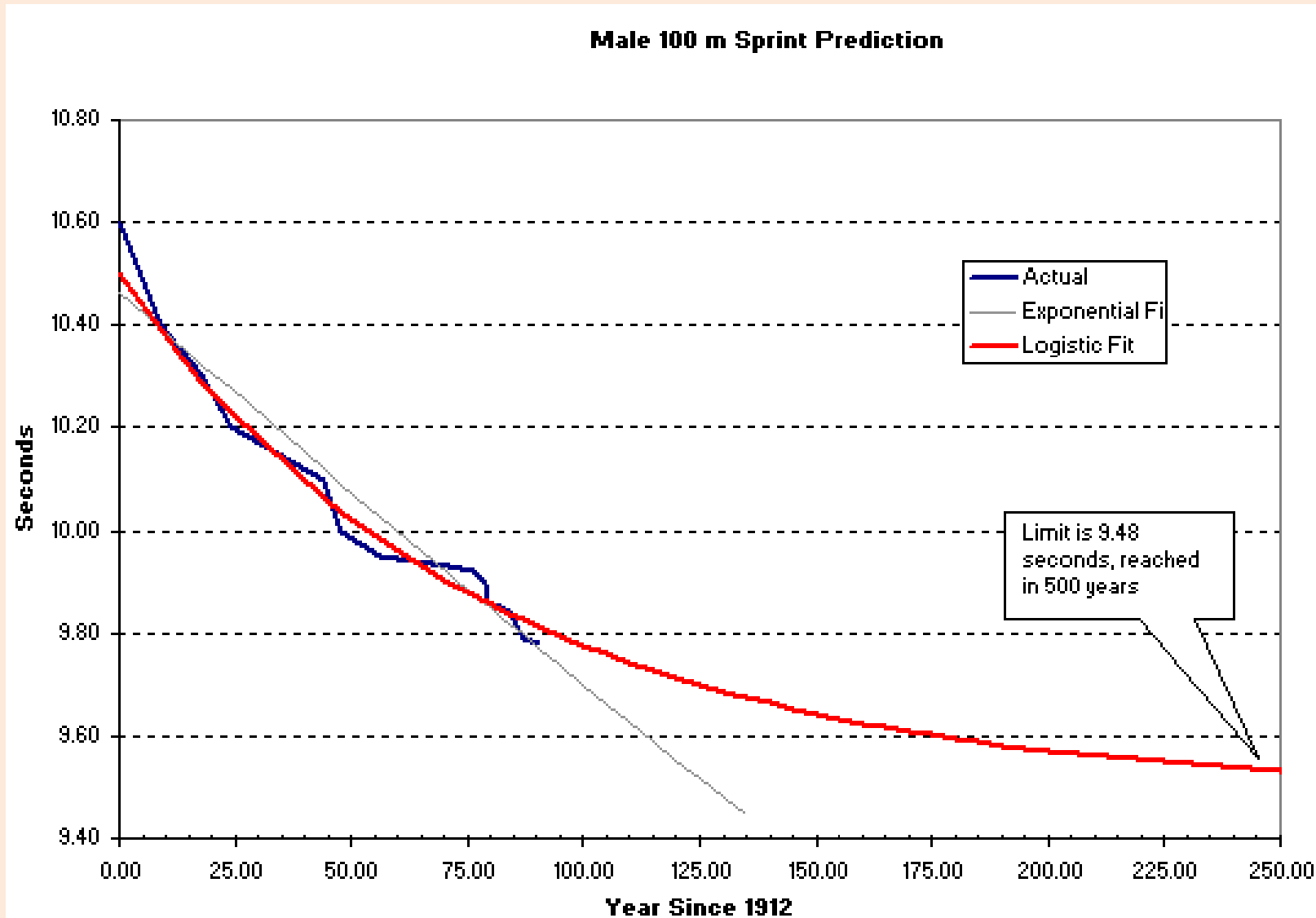
- In principle, we can use any features  $x_i$  that we think are relevant.
- This makes it tempting to use **time as a feature**, and predict future.



We need to be  
Cautious about  
doing this.



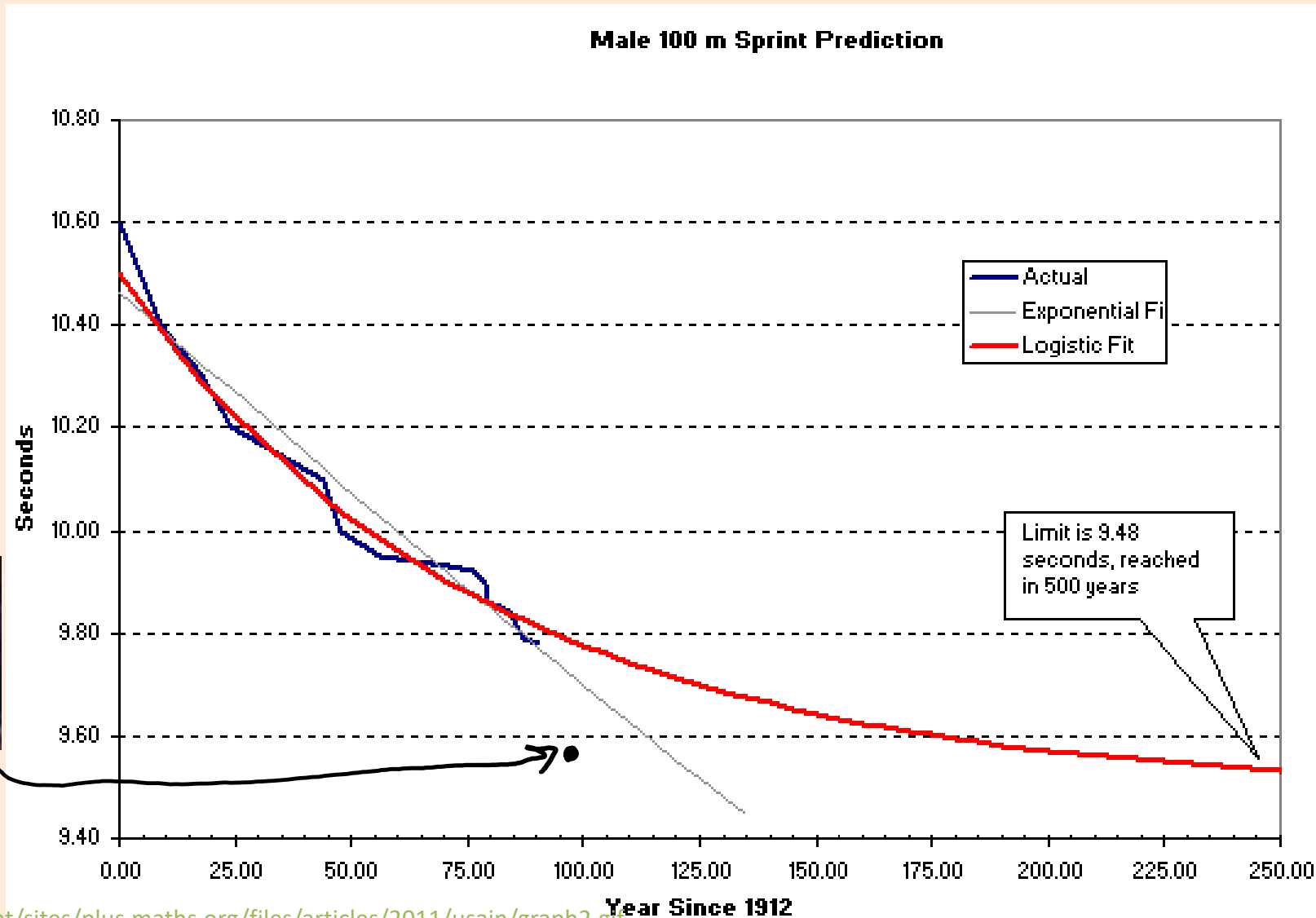
# Predicting 100m times 400 years in the future?



# Predicting 100m times 400 years in the future?



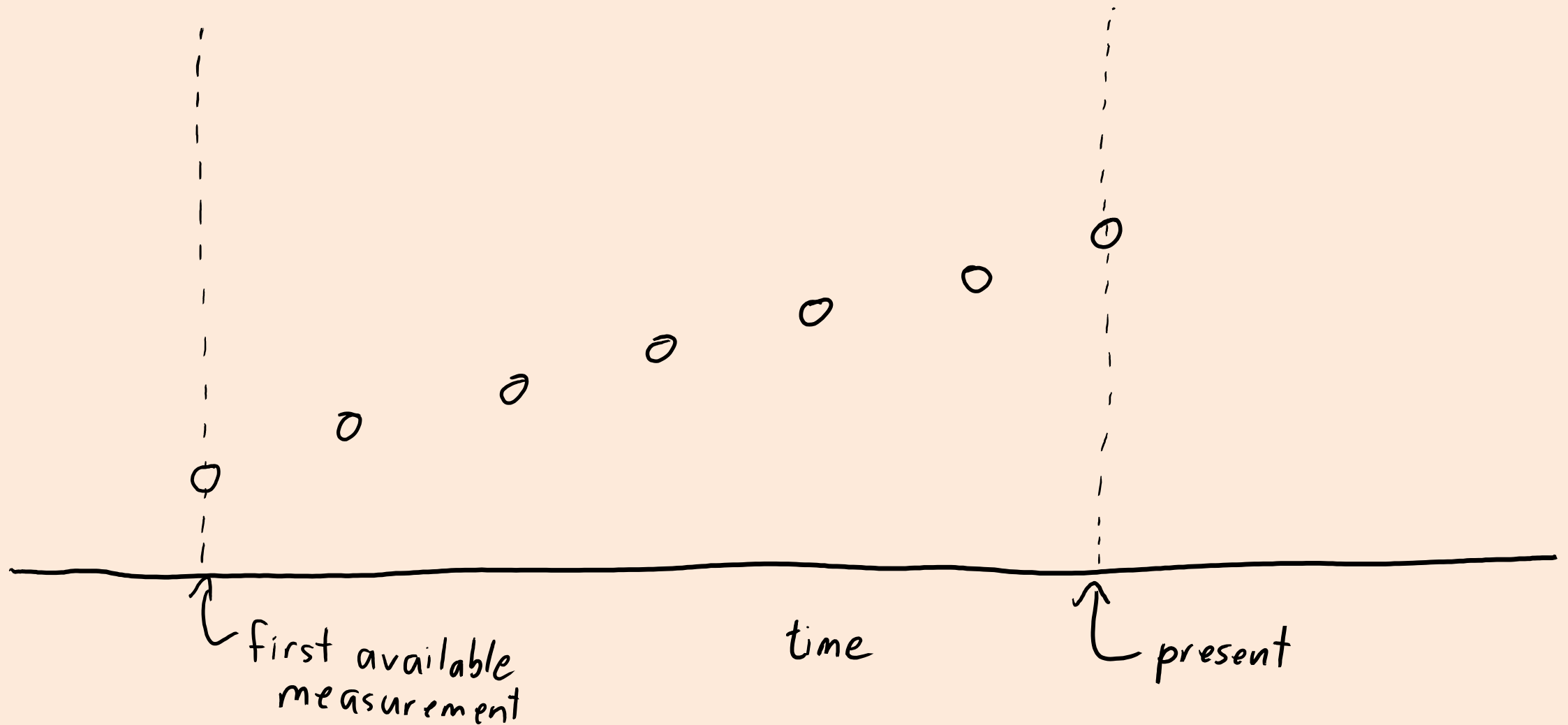
9.58



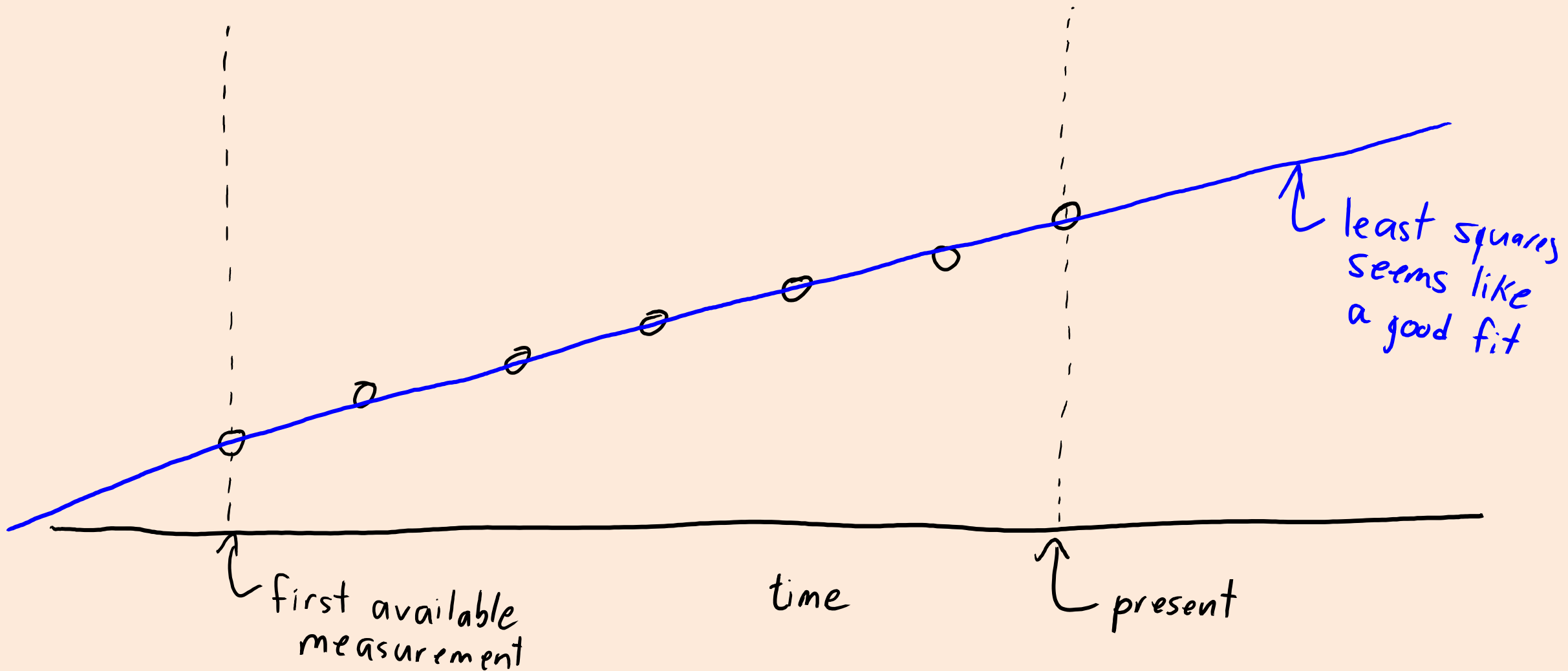
# Interpolation vs Extrapolation

- **Interpolation** is task of predicting “between the data points”.
  - Regression models are good at this if you have enough data and function is continuous.
- **Extrapolation** is task of prediction outside the range of the data points.
  - Without assumptions, regression models can be embarrassingly-bad at this.
- If you run the 100m regression models backwards in time:
  - They predict that **humans used to be really really slow!**
- If you run the 100m regression models forwards in time:
  - They might eventually predict arbitrarily-small 100m times.
  - The linear model actually predicts **negative times** in the future.
    - These time traveling races in 2060 should be pretty exciting!
- Some discussion here:
  - [http://callingbullshit.org/case\\_studies/case\\_study\\_gender\\_gap\\_running.html](http://callingbullshit.org/case_studies/case_study_gender_gap_running.html)

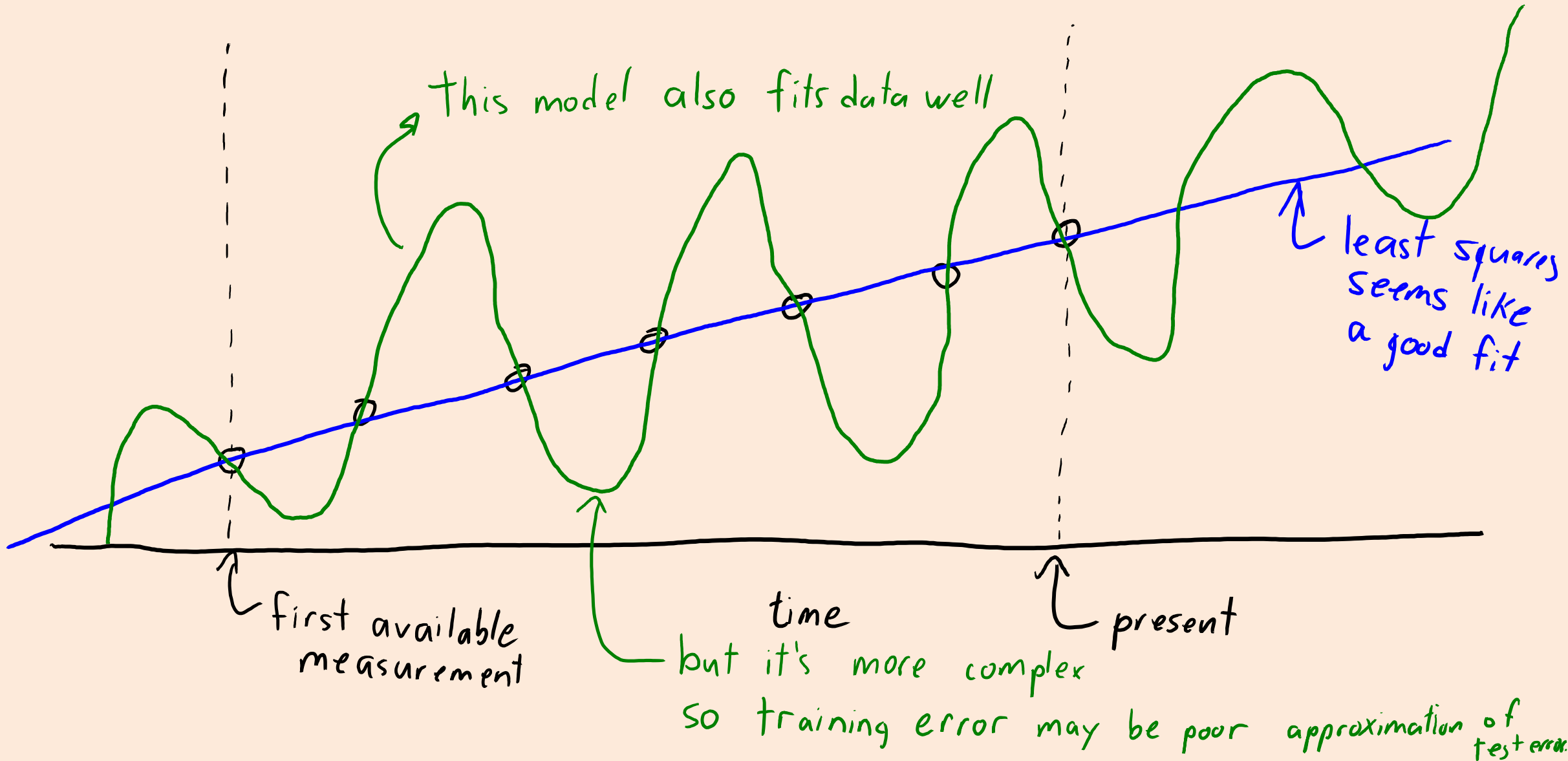
# No Free Lunch, Consistency, and the Future



# No Free Lunch, Consistency, and the Future

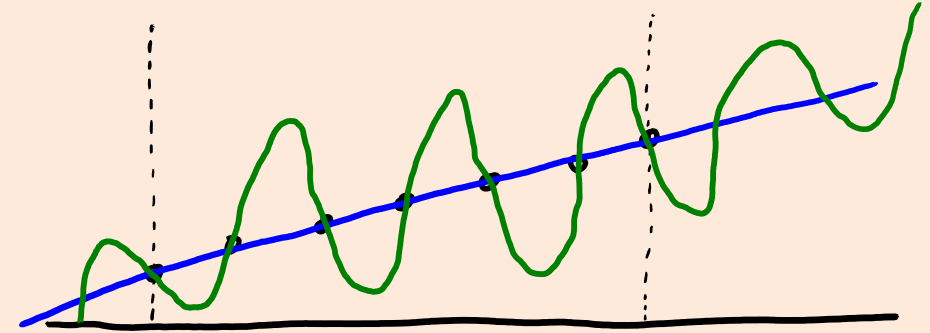


# No Free Lunch, Consistency, and the Future

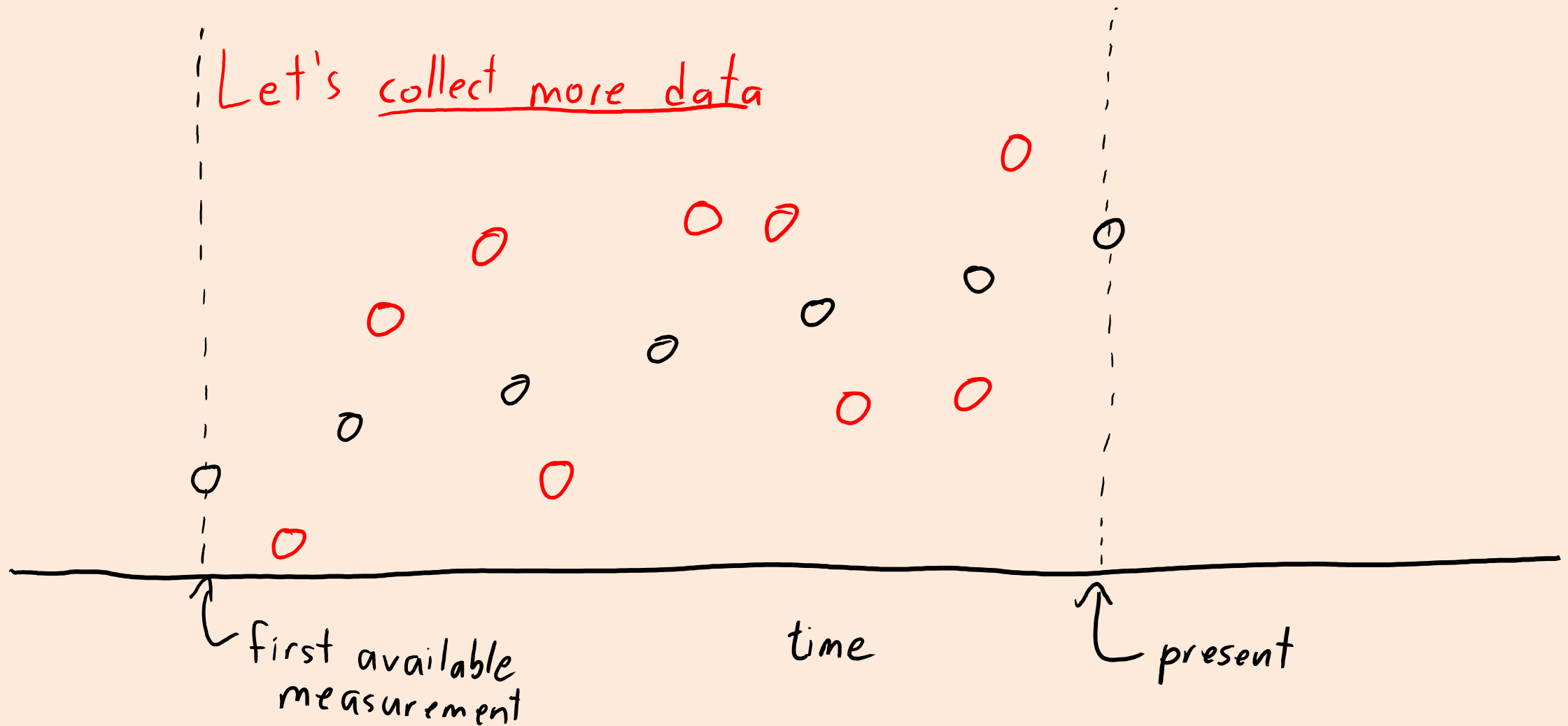


# Ockham's Razor vs. No Free Lunch

- **Ockham's razor** is a problem-solving principle:
  - “Among competing hypotheses, the one with the fewest assumptions should be selected.”
  - Suggests we should **select linear model**.
- **Fundamental trade-off:**
  - If same training error, pick model less likely to overfit.
  - Formal version of Occam's problem-solving principle.
  - Also suggests we should **select linear model**.
- **No free lunch theorem:**
  - There *exists possible datasets* where you should select the **green model**.

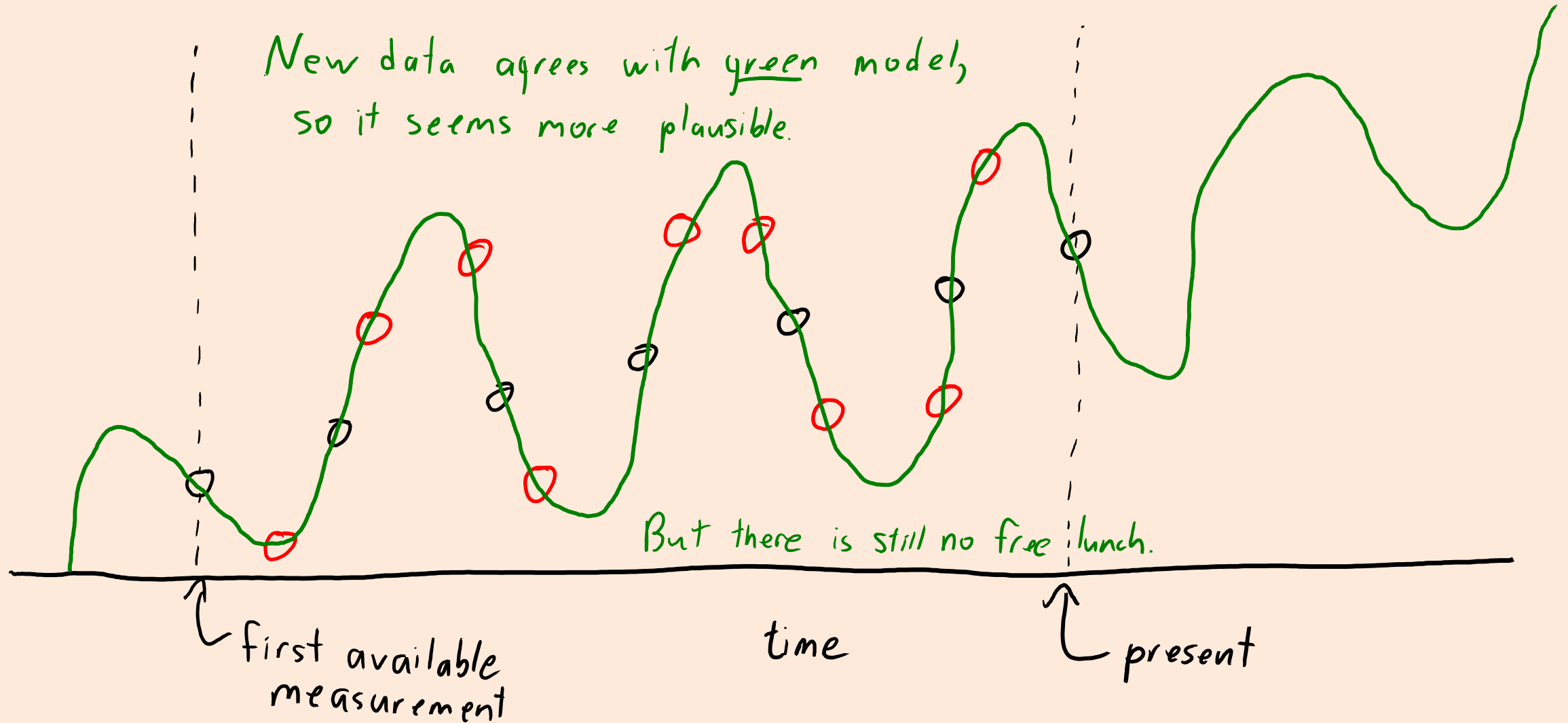


# No Free Lunch, Consistency, and the Future

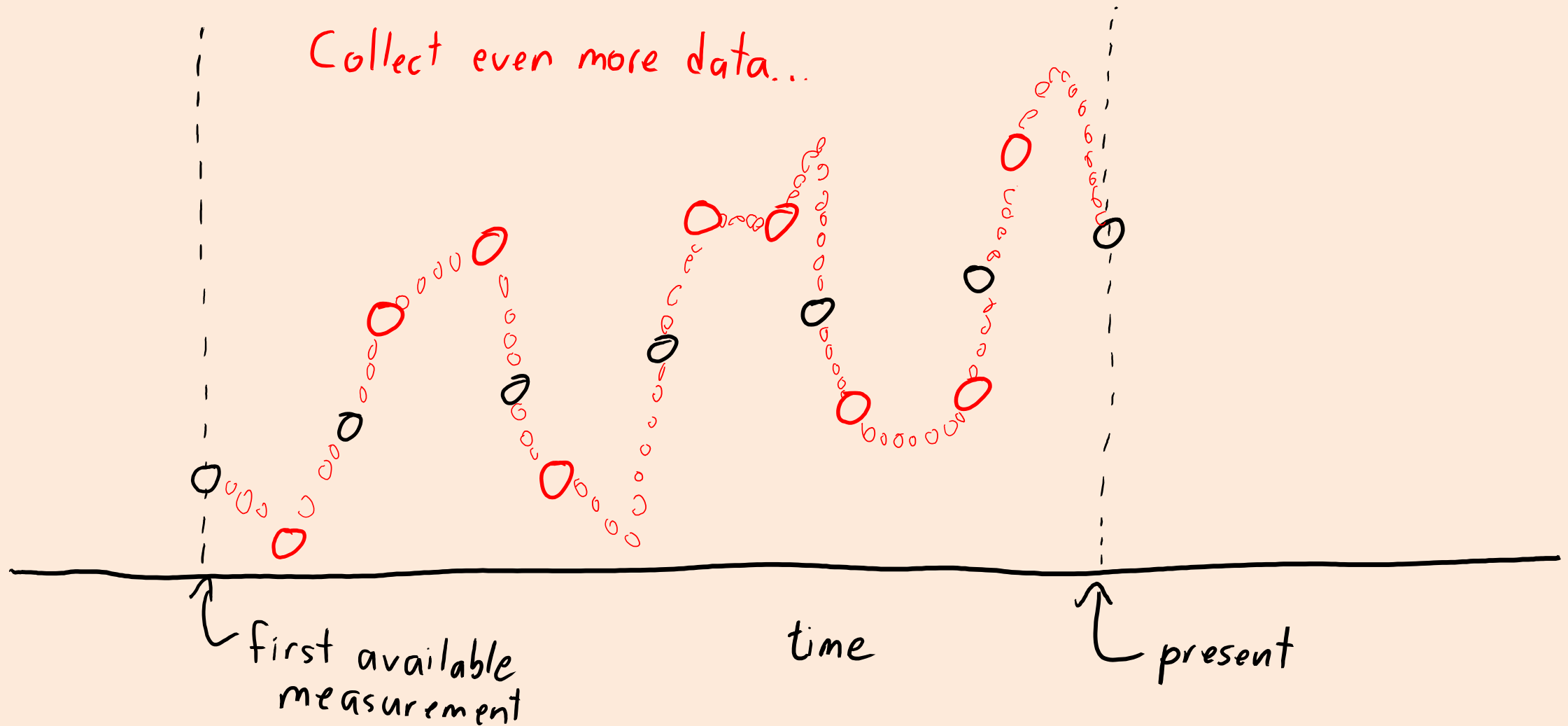




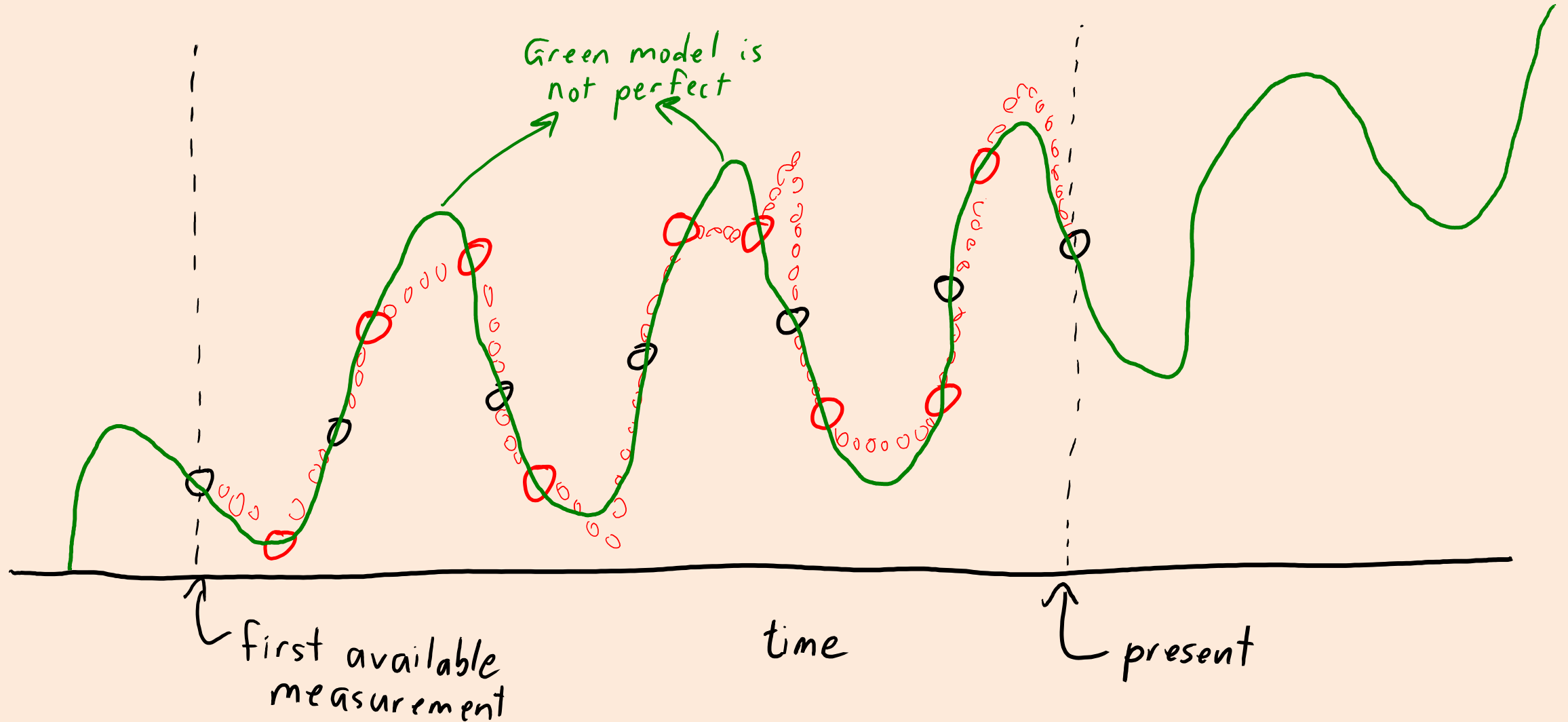
# No Free Lunch, Consistency, and the Future



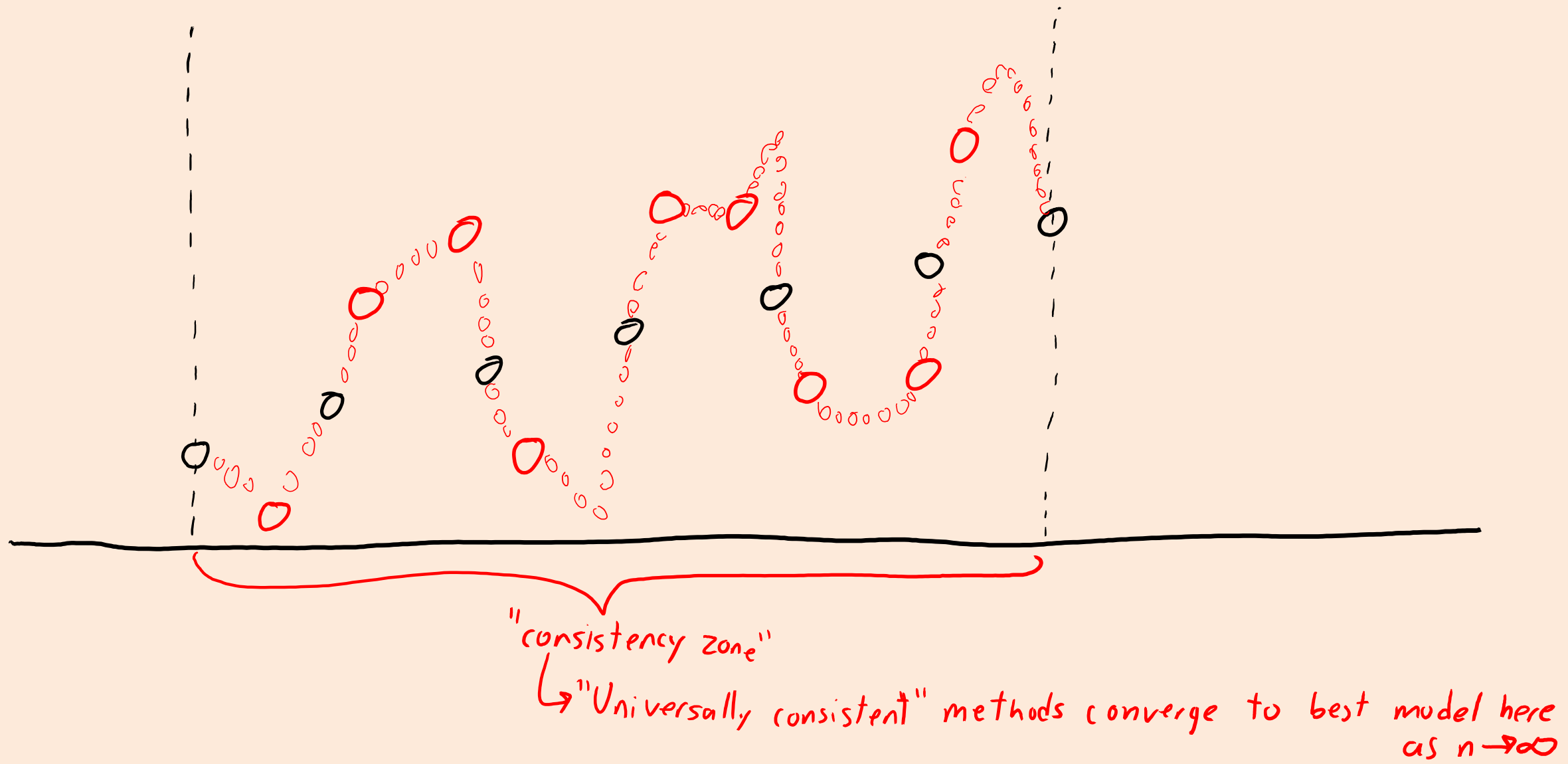
# No Free Lunch, Consistency, and the Future



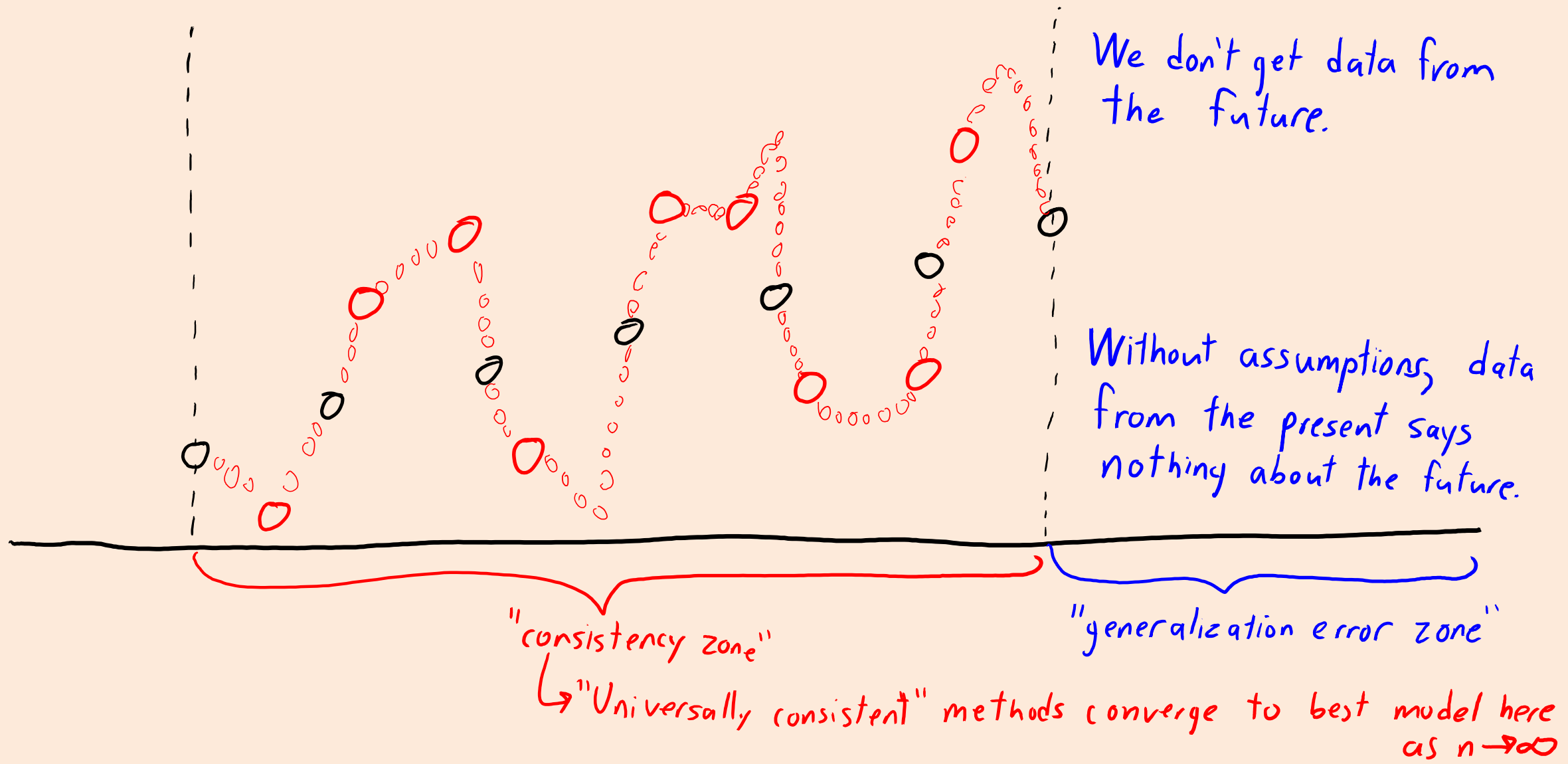
# No Free Lunch, Consistency, and the Future



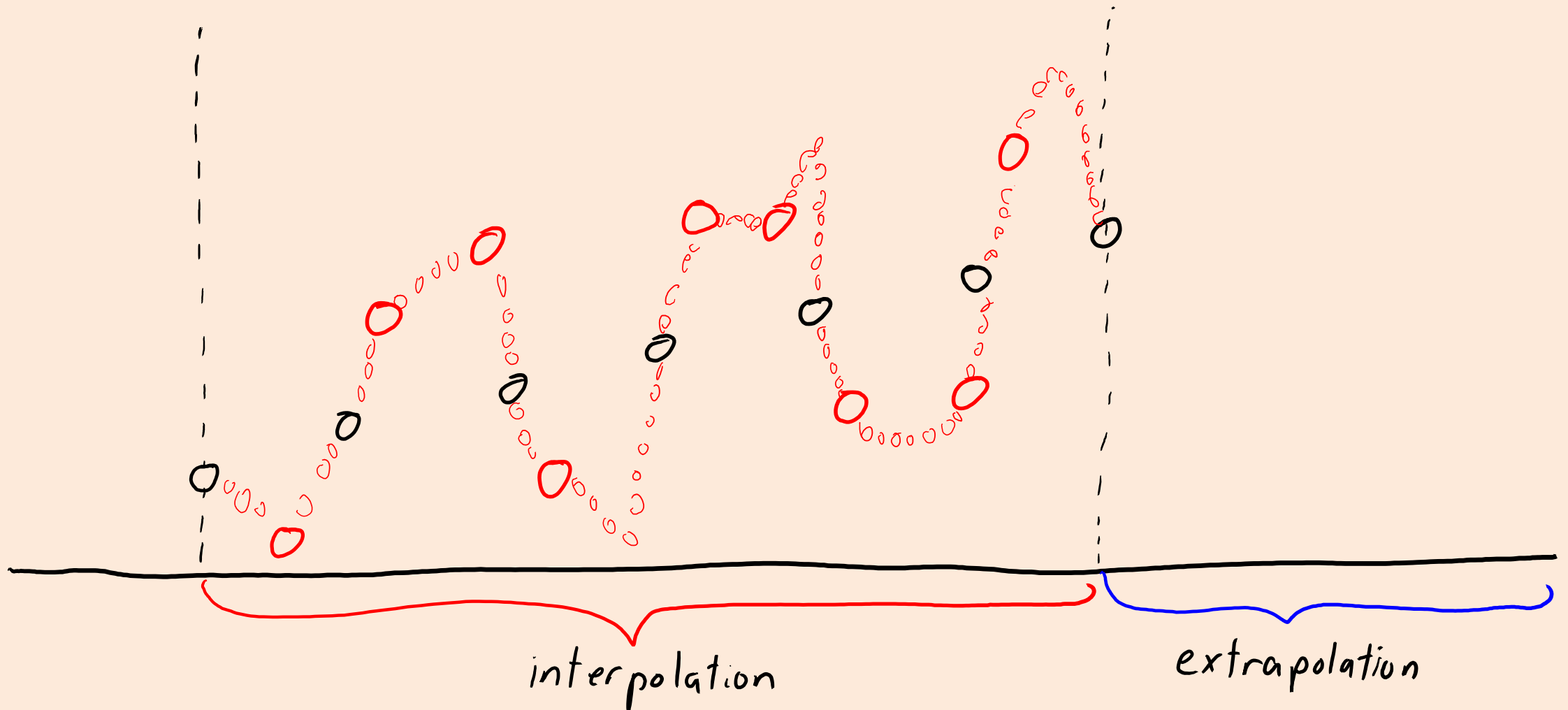
# No Free Lunch, Consistency, and the Future



# No Free Lunch, Consistency, and the Future



# No Free Lunch, Consistency, and the Future



# Summary

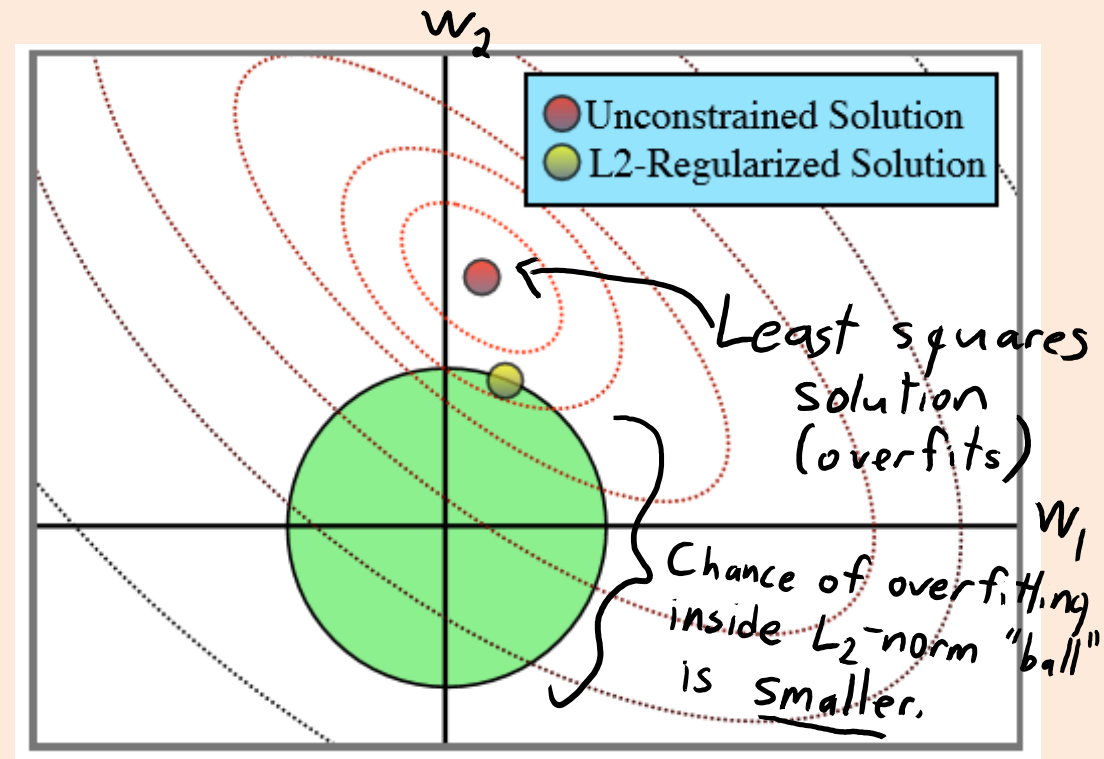
- **Regularization:**
  - Adding a penalty on model complexity.
- **L2-regularization:** penalty on L2-norm of regression weights ' $w$ '.
  - Almost always improves test error.
- **Standardizing features:**
  - For some models it makes sense to have features on the same scale.
- **Interpolation vs. Extrapolation:**
  - Machine learning with large ' $n$ ' is good at predicting “between the data”.
  - Without assumptions, can be arbitrarily bad “away from the data”.
- Next time: learning with an exponential number of irrelevant features.

# L2-Regularization

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2} \sum_{i=1}^n (w^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^d w_j^2 \quad \text{or} \quad f(w) = \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2} \|w\|^2$$

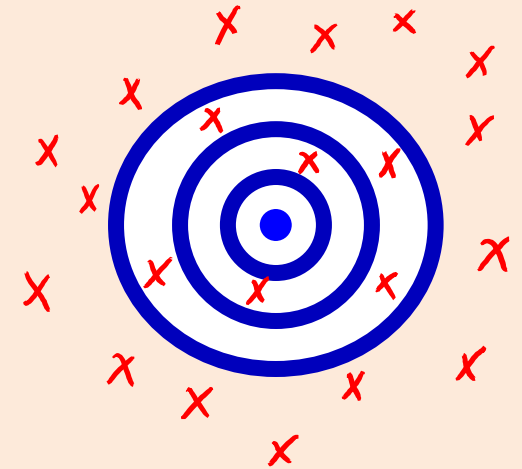
- Equivalent to minimizing squared error but keeping L2-norm small.





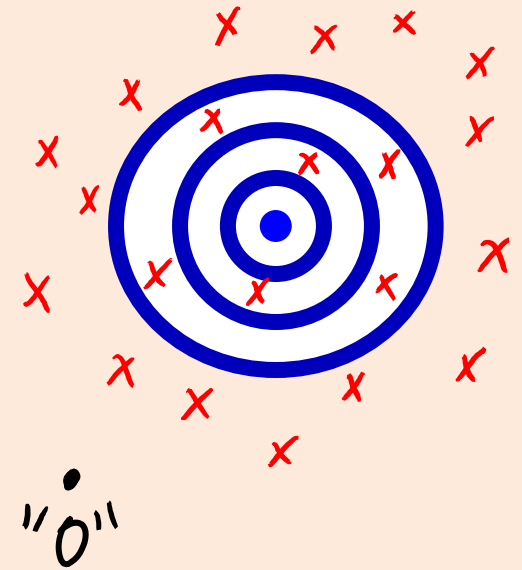
# Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.



# Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.
- Shrinkage of the darts :
  1. Choose some **arbitrary** location '0'.
  2. Measure distances from darts to '0'.



# Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.
- Shrinkage of the darts :
  1. Choose some **arbitrary** location '0'.
  2. Measure distances from darts to '0'.
  3. **Move misses towards '0', by *small* amount proportional to distance from 0.**
- If small enough, **darts will be closer to center on average.**



# Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.
- Shrinkage of the darts :
  1. Choose some **arbitrary** location '0'.
  2. Measure distances from darts to '0'.
  3. **Move misses towards '0', by *small* amount proportional to distance from 0.**
- If small enough, **darts will be closer to center on average.**



Visualization of the related higher-dimensional paradox that the mean of data coming from a Gaussian is not the best estimate of the mean of the Gaussian in 3-dimensions or higher: <https://www.naftaliharris.com/blog/steinviz>