CPSC 340:
Machine Learning and Data Mining

Regularization

Fall 2016

Admin

Ignore Assignment 1 marks on Connect.

- Assignment 2:
  - 2 late days to hand it in Friday, 3 for Monday.
- Assignment 3 is out.
  - Due next Wednesday (so we can release solutions before the midterm).
- Tutorial room change: T1D (Monday @5pm) moved to DMP 101.

- Assignment tips:
  - Put your name and ID numbers on your assignments.
  - Do the assignment from this year.

Last Time: Normal Equations and Change of Basis

- Last time we derived normal equations:

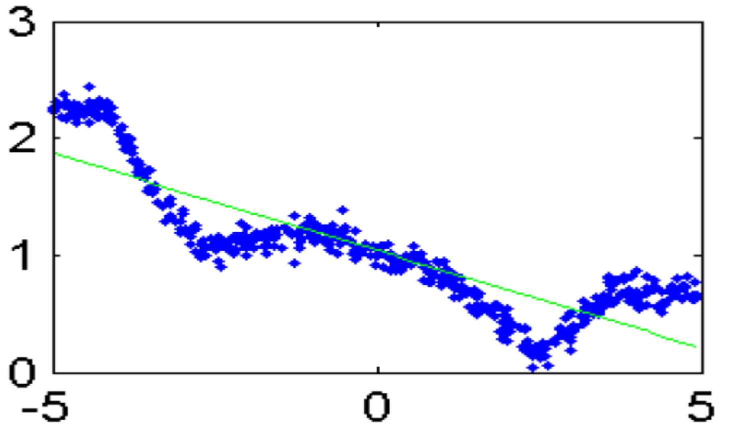
$$X^T X w = X^T y$$

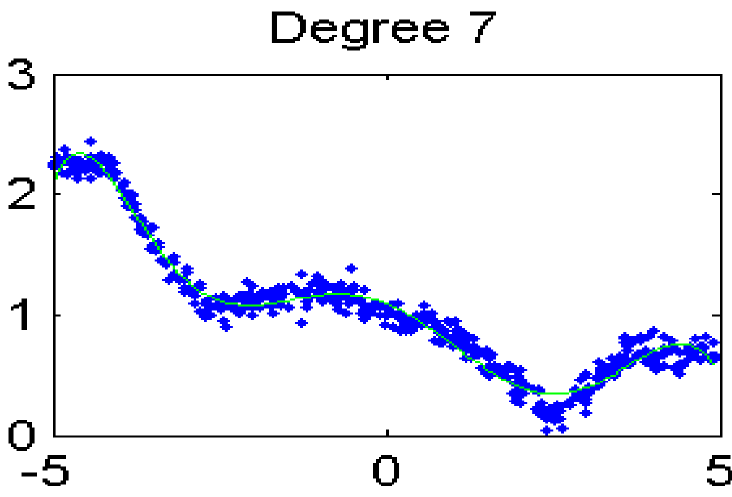
  - Solutions 'w' minimize squared error in linear model.
- We also discussed change of basis:
  - E.g., polynomial basis:

Replace $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ with $Z = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$

  - Let's you fit non-linear models with linear regression.

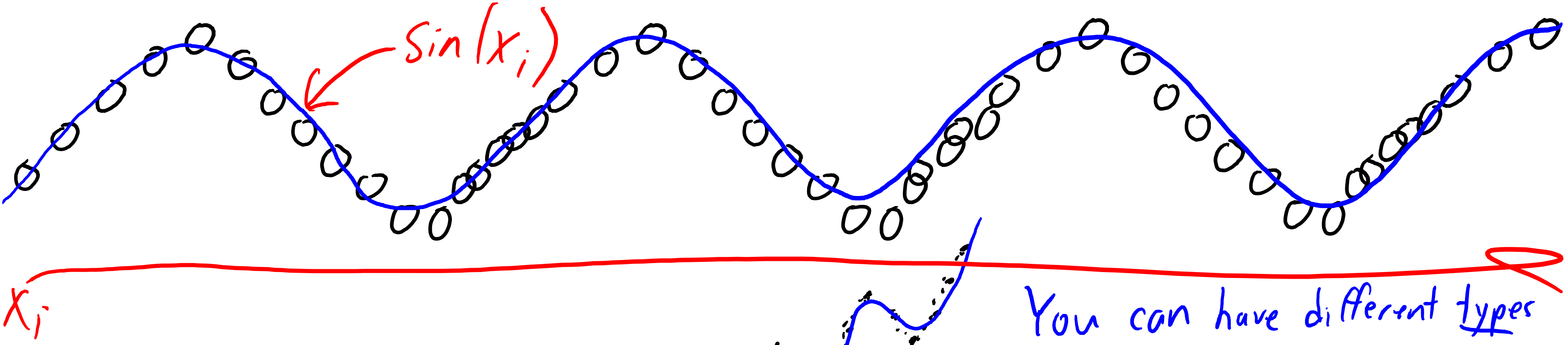
$$y_i = w^T z_i = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3 + \cdots + w_p x_i^p$$





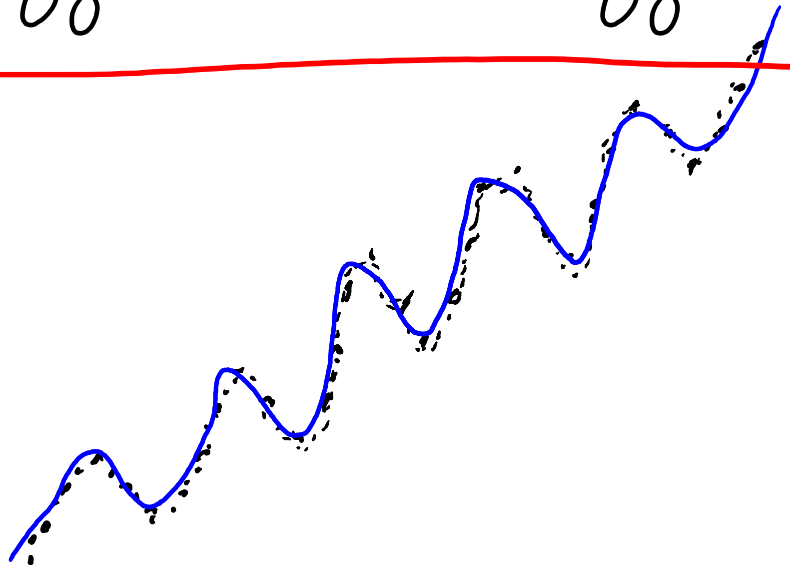
Parametric vs. Non-Parametric Bases

- Polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.



For periodic data we might use

$$Z = \begin{bmatrix} \sin(x_1) \\ \sin(x_2) \\ \vdots \\ \sin(x_n) \end{bmatrix}$$



You can have different types of bases

$$Z = \begin{bmatrix} x_1 & \sin(6x_1) \\ x_2 & \sin(6x_2) \\ \vdots & \vdots \\ \underbrace{x_n}_{\text{linear}} & \underbrace{\sin(6x_n)}_{\text{periodic}} \end{bmatrix}$$

Parametric vs. Non-Parametric Bases

- Polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.
  - But the right basis may not be obvious.
- What happens if we use the wrong basis?
  - As ‘n’ increases, we can fit ‘w’ more accurately.
  - But eventually more data doesn’t help if basis isn’t “flexible” enough.
- Alternative is non-parametric bases:
  - Size of basis (number of features) grows with ‘n’.
  - Model gets more complicated as you get more data.
  - You can model very complicated functions where you don’t know the right basis.

Non-Parametric Basis: RBFs

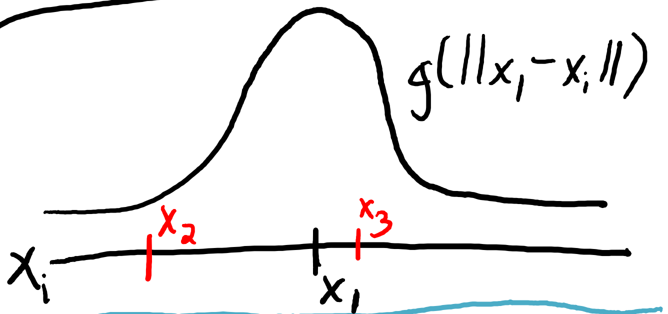
- Radial basis functions (RBFs):
  - Non-parametric bases that depend on distances to training points.

Replace $X = \underbrace{\begin{bmatrix} \quad & \quad \\ & \\ & \end{bmatrix}}_{d} \Big\} n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}}_{n} \Big\} n$

  - Most common ‘g’ is Gaussian RBF:

$$g(\alpha) = \exp\left(-\frac{\alpha^2}{2\sigma^2}\right)$$

Parameter



    - Variance σ^2 controls influence of nearby points.
    - This affects fundamental trade-off (set it using a validation set).

Do we need $\sigma\sqrt{2\pi}$?
– No because $w^T x_i = (\frac{1}{\beta} w)^T (\beta x_i)$

Non-Parametric Basis: RBFs

- Radial basis functions (RBFs):
  - Non-parametric bases that depend on distances to training points.

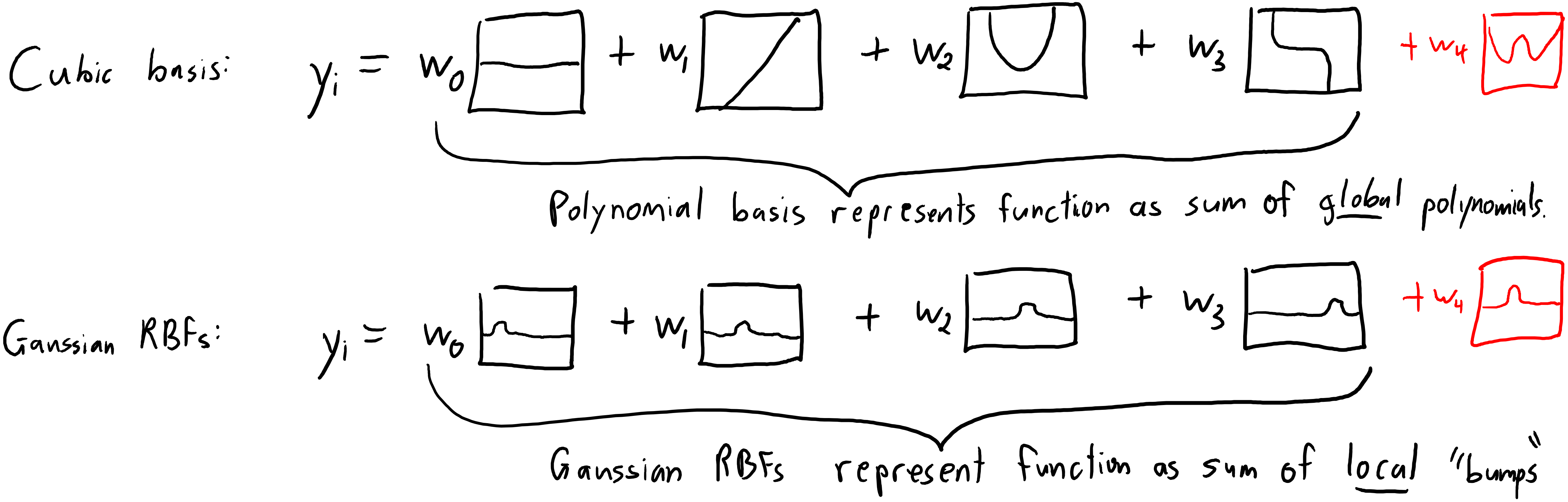
Replace $X = \underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{d} \Big\} n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}}_{n} \Big\} n$

To make predictions on $\hat{X} = \underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{d} \Big\} t$ use $\hat{Z} = \underbrace{\begin{bmatrix} \\ g(\|\hat{x}_i - x_j\|) \\ \\ \end{bmatrix}}_{n} \Big\} t$

I.e., $y_i = w^T \hat{z}_i$
or $y = \hat{Z} w$

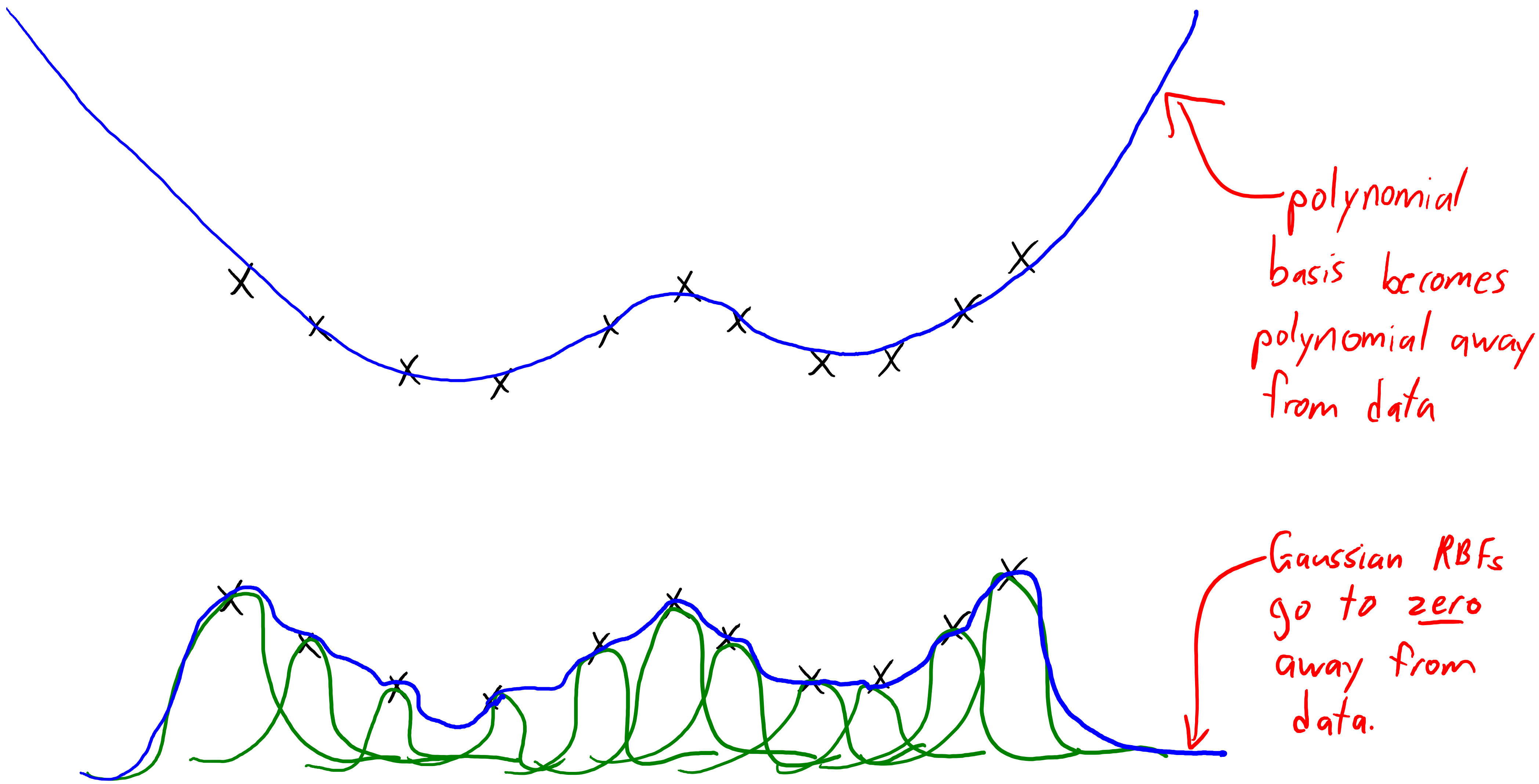
Number of "features" is number of training examples.

Non-Parametric Basis: RBFs



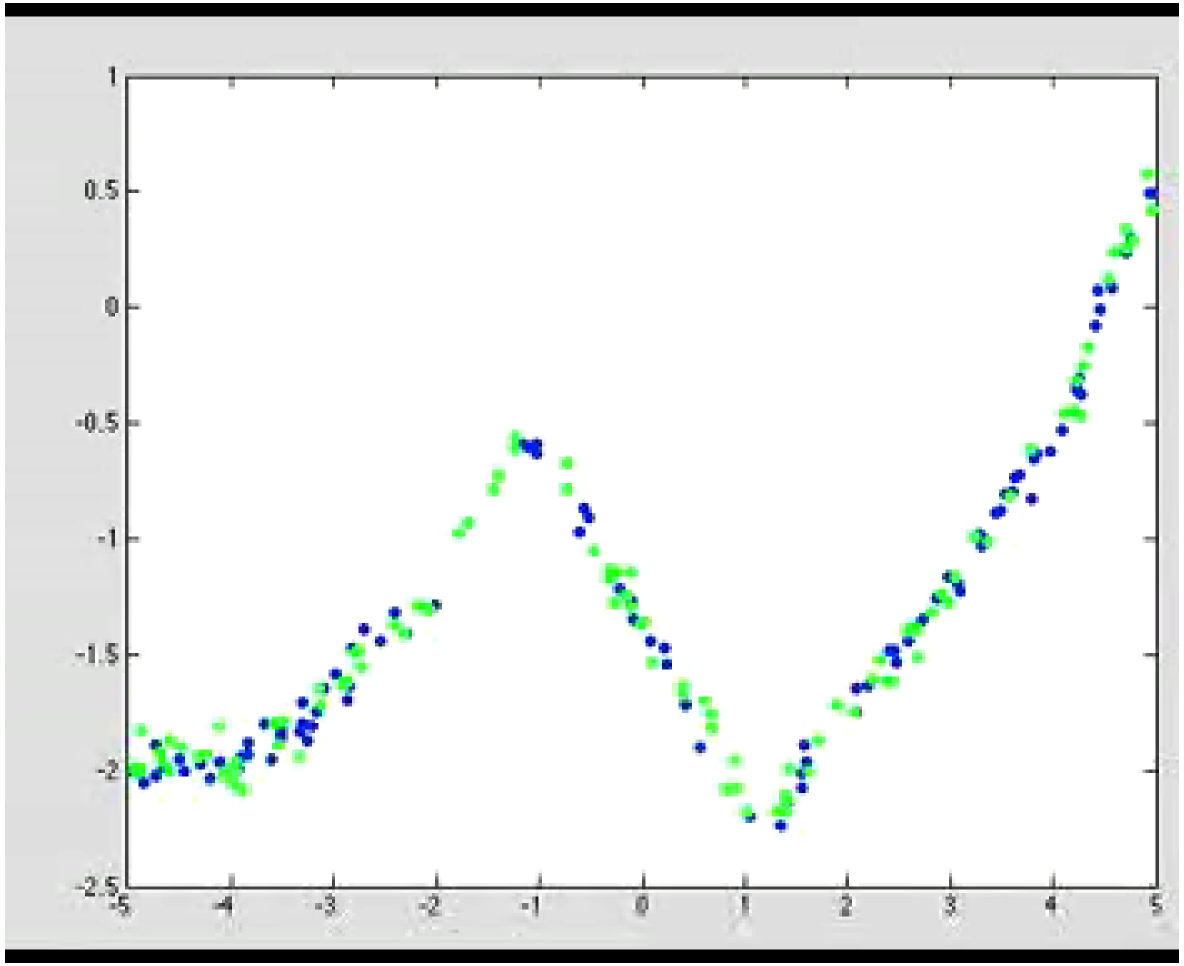
- Gaussian RBFs are universal approximators (compact subets of $\mathbb{R}^d$)
  - Can approximate any continuous function to arbitrary precision.
  - Achieve irreducible error as 'n' goes to infinity.

Interpolation vs. Extrapolation



Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



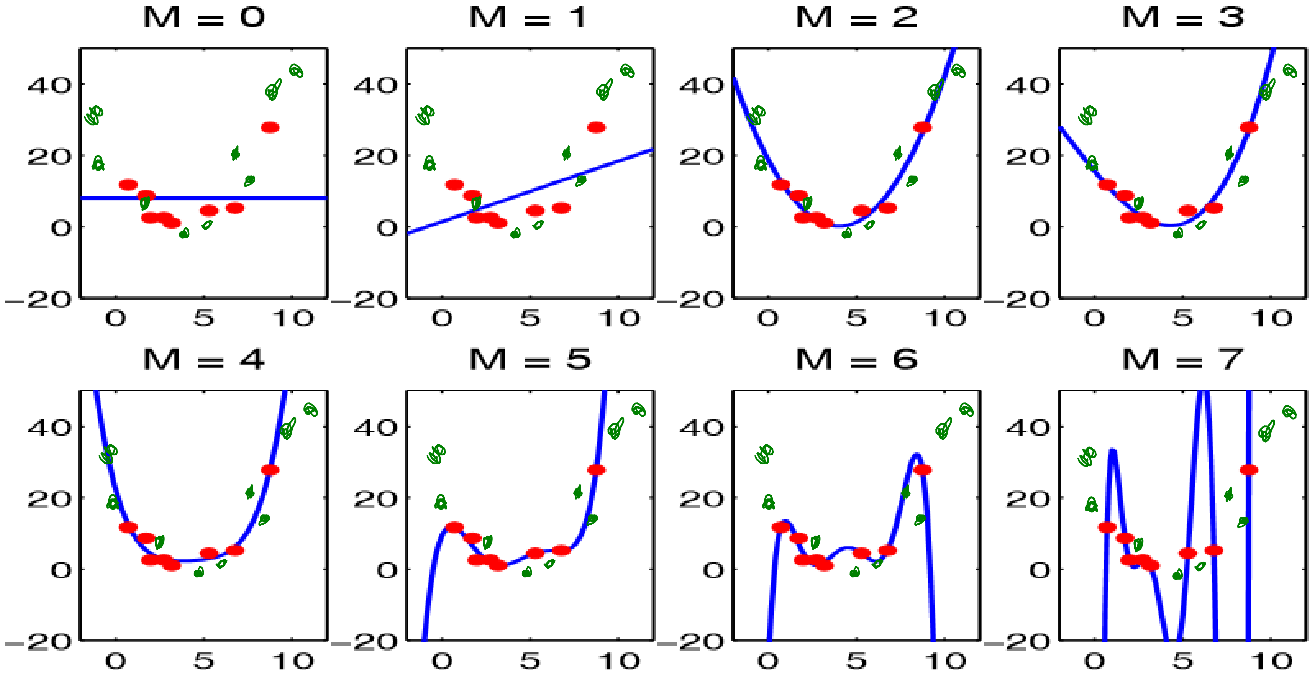
Could add bias and linear basis:

$$
Z = \begin{array}{c}
\left[\begin{array}{cccccc}
1 & - & x_1 & - & g(\|x_1 - x_1\|) & \cdots & g(\|x_1 - x_n\|) \\
1 & - & x_2 & - & \vdots & \ddots & \vdots \\
1 & - & x_3 & - & & & \\
\vdots & & \vdots & & & & \\
1 & - & x_n & - & g(\|x_1 - x_n\|) & \cdots & g(\|x_n - x_n\|)
\end{array}\right] \\
\begin{array}{ccc}
\underbrace{\qquad}_{1} & \underbrace{\qquad\qquad}_{d} & \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{n}
\end{array}
\end{array}
$$

This reverts to linear regression instead of 0 away from data.

Last Time: Polynomial Degree and Training vs. Tesing

- As the polynomial degree increases, the training error goes down.
- But training error becomes worse approximation test error.



- Same effect as we decrease variance in Gaussian RBF.
- 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 or small σ^2 in RBFs.
- But complex models can overfit.
- So what do we do???

- There are many possible answers:
  - Model averaging: average over multiple models to decrease variance.
  - Regularization: add a penalty on the complexity of the model.

L2-Regularization

"lambda"

- Standard regularization strategy is L2-regularization:

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

- Intuition: large w_j tend to lead to overfitting (cancel each other).
- So minimize squared error plus penalty on L2-norm of 'w'.
  - This objective balances getting low error vs. having small slope 'w'.
    - You can increase the error if it makes 'w' much smaller.
    - Reduces overfitting.
  - Regularization parameter $\lambda > 0$ controls "strength" of regularization.
    - Large λ puts large penalty on slope.

L2-Regularization

- Standard regularization strategy is L2-regularization:

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

- In terms of fundamental trade-off:
  - Regularization increases training error.
  - Regularization makes training error better approximation of test error.
- How should you choose λ?
  - Theory: as 'n' grows λ should be in the range $O(1)$ to $O(n^{1/2})$.
  - Practice: optimize validation set or cross-validation error.
    - This almost always decreases the test error.

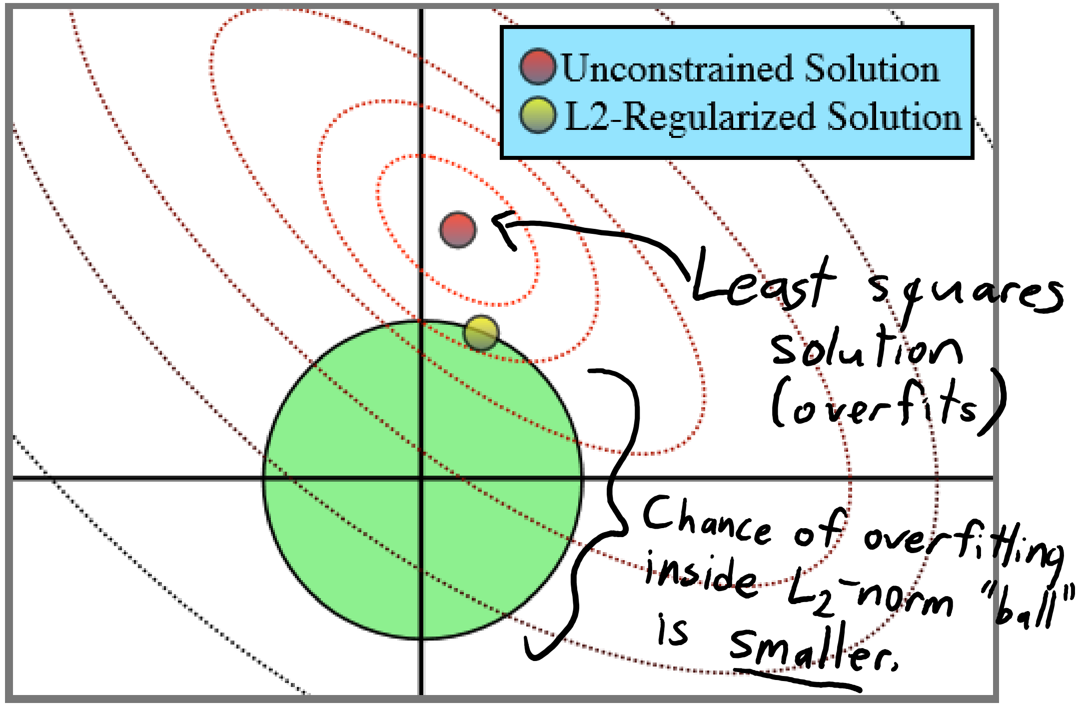
L2-Regularization

- Standard regularization strategy is L2-regularization:

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

- Equivalent to minimizing squared error with L2-norm constraint:

- Connection to Occam's razor

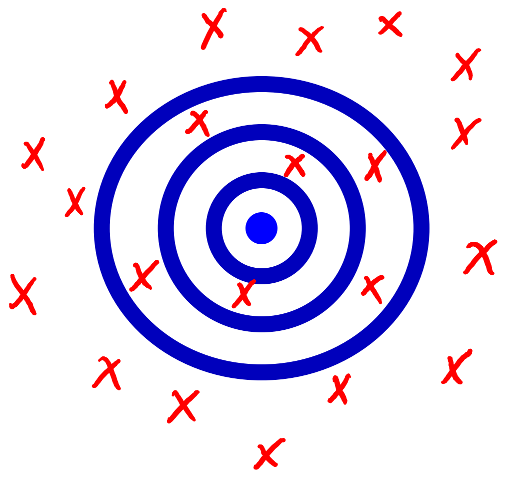


Why use L2-Regularization?

- It's a weird thing to do, but Mark says "always use regularization".
  - "Almost always decreases test error" should already convince you.
- Mike says "try to make the objective function reflect test error"
  - Create an optimization problem that you actually want to solve.
- But here are 6 more reasons:
  1. Solution 'w' is unique.
  2. X^TX does not need to be invertible.
  3. Less sensitive to changes in X or y.
  4. Makes algorithms for computing 'w' converge faster.
  5. Stein's paradox: if $d \geq 3$, 'shrinking' moves us closer to 'true' w.
  6. Worst case: just set λ small and get the same performance.

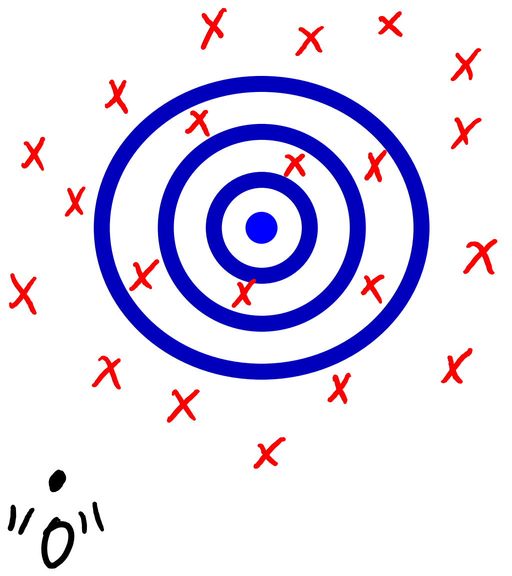
Shrinking is Weird and Magical

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



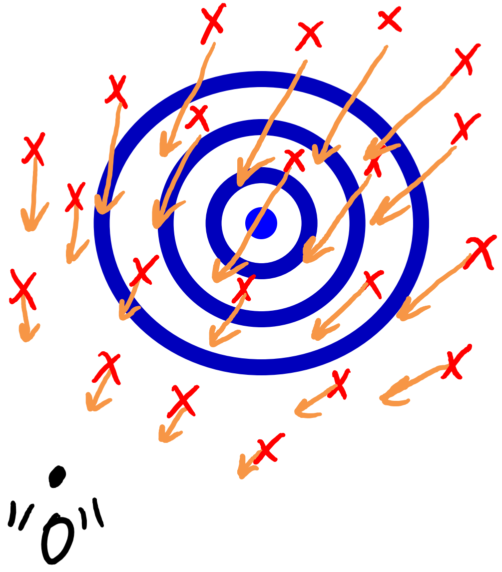
Shrinking is Weird and Magical

- 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'.



Shrinking is Weird and Magical

- 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 distances.
- If small enough, darts will be closer to center on average.



Visualization of the related Stein's paradox:
https://www.naftaliharris.com/blog/steinviz

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!

Example:

Find regularized value of 'w' for particular λ and σ by

minimizing $f(w) = \frac{1}{2}\|Zw - y\|^2 + \frac{\lambda}{2}\|w\|^2$

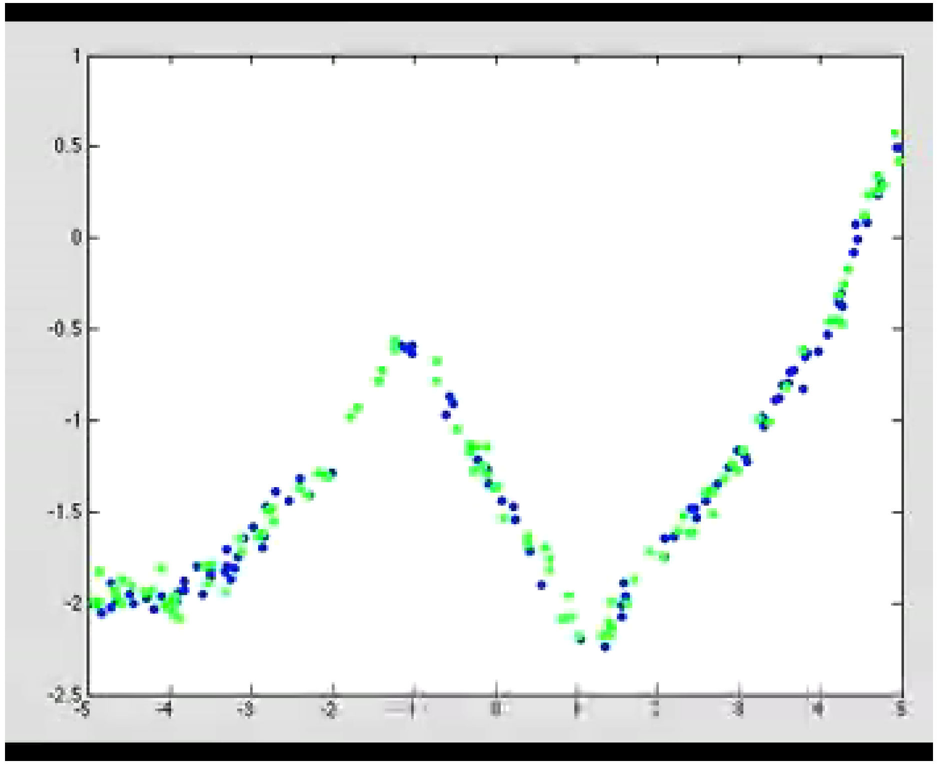
RBF basis with variance σ

And choose λ and σ to minimize

$\frac{1}{2}\|\hat{Z}\hat{w} - \hat{y}\|^2$

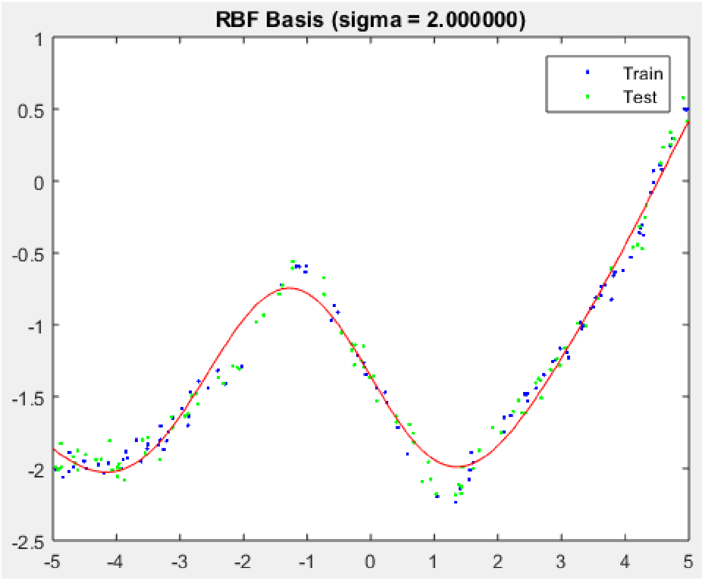
Validation set

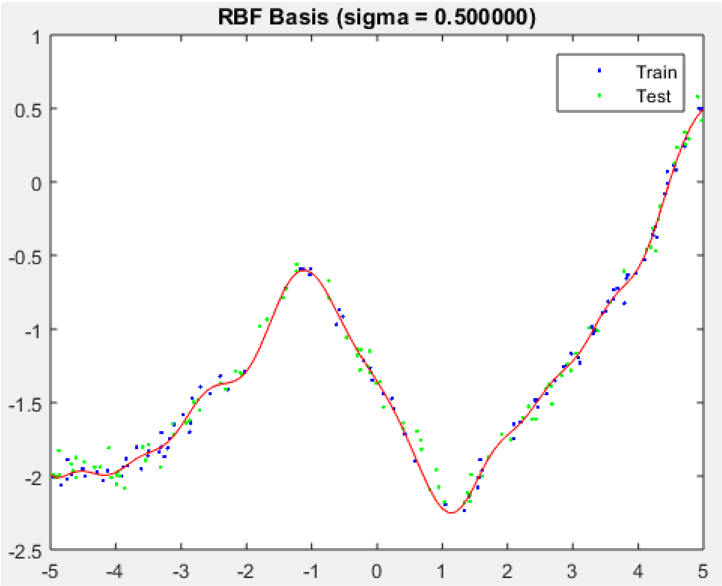
Regularized value of w

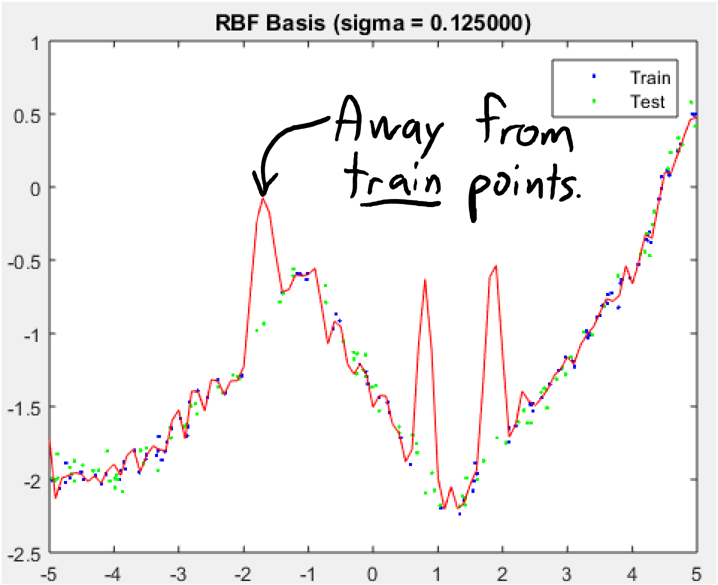


RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!







  - Can add bias or linear/poly basis to do better away from data.
  - Expensive at test time: need distance to all training examples.

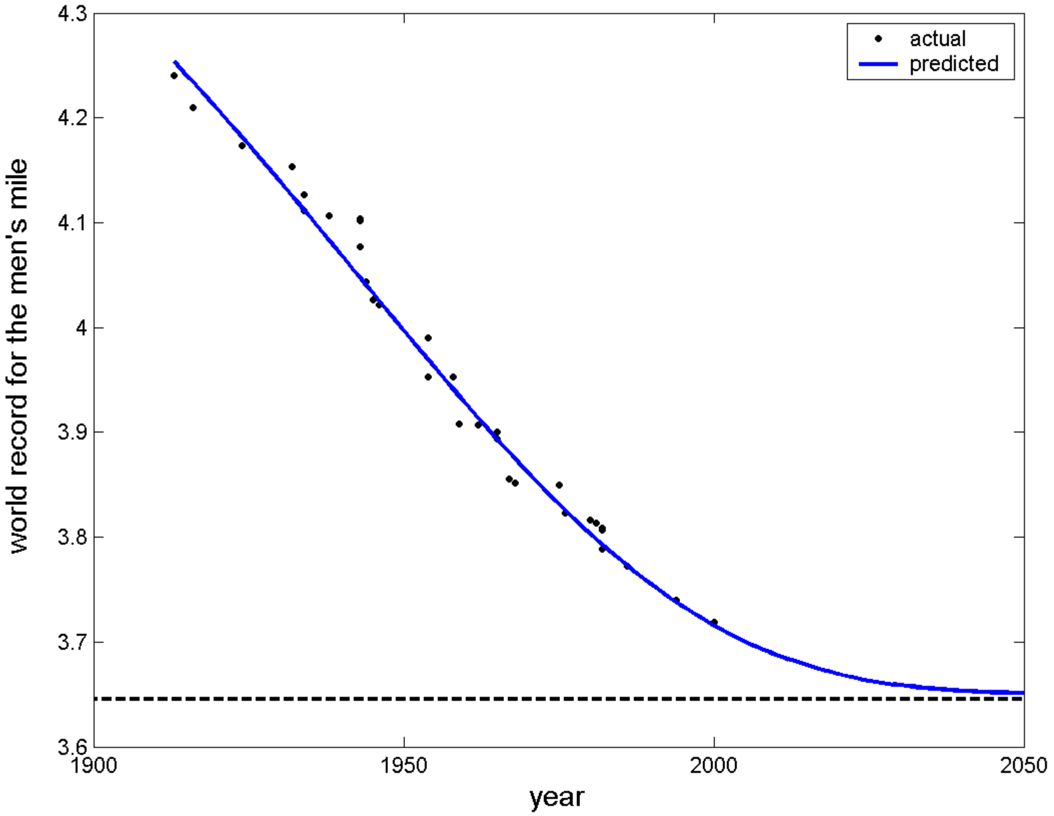
Summary

- Radial basis functions:
  - Non-parametric bases that can model any function.
- Regularization:
  - Adding a penalty on model complexity.
  - Improves test error because it is magic.
- L2-regularization: penalty on L2-norm of regression weights ‘w’.

- Next time:
  - The most important algorithm in machine learning.

Bonus Slide: Predicting the Future

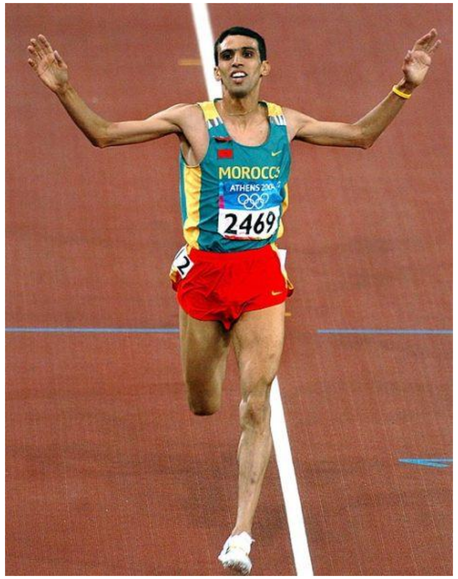
- In principle, we can use any features x_i that we think are relevant.
- This makes it tempting to use time as a feature, and predict future.

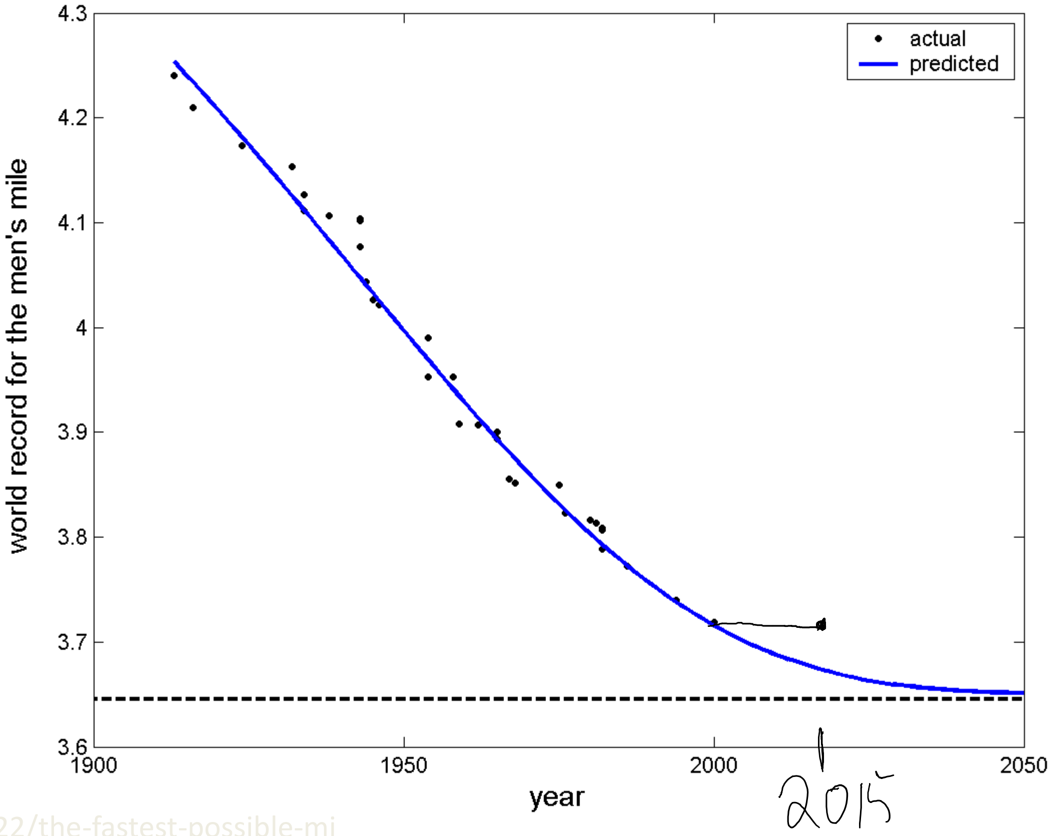


https://gravityandlevity.wordpress.com/2009/04/22/the-fastest-possible-mile/

Bonus Slide: 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.

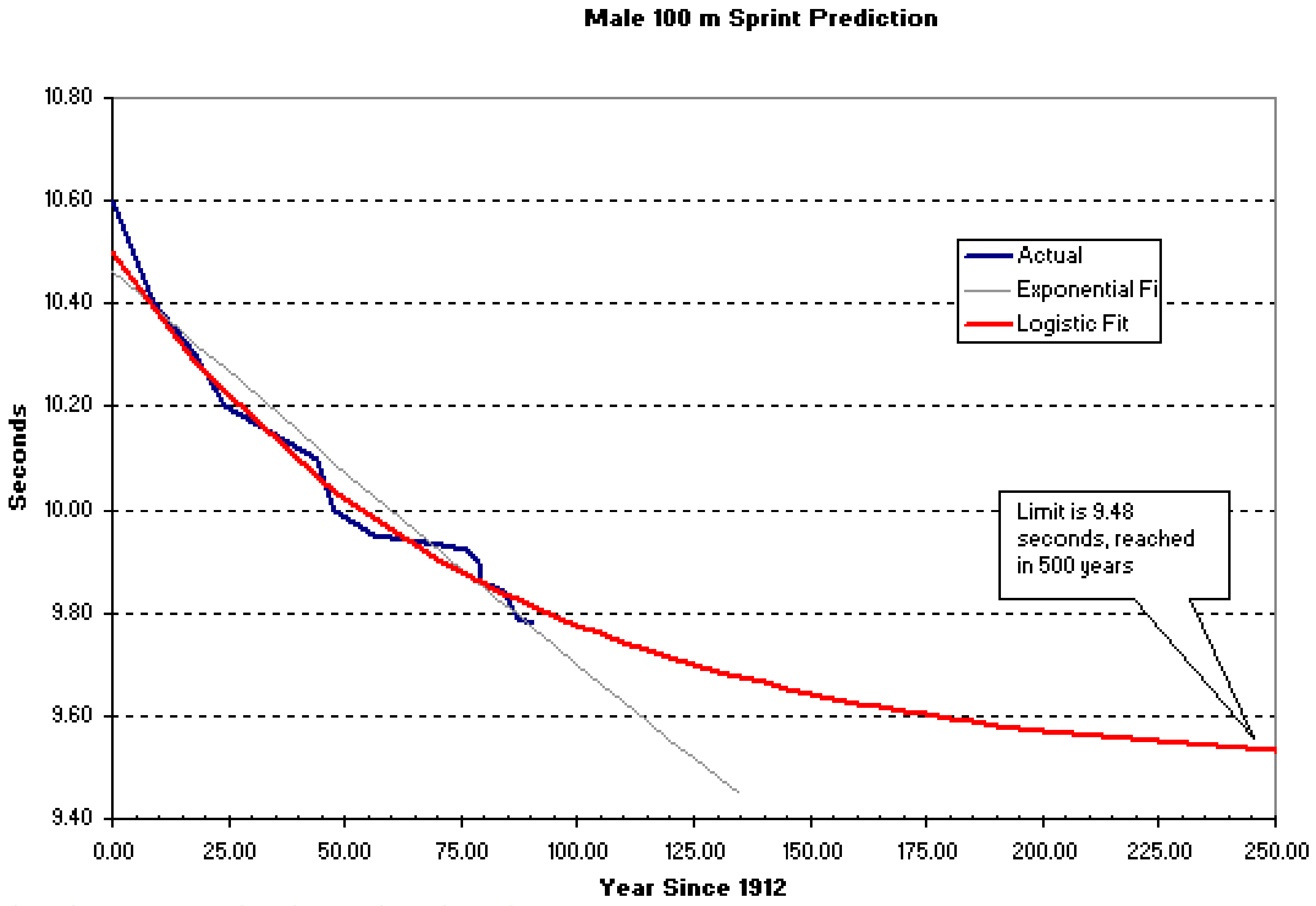




We need to be cautious about doing this.

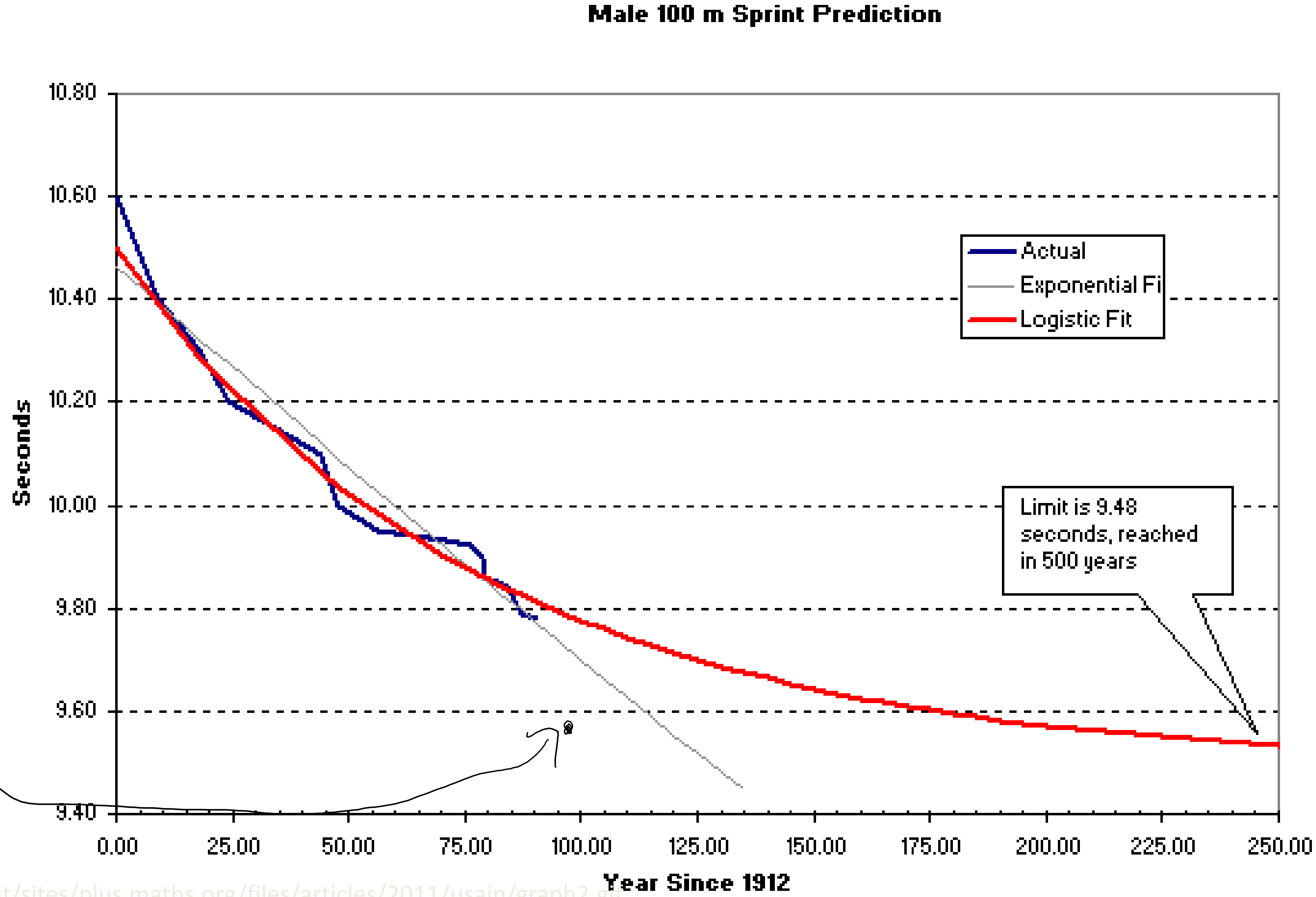
https://gravityandlevity.wordpress.com/2009/04/22/the-fastest-possible-mi
https://overthehillsports.wordpress.com/tag/hicham-el-guerrouj/le/

Bonus Slide: Predicting 100m times 400 years in the future?



https://plus.maths.org/content/sites/plus.maths.org/files/articles/2011/usain/graph2.gif

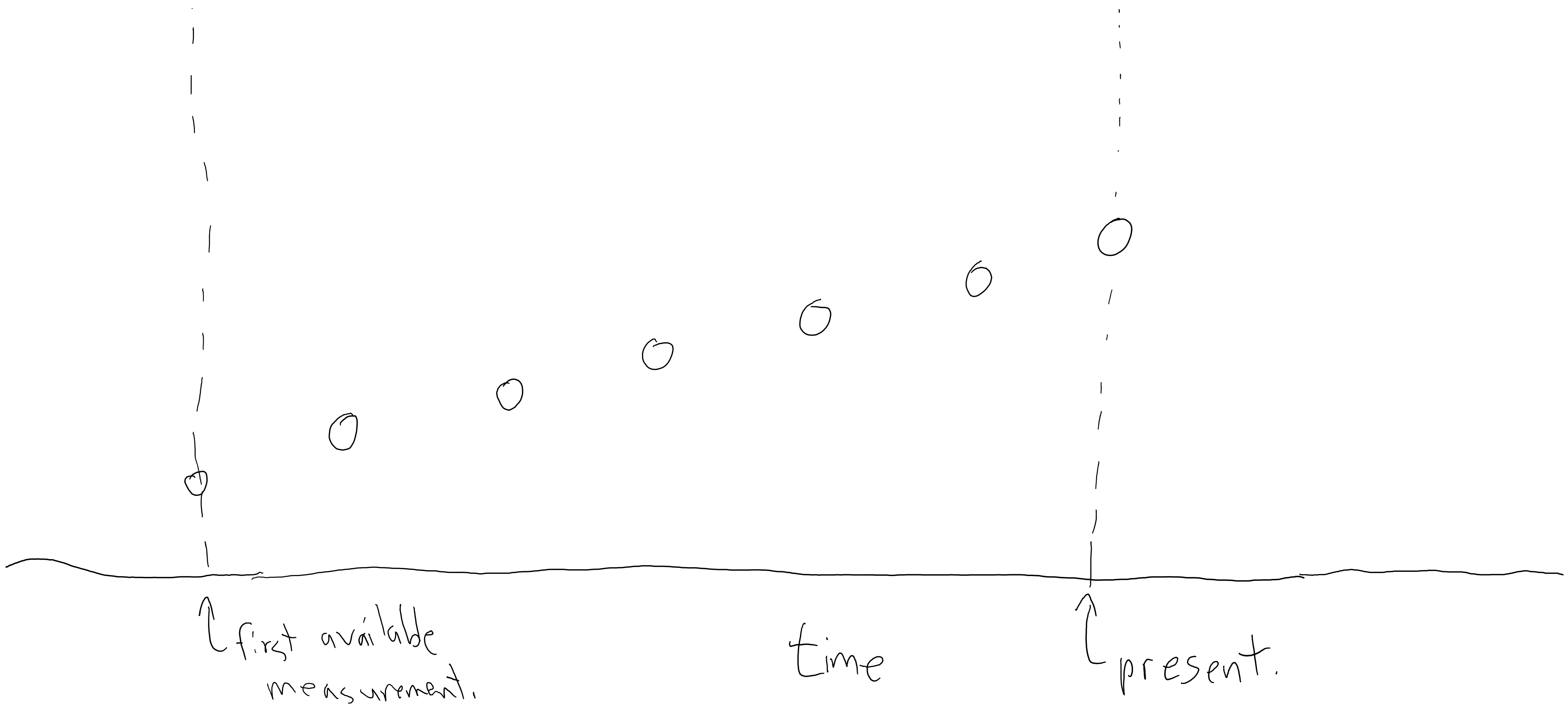
Bonus Slide: Predicting 100m times 400 years in the future?



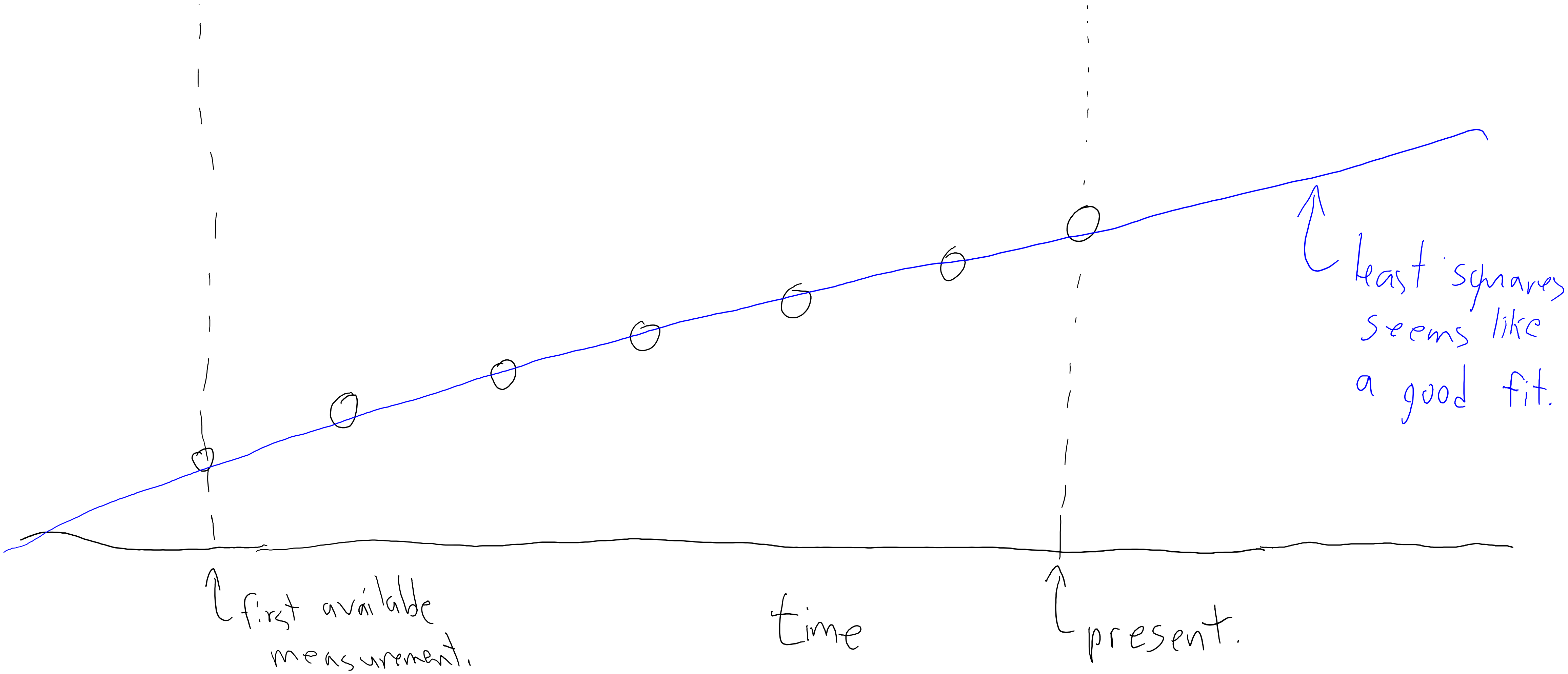
9.58

https://plus.maths.org/content/sites/plus.maths.org/files/articles/2011/usain/graph2.gif
http://www.washingtonpost.com/blogs/london-2012-olympics/wp/2012/08/08/report-usain-bolt-invited-to-tryout-for-manchester-united/

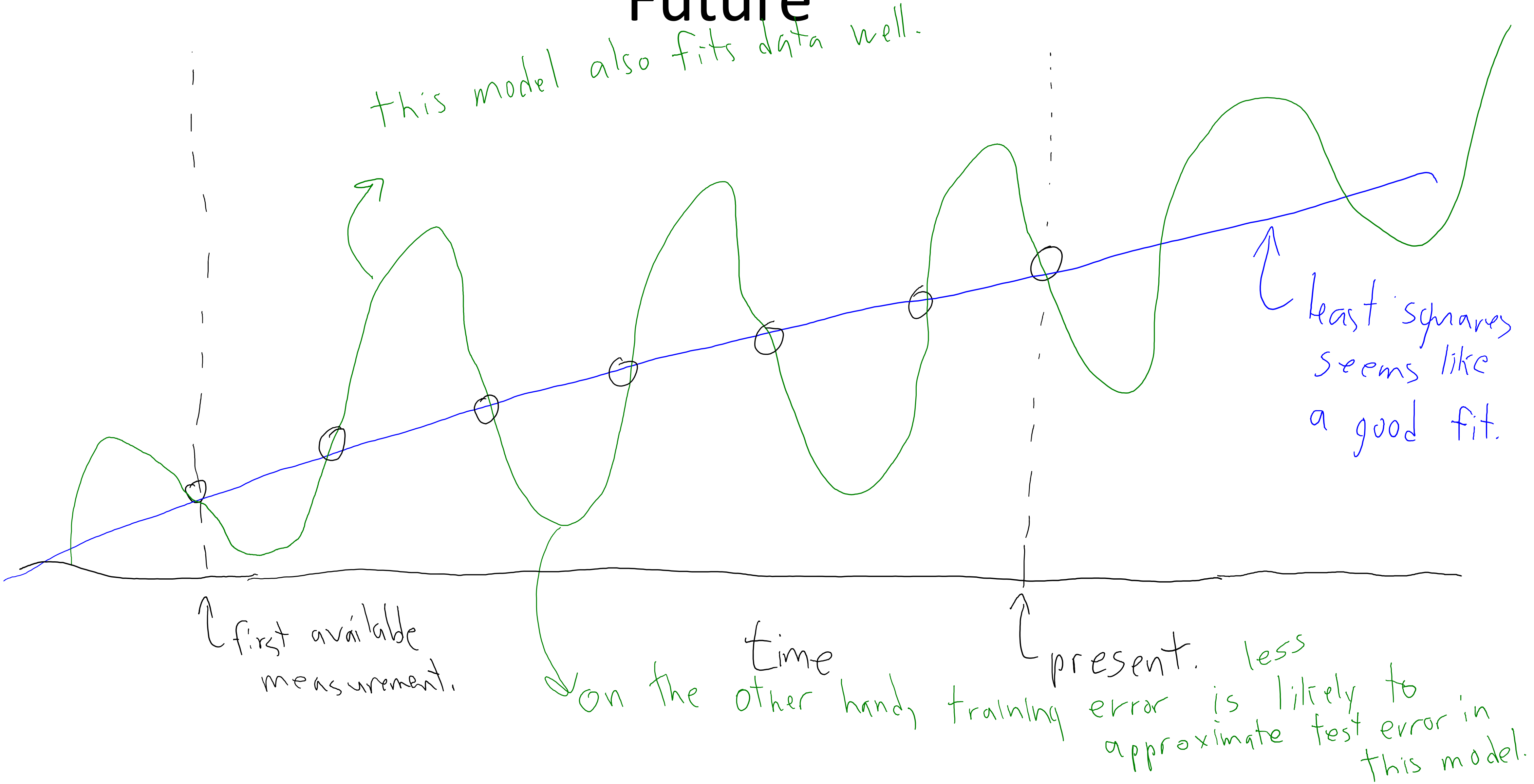
Bonus Slide: No Free Lunch, Consistency, and the Future



Bonus Slide: No Free Lunch, Consistency, and the Future

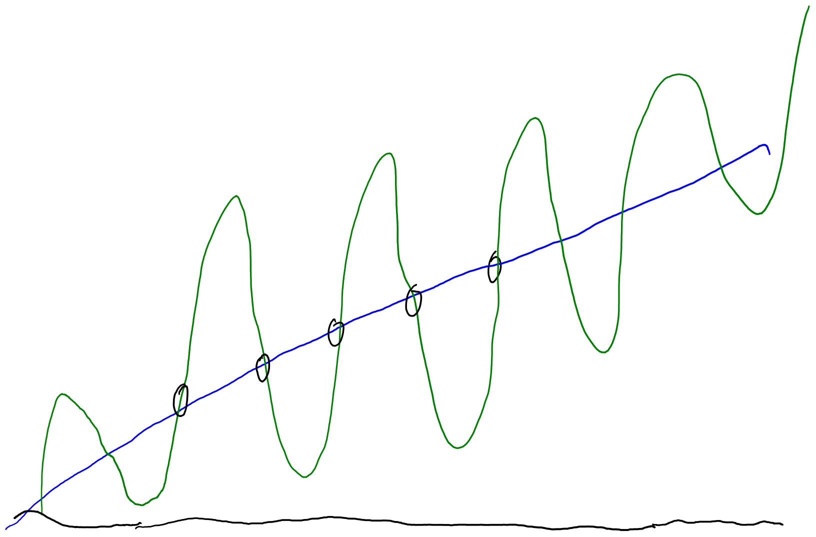


Bonus Slide: No Free Lunch, Consistency, and the Future

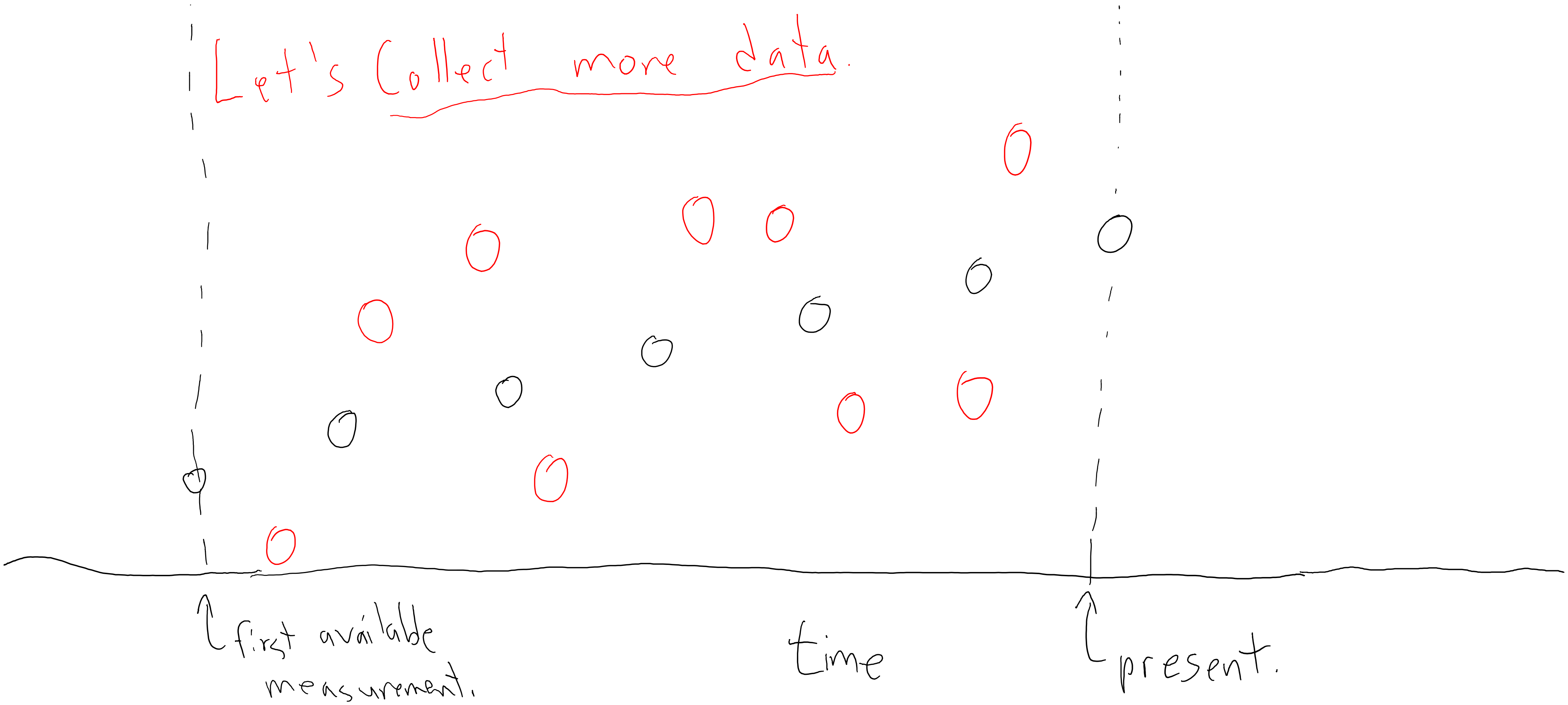


Bonus Slide: Ockham's Razor vs. No Free Lunch

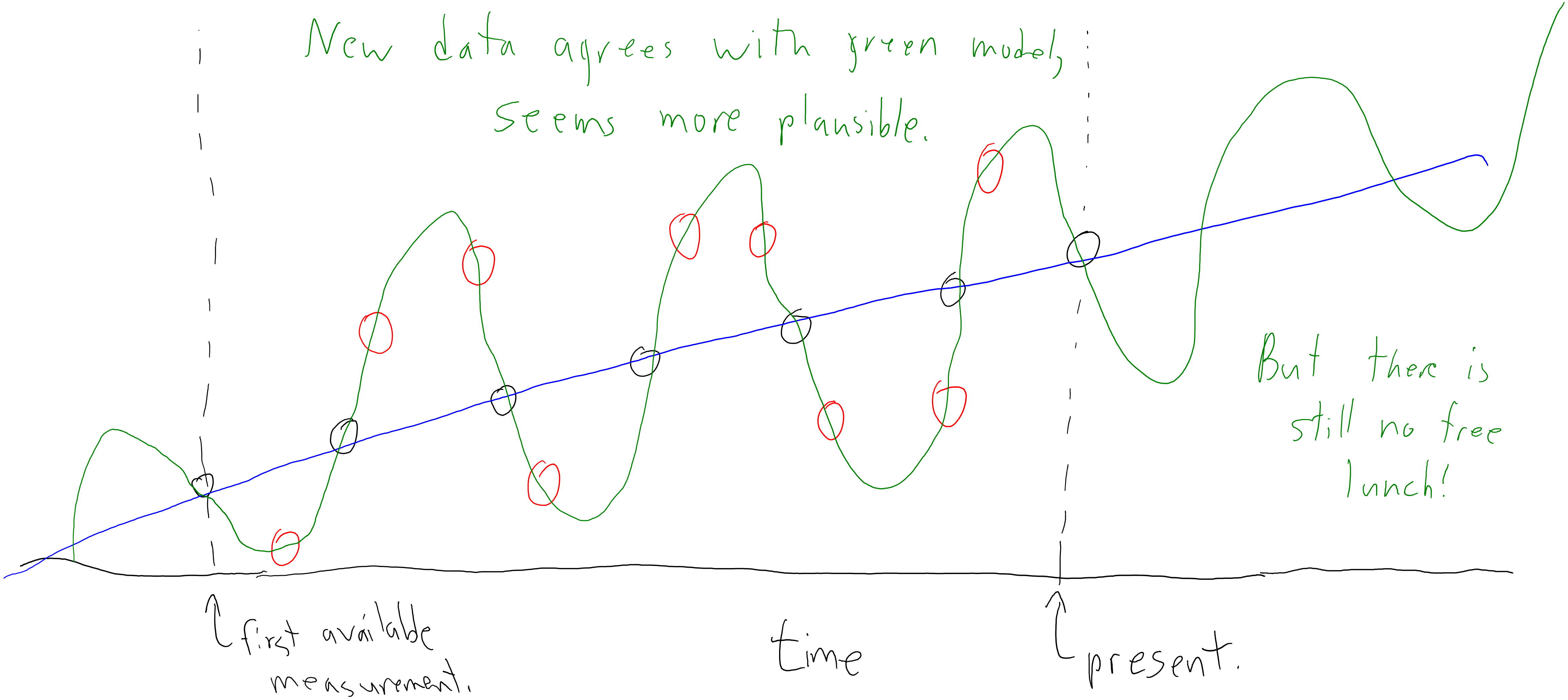
- Ockham's razor is a problem-solving principle:
  - "Among competing hypotheses, the one with the fewest assumptions should be selected."
  - Suggests we should select linear model.
- Fundamental theorem of ML:
  - If training same error, pick model less likely to overfit.
  - Formal version of Occam's 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.



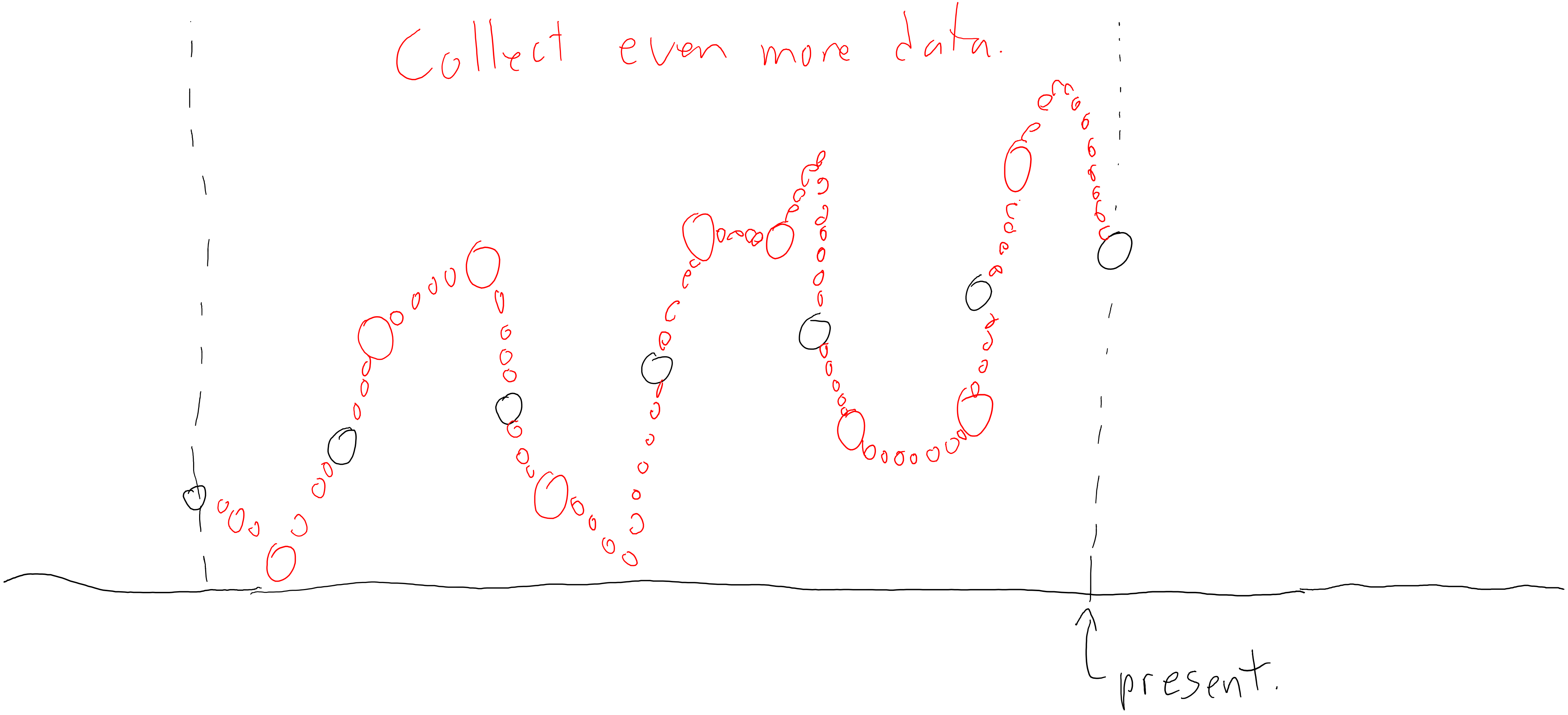
Bonus Slide: No Free Lunch, Consistency, and the Future



Bonus Slide: No Free Lunch, Consistency, and the Future

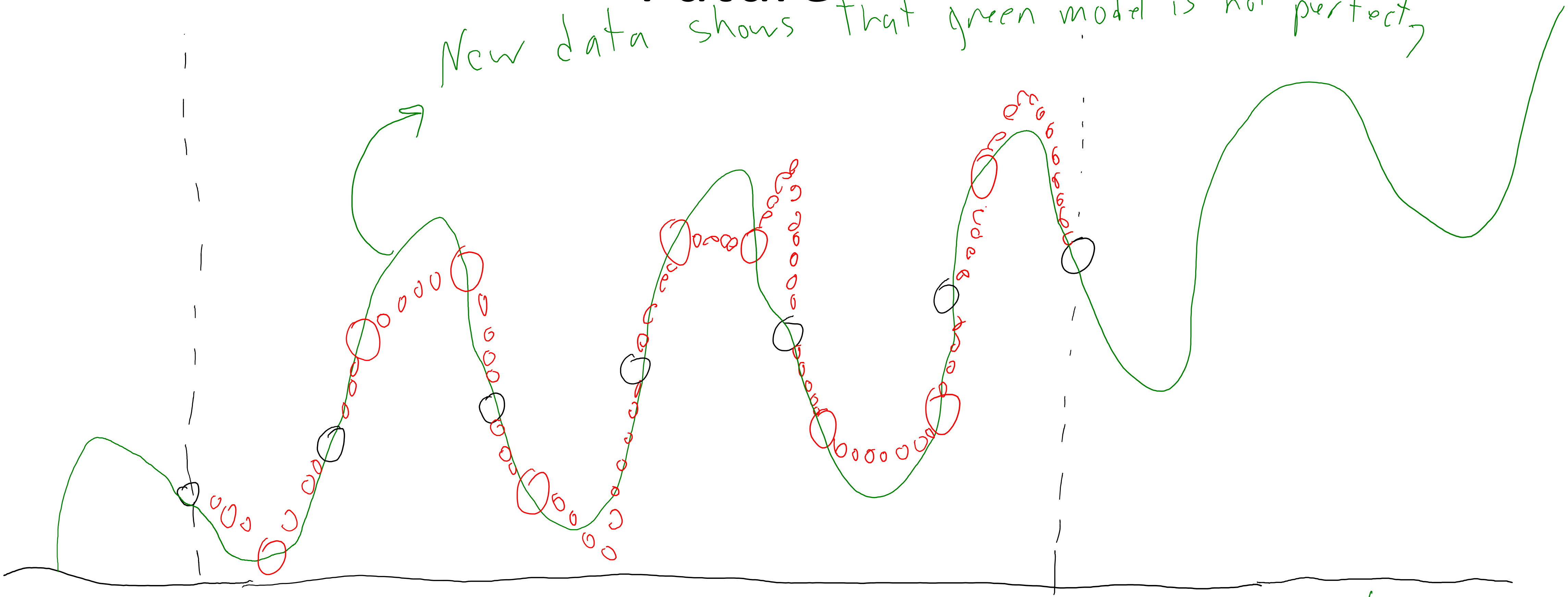


Bonus Slide: No Free Lunch, Consistency, and the Future



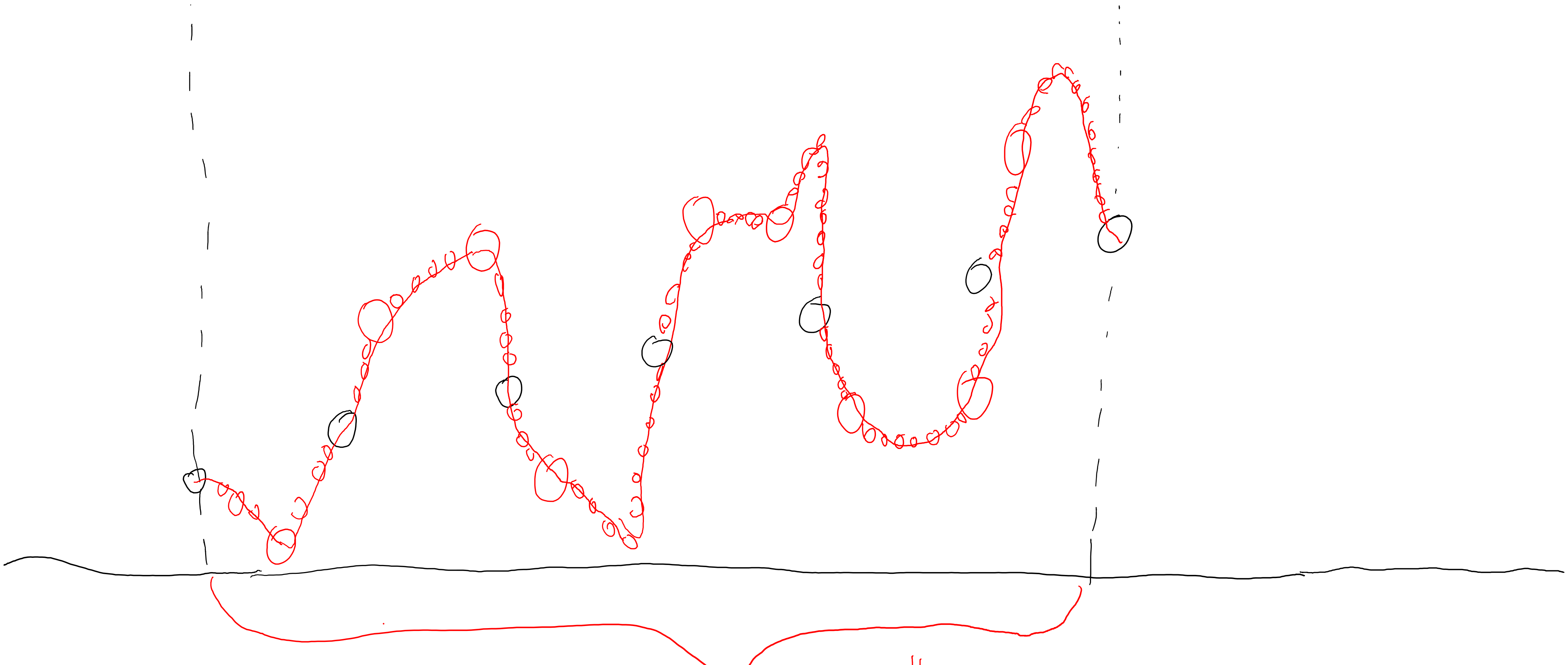
Bonus Slide: No Free Lunch, Consistency, and the Future

New data shows that green model is not perfect,



But, with universally consistent estimator, we can use more data to get closer to true model.

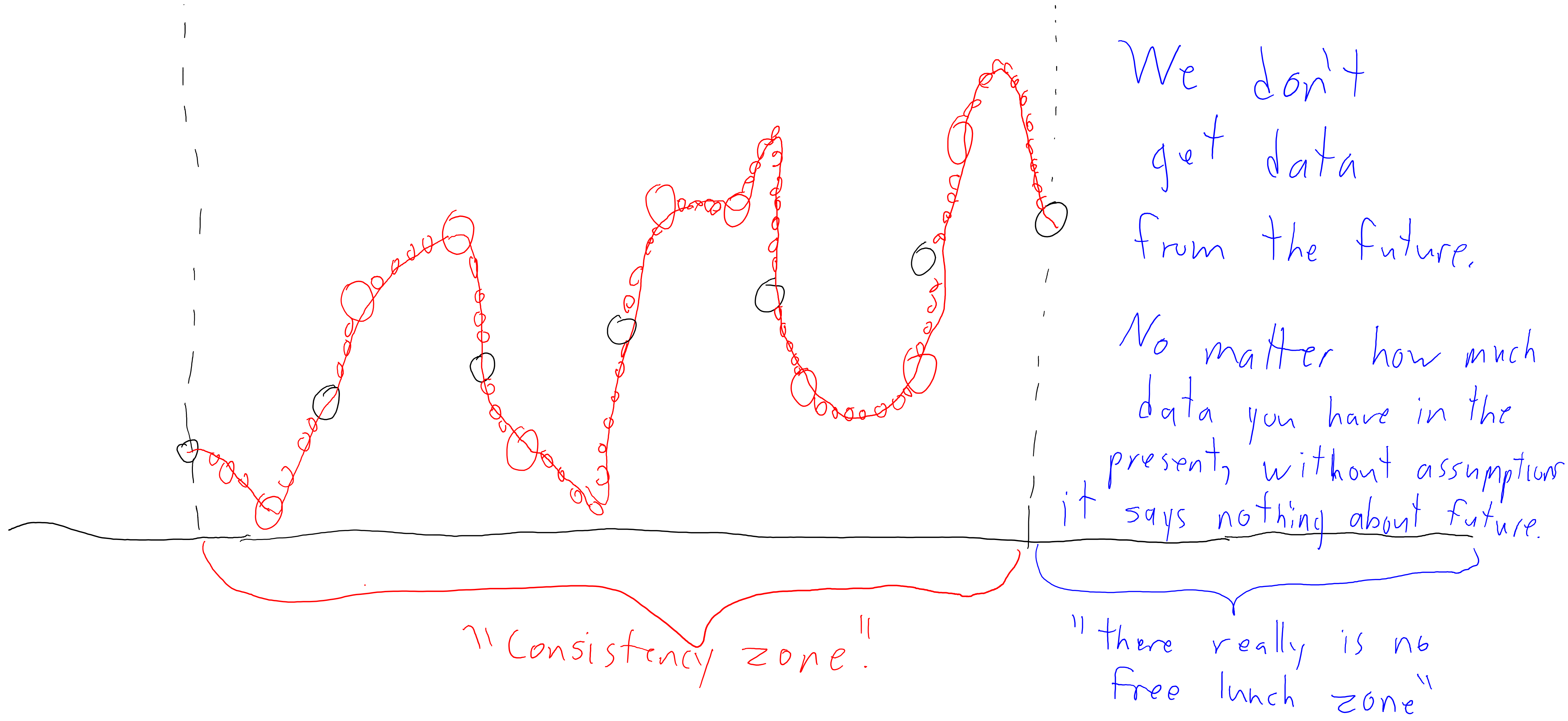
Bonus Slide: No Free Lunch, Consistency, and the Future



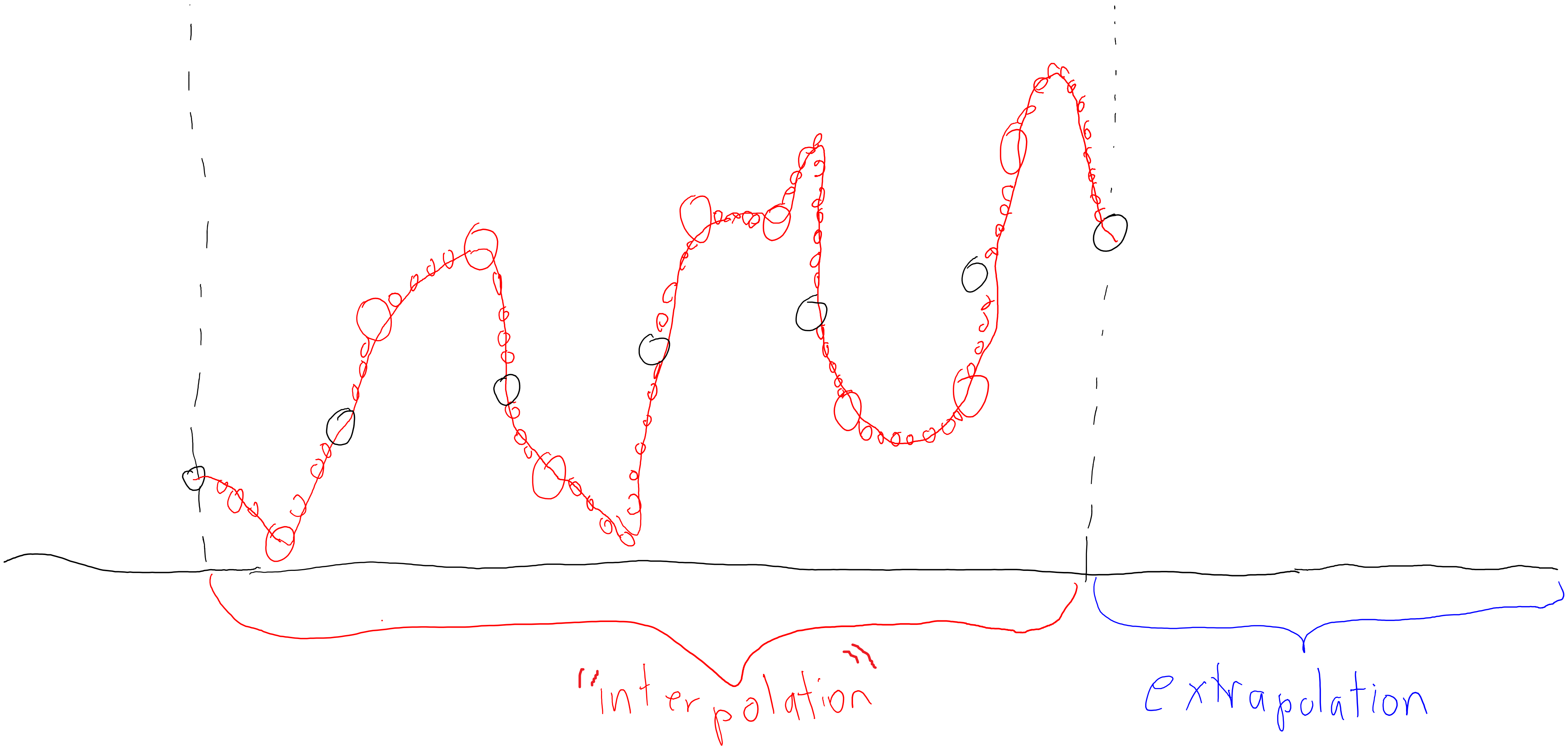
"Consistency zone"

Converge to best model as $n \to \infty$, if we use a "universally consistent" method.

Bonus Slide: No Free Lunch, Consistency, and the Future

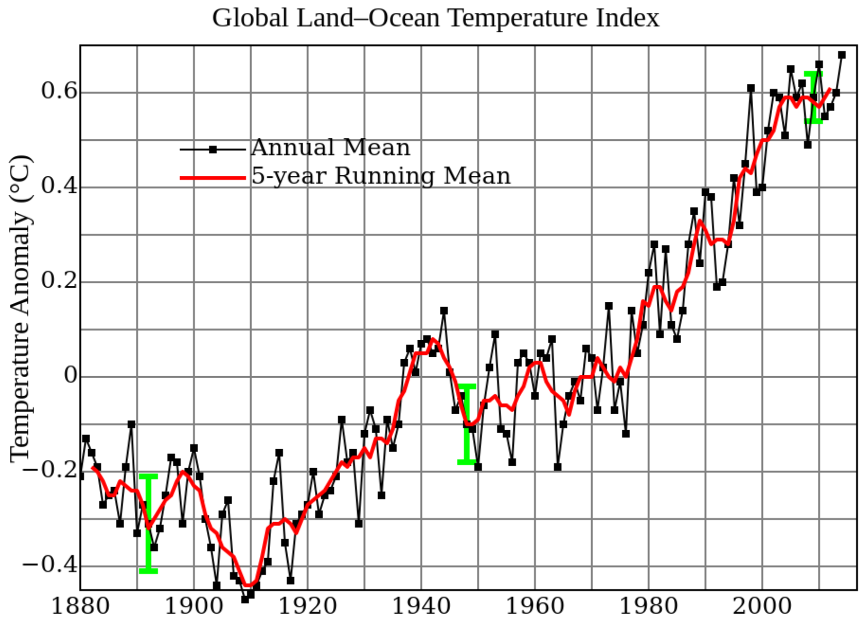


Bonus Slide: No Free Lunch, Consistency, and the Future



Bonus Slide: Application: Climate Models

- Has Earth warmed up over last 100 years? (Consistency zone)
  - Data clearly says ‘yes’.

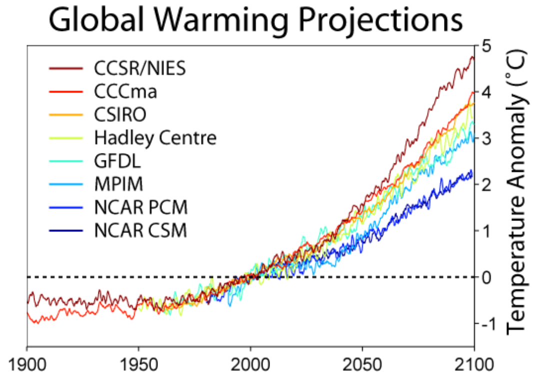


- Will Earth continue to warm over next 100 years? (Really NFL zone)
  - We should be more skeptical about models that predict future events.

https://en.wikipedia.org/wiki/Global_warming

Bonus Slide: Application: Climate Models

- So should we all become global warming skeptics?
- If we average over models that overfit in *indepednent* ways, we expect the test error to be lower, so this gives more confidence:

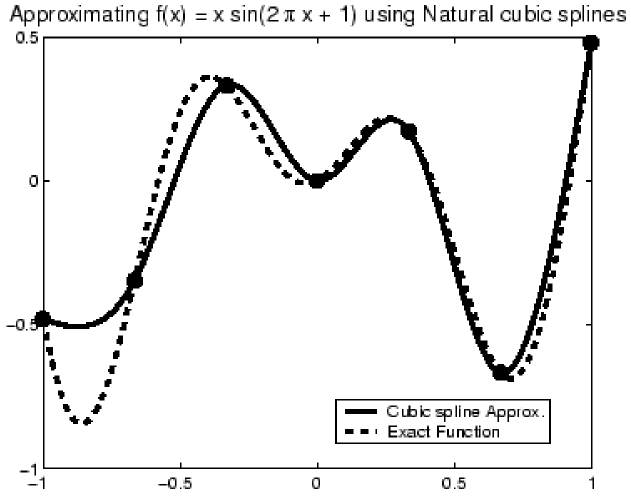


  – We should be skeptical of individual models, but agreeing predictions made by models with different data/assumptions are more likely be true.
- If all near-future predictions agree, they are likely to be accurate.
- As we go further in the future, variance of average will be higher.

https://en.wikipedia.org/wiki/Global_warming

Bonus Slide: Splines in 1D

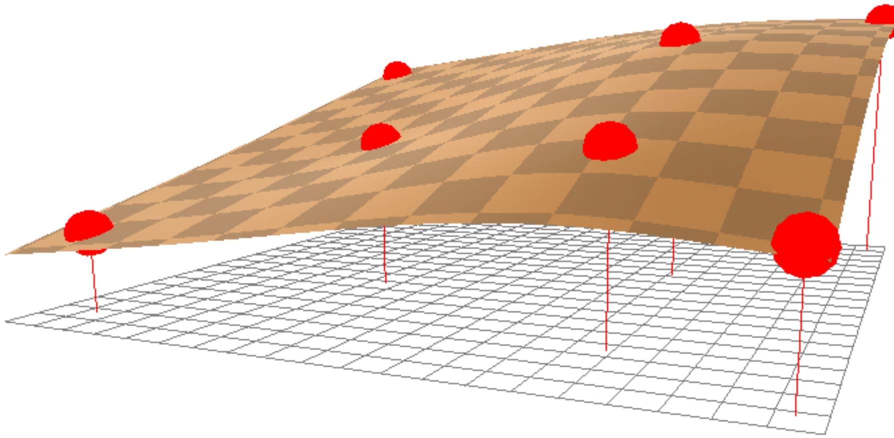
- For 1D interpolation, alternative to polynomials/RBFs are splines:
  - Use a polynomial in the region between each data point.
  - Constrain some derivatives of the polynomials to yield a unique solution.
- Most common example is cubic spline:
  - Use a degree-3 polynomial between each pair of points.
  - Enforce that f'(x) and f''(x) of polynomials agree at all point.
  - "Natural" spline also enforces f''(x) = 0 for smallest and largest x.
- Non-trivial fact: natural cubic splines are sum of:
  - Y-intercept.
  - Linear basis.
  - RBFs with $g(\alpha) = \alpha^3$.
    - Different than Gaussian RBF because it *increases with distance*.



http://www.physics.arizona.edu/~restrepo/475A/Notes/sourcea-/node35.html

Bonus Slide: Spline in Higher Dimensions

- Splines generalize to higher dimensions if data lies on a grid.
  - For more general ("scattered") data, there isn't a natural generalization.
- Common 2D "scattered" data interpolation is thin-plate splines:
  - Based on curve made when bending sheets of metal.
  - Corresponds to RBFs with $g(\alpha) = \alpha^2 \log(\alpha)$.
- Natural splines and thin-plate splines: special cases of "polyharmonic" splines:
  - Less sensitive to parameters than Gaussian RBF.



http://step.polymtl.ca/~rv101/thinplates/