CPSC 340: Machine Learning and Data Mining

More Regularization

Fall 2017

Admin

- Assignment 3:
  - Out soon, due Friday of next week.

- Midterm:
  - You can view your exam during instructor office hours or after class Friday.
    - But no instructor office hours this week (Mark is away).

Last Time: L2-Regularization

- We discussed regularization:
  - Adding a continuous penalty on the model complexity:

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

  - Best parameter λ almost always leads to improved test error.
    - L2-regularized least squares is also known as "ridge regression".
    - Can be solved as a linear system like least squares.
  - Numerous other benefits:
    - Solution is unique, less sensitive to data, gradient descent converges faster.

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*(100 mL) gives the same model as w_j*(0.1 L) with a different w_j.

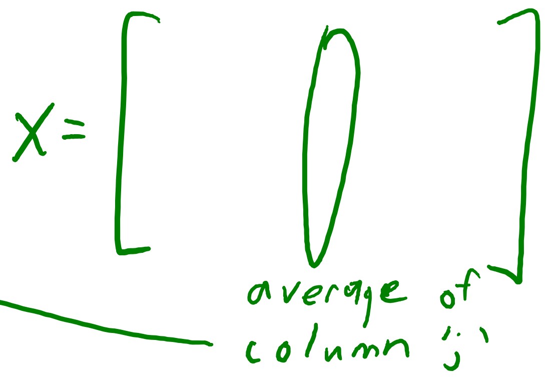
Features with Different Scales

- 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



- It is common to standardize continuous features:
  - For each feature:
    1. Compute mean and standard deviation:

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

    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_j' 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

$X = \left[\quad \bigcirc \quad \right]$

average of column 'j'

- It is common to standardize continuous features:
  - For each feature:
    1. Compute mean and standard deviation:

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

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

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

  - Now changes in 'w_j' have similar effect for any feature 'j'.
- If we're doing 10-fold cross-validation:
  - Compute the μ_j and σ_j based on the 9 training folds.
  - Standardize the remaining ("validation") fold with this "training" μ_j and σ_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:

    Replace y_i 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:

    Use $\log(y_i)$ 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.

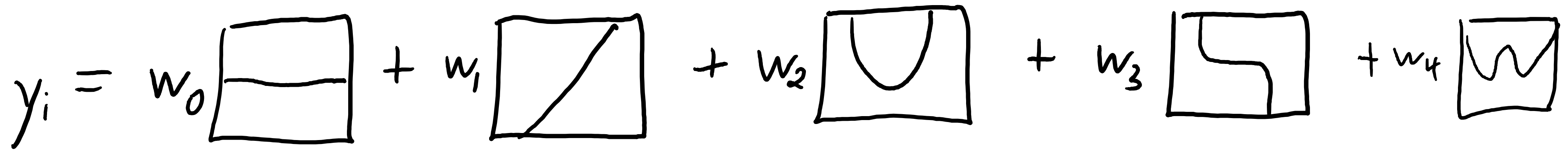
$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{1}{2}\sum_{j=1}^{d} \lambda_j w_j^2$$

Make λ_1 smaller for bias.

(pause)

Parametric vs. Non-Parametric Transforms

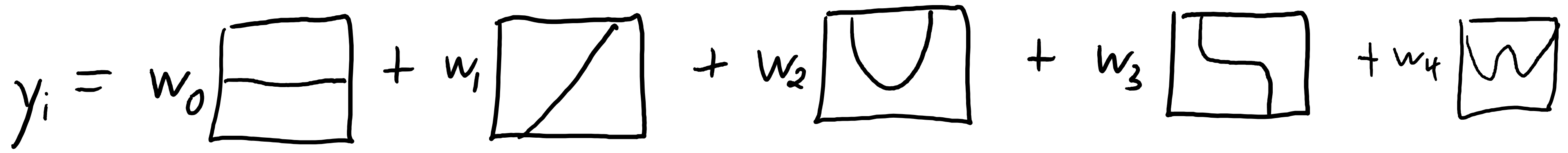
- We've been using linear models with polynomial bases:



- But polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.
  - If we use the wrong basis, our accuracy is limited even with lots of data.
  - But the right basis may not be obvious.

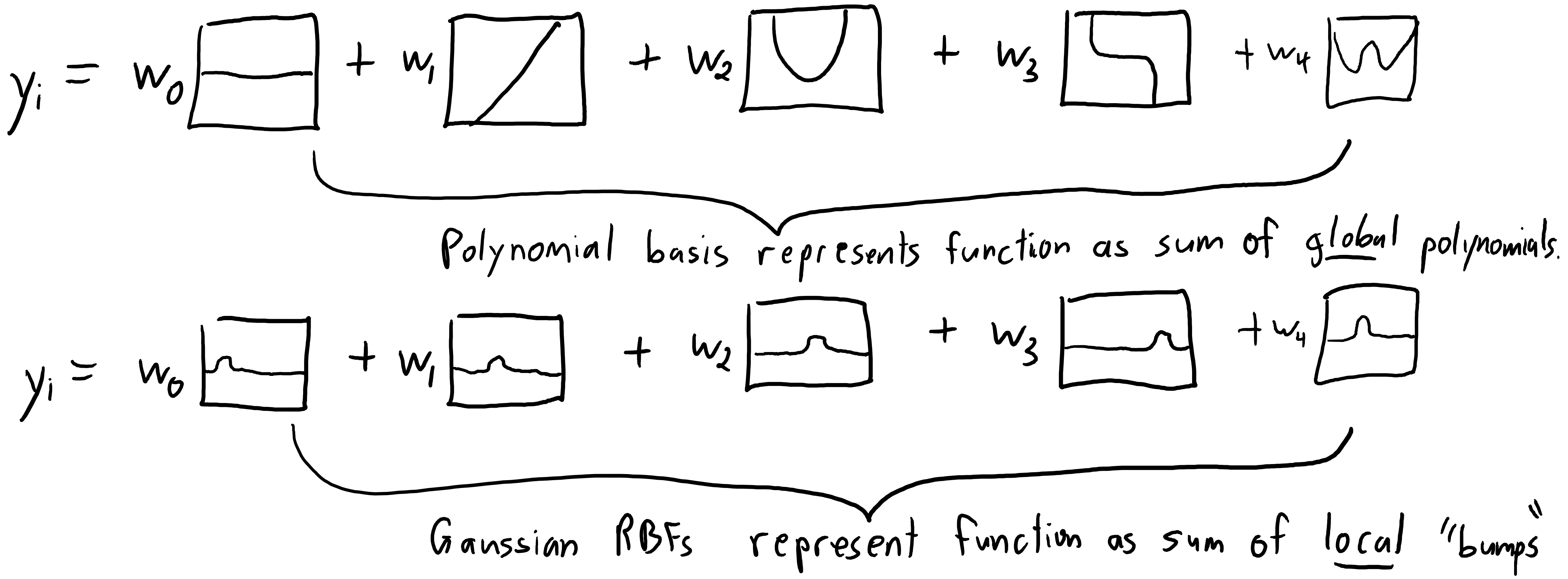
Parametric vs. Non-Parametric Transforms

- We’ve been using linear models with polynomial bases:



- Alternative is non-parametric bases:
  - Size of basis (number of features) grows with ‘n’.
  - Model gets more complicated as you get more data.
  - Can model complicated functions where you don’t know the right basis.
    - With enough data.
  - Classic example is “Gaussian RBFs”.

Gaussian RBFs: A Sum of "bumps"


$y_i = w_0 + w_1 + w_2 + w_3 + w_4$
$y_i = w_0 + w_1 + w_2 + w_3 + w_4$

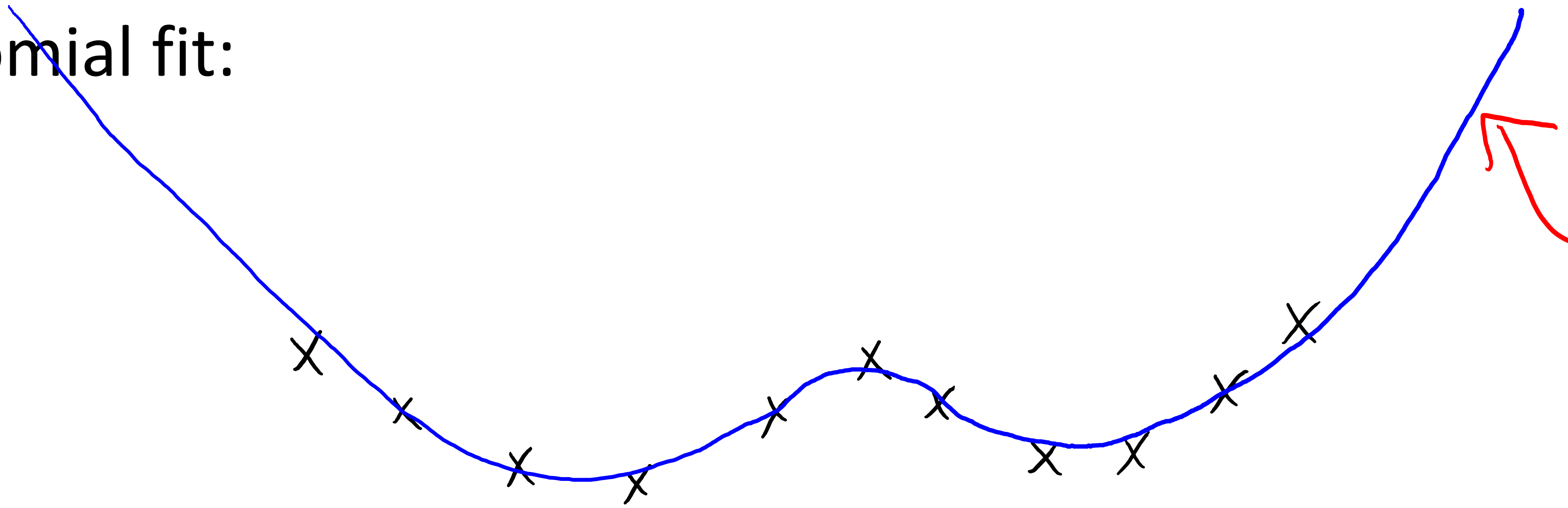
Polynomial basis represents function as sum of <u>global</u> polynomials.

Gaussian RBFs represent function as sum of <u>local</u> "bumps"

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

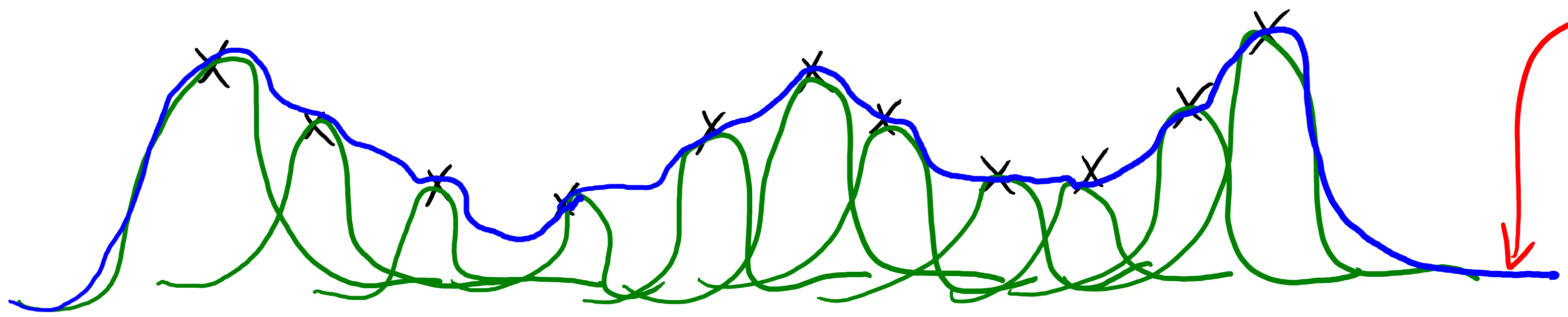
Gaussian RBFs: A Sum of "Bumps"

- Polynomial fit:



polynomial basis becomes polynomial away from data

- Constructing a function from bumps:

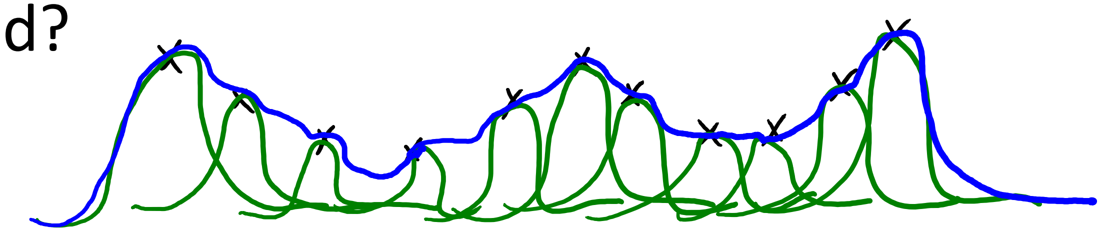


Gaussian RBFs go to zero away from data.

- Bonus slides: challenges of "far from data" (and future) predictions.

Gaussian RBF Parameters

- Some obvious questions:
  1. How many bumps should we use?
  2. Where should the bumps be centered?
  3. How high should the bumps go?
  4. How wide should the bumps be?



- The usual answers:
  1. We use ‘n’ bumps (non-parametric basis).
  2. Each bump is centered on one training example x_i.
  3. Fitting regression weights ‘w’ gives us the heights (and signs).
  4. The width is a hyper-parameter (narrow bumps == complicated model).

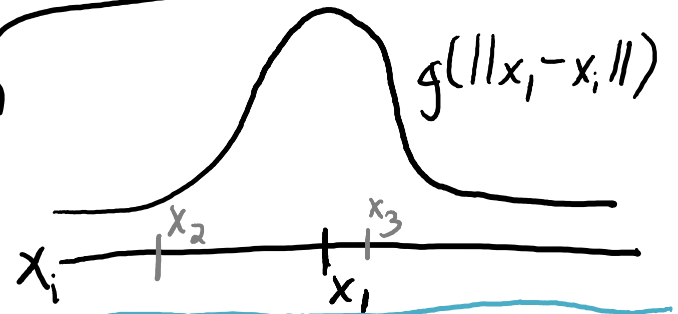
Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

Replace $X = \underbrace{\left[\begin{array}{c} \\ \\ \\ \end{array}\right]}_{d} \Big\} n$ 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:

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



  - Variance σ^2 is a hyper-parameter controlling “width”.
    - This affects fundamental trade-off (set it using a validation set).

Do we need $\sigma\sqrt{2\pi}$?
–No because $v^T z_i = \left(\frac{1}{\beta} v\right)^T (\beta z_i)$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

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

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

Number of "features" is number of training examples.

Gaussian RBFs: Pseudo-Code

Input: data $\{X, y\}$ and hyper-parameters $\{\lambda, \sigma^2\}$

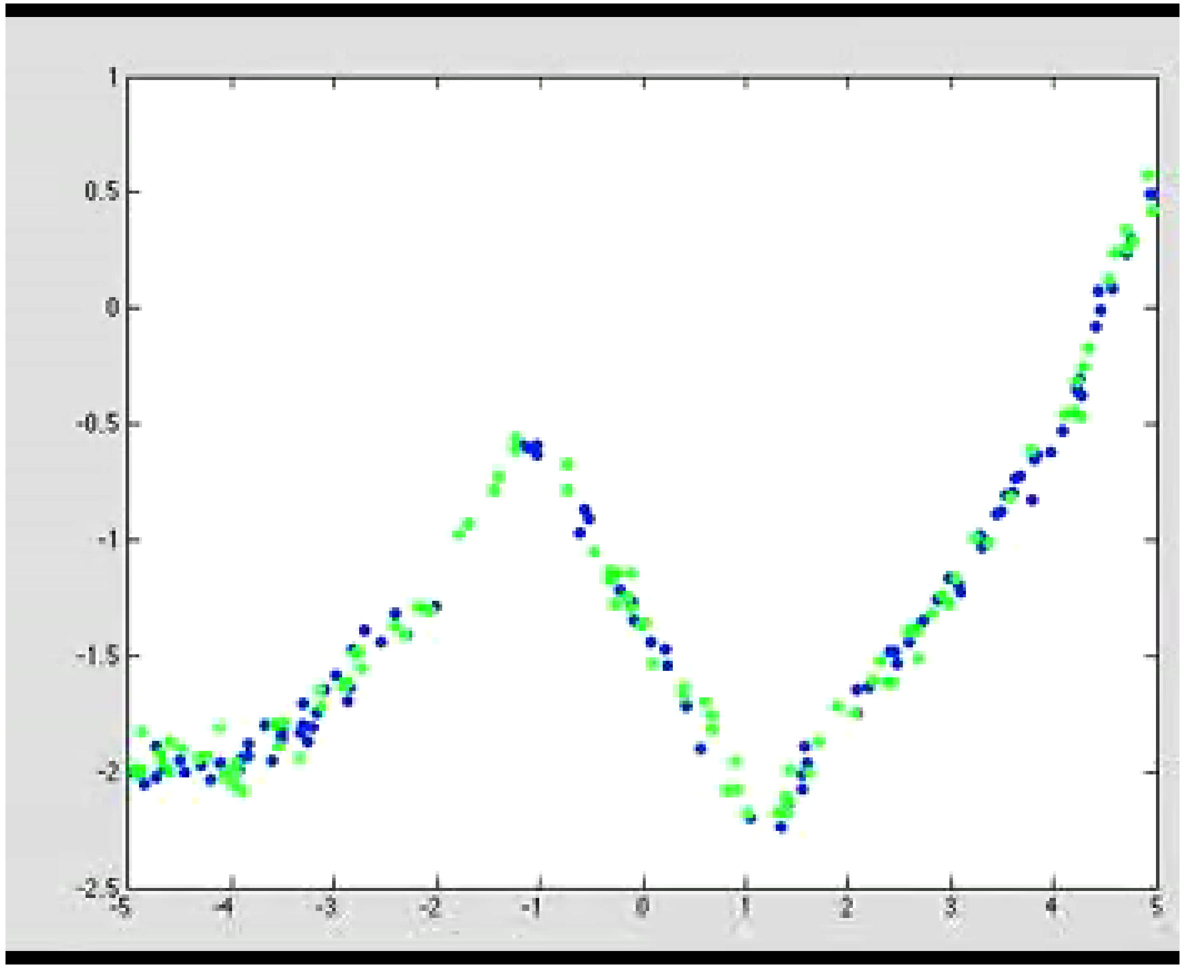
```
  $Z = \text{zeros}(n, n)$
  for i1 in 1:n
      for i2 in 1:n
          $Z[i1, i2] = \exp(-\text{norm}(X[i1,:] - X[i2,:])^2 / 2\sigma^2)$
  $v = (Z^T Z + \lambda I)^{-1} Z^T y$
```

With test data $\hat{X}$: form $\tilde{Z}$ based on distances to training examples.

predict $\hat{y} = \tilde{Z} v$

Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



Could add <u>bias</u> and linear basis:

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

This reverts to <u>linear regression</u> instead of 0 away from data.

(pause)

RBFs and Regularization

- Radial basis functions (RBFs):
  - Basis functions that depend on distances to training points:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

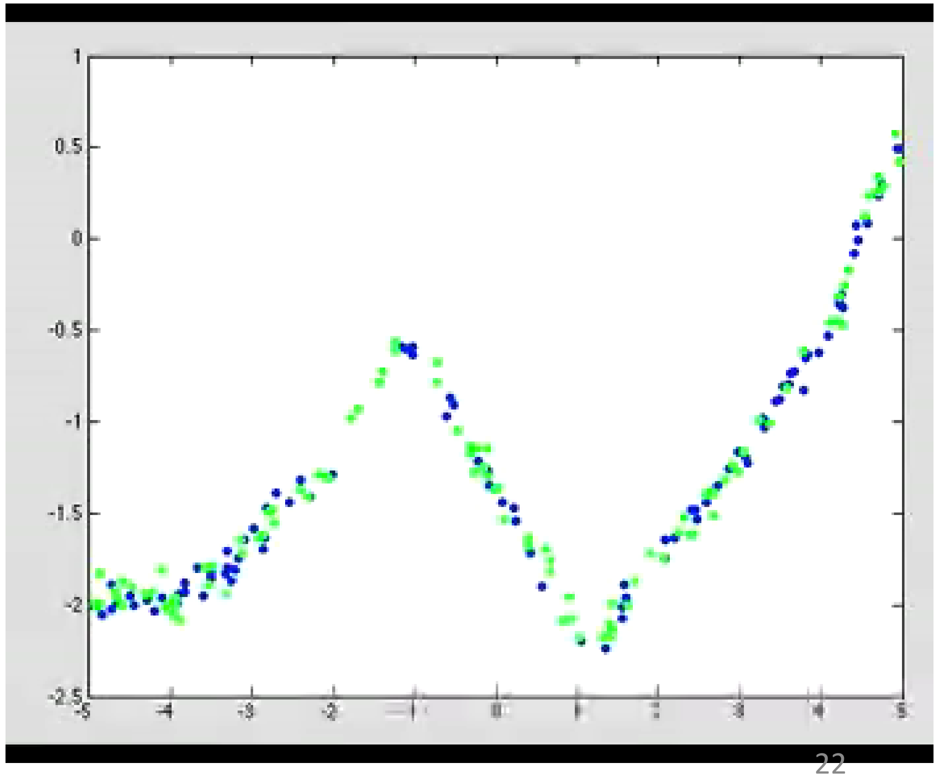
  - Flexible bases that can model any continuous function.
  - But with ‘n’ data points RBFs have ‘n’ basis functions.

- How do we avoid overfitting with this huge number of features?
  - We regularize ‘w’ and use validation error to choose σ and λ.

RBFs, Regularization, and Validation

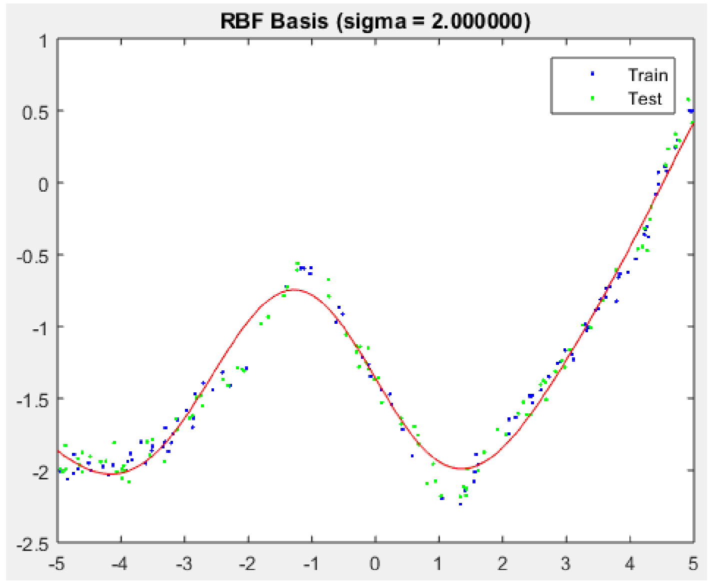
- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!

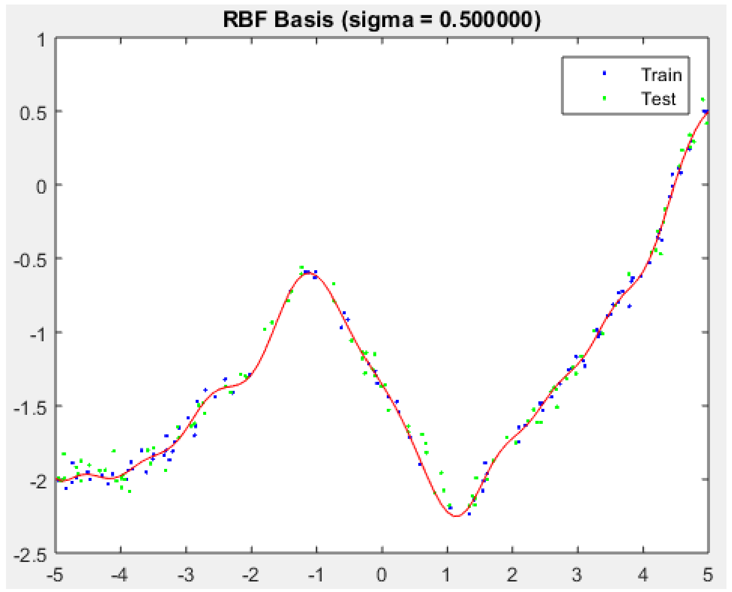
```
for each value of $\lambda$ and $\sigma$:
    - Compute $Z$ on training data (and $\sigma$)
    - Compute best $v$: $v = (Z^TZ + \lambda I)^{-1}Z^Ty$
    - Compute $\tilde{Z}$ on validation data (using train data distances)
    - Make predictions $\hat{y} = \tilde{Z}v$
    - Compute validation error $\|\hat{y} - \tilde{y}\|^2$
```

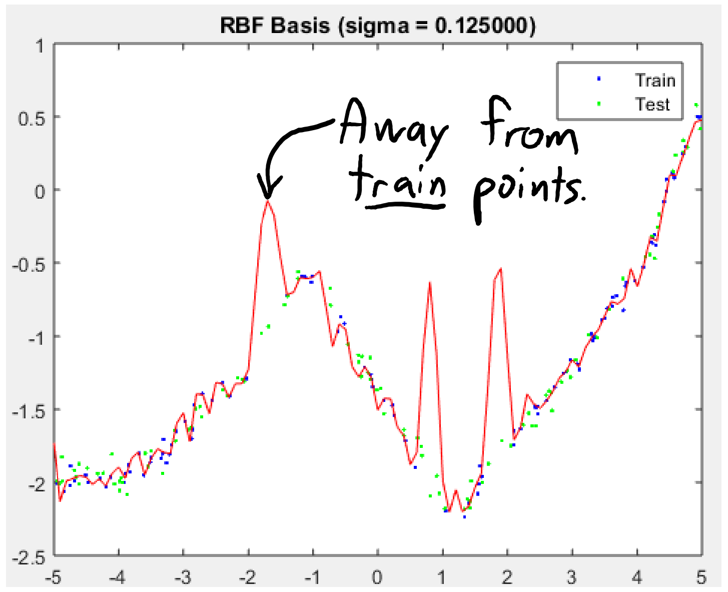


RBFs, Regularization, and Validation

- 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: needs distance to all training examples.

Hyper-Parameter Optimization

- In this setting we have 2 hyper-parameters (σ and λ).
- More complicated models have even more hyper-parameters.
  - This makes searching all values expensive (increases over-fitting risk).

- Leads to the problem of hyper-parameter optimization.
  - Try to efficiently find "best" hyper-parameters.

- Simplest approaches:
  - Exhaustive search: try all combinations among a fixed set of σ and λ values.
  - Random search: try random values.

Hyper-Parameter Optimization

- Other common hyper-parameter optimization methods:
  - Exhaustive search with pruning:
    - If it "looks" like test error is getting worse as you decrease λ, stop decreasing it.
  - Coordinate search:
    - Optimize one hyper-parameter at a time, keeping the others fixed.
    - Repeatedly go through the hyper-parameters
  - Stochastic local search:
    - Generic global optimization methods (simulated annealing, genetic algorithms, etc.).
  - Bayesian optimization (Mike's PhD research topic):
    - Use regression to build model of how hyper-parameters affect validation error.
    - Try the best guess based on the model.

(pause)

Previously: Search and Score

- We talked about search and score for feature selection:
  - Define a "score" and "search" for features with the best score.
- Usual scores count the number of non-zeroes ("L0-norm"):

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda \underbrace{\|w\|_0}_{\text{number of non-zeroes in 'w'}}$$

- But it's hard to find the 'w' minimizing this objective.
- We discussed forward selection, but requires fitting $O(d^2)$ models.
  - For robust regression, need to run gradient descent $O(d^2)$ times.
  - With regularization, need to search for lambda $O(d^2)$ times.

L1-Regularization

- Consider regularizing by the L1-norm:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

- Like L2-norm, it's convex and improves our test error.
- Like L0-norm, it encourages elements of 'w' to be exactly zero.

- L1-regularization simultaneously regularizes and selects features.
  - Very fast alternative to search and score.
  - Sometimes called "LASSO" regularization.

Regularizers and Sparsity

- L1-regularization give sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- Without regularization, we could choose any of these 3.
  - They all have same error, so regularization will "break tie".
- With L0-regularization, we would choose w^2:

$$\|w^1\|_0 = 2 \qquad \|w^2\|_0 = 1 \qquad \|w^3\|_0 = 2$$

Regularizers and Sparsity

- L1-regularization give sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- With L2-regularization, we would choose w^3:

$$\|w^1\|^2 = 100^2 + 0.02^2 = 10000.0004 \qquad \|w^2\|^2 = 100^2 + 0^2 = 10000 \qquad \|w^3\|^2 = 99.99^2 + 0.02^2 = 9998.0005$$

- L2-regularization focuses on decreasing largest (makes w_j similar).

Regularizers and Sparsity

- L1-regularization give sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- With L1-regularization, we would choose w^2:

$$\|w^1\|_1 = 100 + 0.02 = 100.02 \qquad \|w^2\|_1 = 100 + 0 = 100 \qquad \|w^3\|_1 = 99.99 + 0.02 = 100.01$$

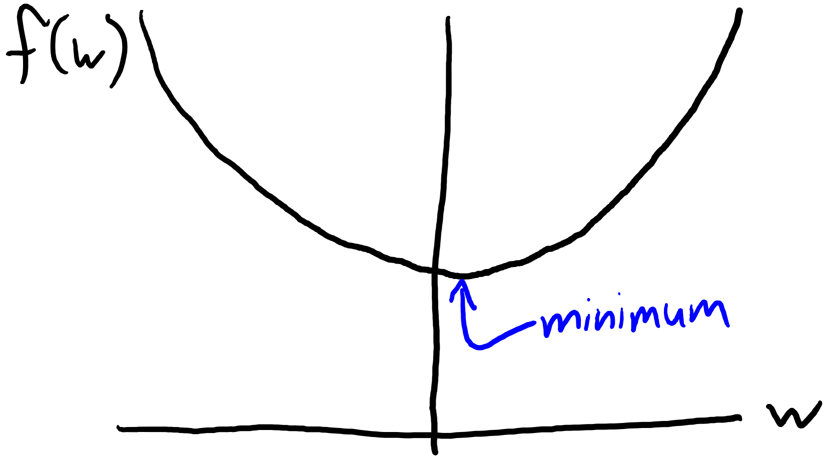
- L1-regularization focuses on decreasing all w_j until they are 0.

Sparsity and Least Squares

- Consider 1D least squares objective:

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

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



- This variable does not look relevant (minimum is close to 0).
  - But for finite 'n' the minimum is unlikely to be exactly zero.

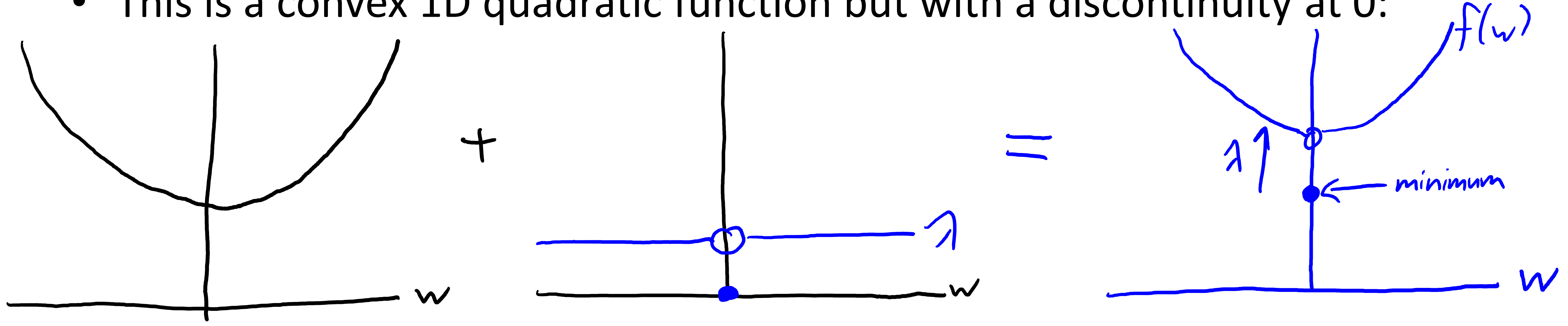
$f'(0) = 0$ only happens if $\sum_{i=1}^{n} y_i x_i = 0$. (bonus)

Sparsity and L0-Regularization

- Consider 1D L0-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \underbrace{\lambda \|w\|_0}_{\begin{cases}\lambda & \text{if } w \neq 0 \\ 0 & \text{if } w = 0\end{cases}}$$

- This is a convex 1D quadratic function but with a discontinuity at 0:



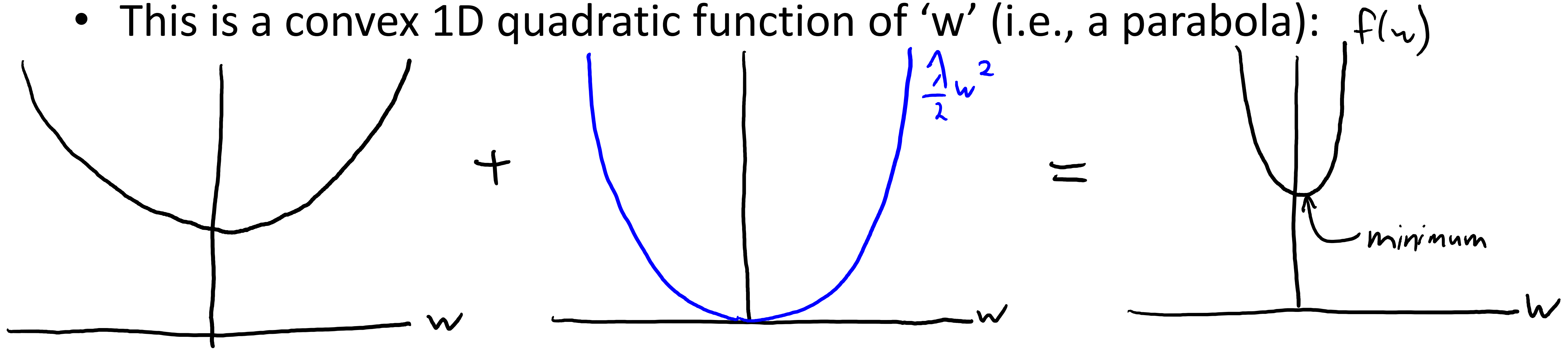
- L0-regularized minimum is often exactly at the ‘discontinuity’ at 0:
  - Sets the feature to exactly 0 (does feature selection), but is non-convex.

Sparsity and L2-Regularization

- Consider 1D L2-regularized least squares objective:

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

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



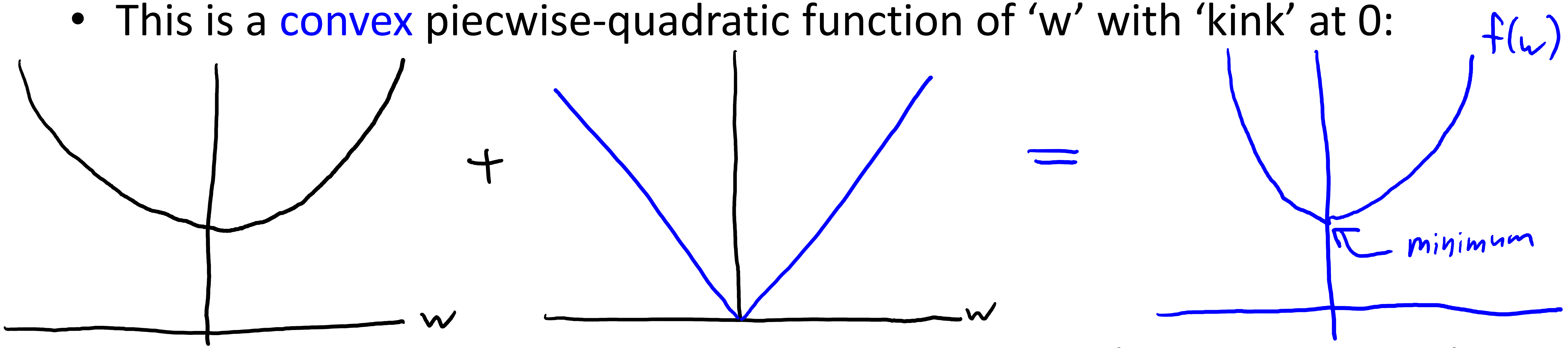
- L2-regularization moves it closer to zero, but not all the way to zero.
  - It doesn't do feature selection ("penalty goes to 0 as slope goes to 0"). $\rightarrow f'(0) = 0$ only if $\sum_{i=1}^{n} y_i x_i = 0$

Sparsity and L1-Regularization

- Consider 1D L1-regularized least squares objective:

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

- This is a convex piecwise-quadratic function of 'w' with 'kink' at 0:

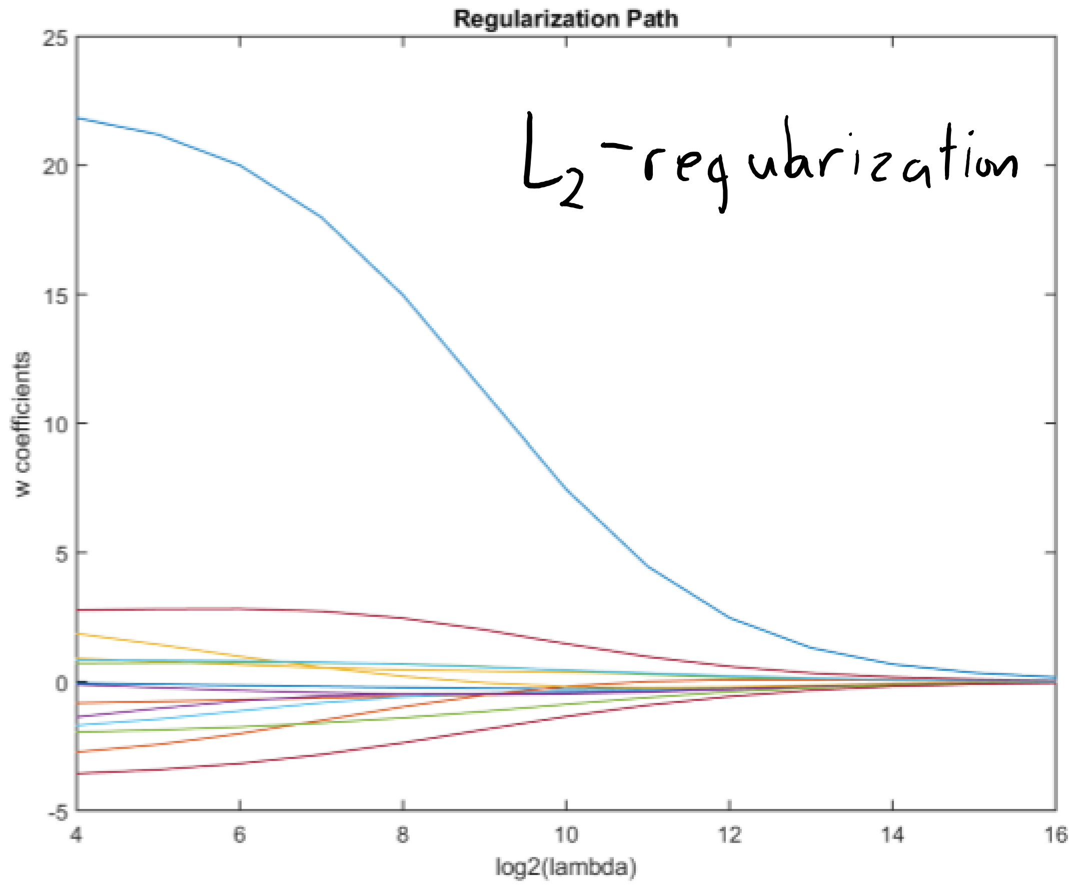


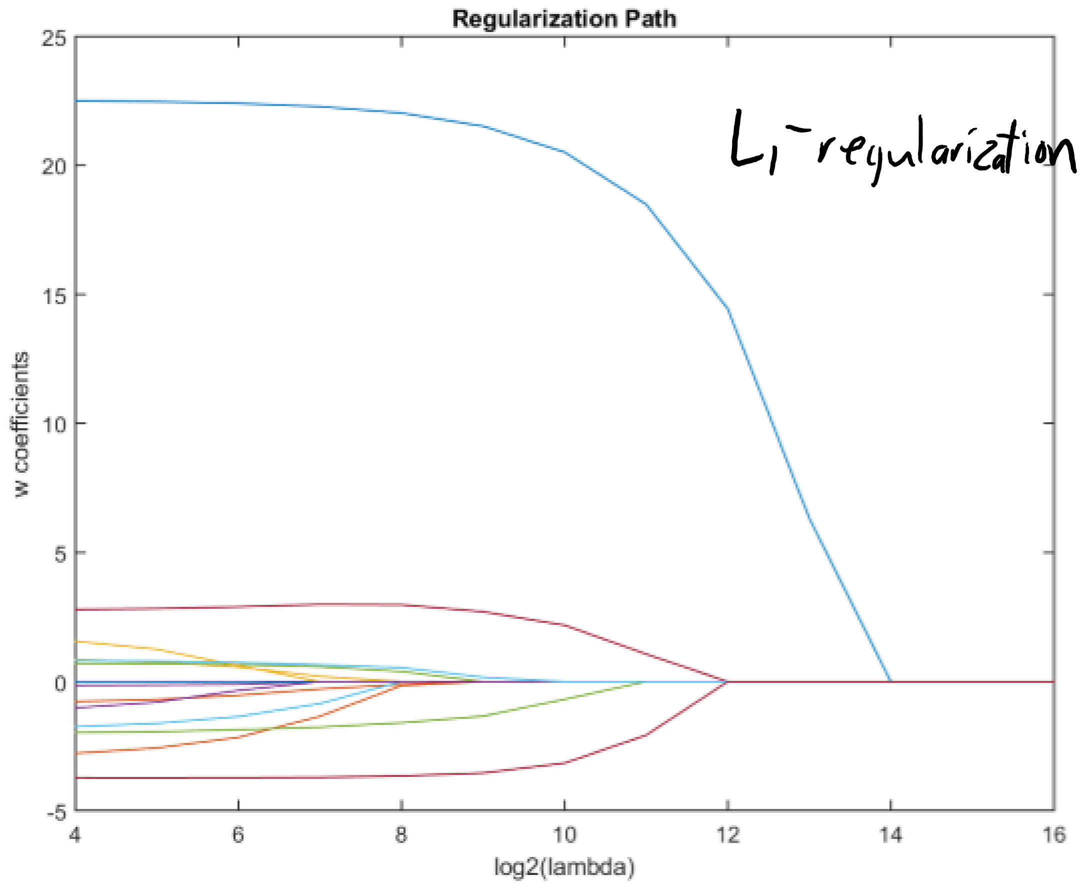
- L1-regularization tends to set variables to exactly 0 (feature selection).
  - Penalty on slope is λ even if you are close to zero.
  - Big λ selects few features, small λ allows many features.

$\rightarrow$ Happens when $\left|\sum_{i=1}^{n} x_i y_i\right| \leq \lambda$ (bonus)

L2-Regularization vs. L1-Regularization

- Regularization path of w_i values as 'λ' varies:





- Bonus slides: details on why only L1-regularization gives sparsity.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All ‘w’ tend to be non-zero.
  - Can learn with *linear* number of irrelevant features.
    - E.g., only O(d) relevant features.

- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.
  - Many ‘w’ tend to be zero.
  - Can learn with **exponential** number of irrelevant features.
    - E.g., only O(log(d)) relevant features.
    [Paper on this result by Andrew Ng]

L1-loss vs. L1-regularization

- Don't confuse the L1 loss with L1-regularization!
  - L1-loss is robust to outlier data points.
    - You can use instead of removing outliers.
  - L1-regularization is robust to irrelevant features.
    - You can use instead of removing features.
- And note that you can be robust to outliers and select features:

$$f(w) = \|Xw - y\|_1 + \lambda\|w\|_1$$

- Why aren't we smoothing and using "Huber regularization"?
  - Huber regularizer is still robust to irrelevant features.
  - But it's the non-smoothness that sets weights to exactly 0.
    - And gradient descent doesn't work well for solving L1-regularization problems.

Summary

- Standardizing features:
  - For some models it makes sense to have features on the same scale.
- Radial basis functions:
  - Non-parametric bases that can model any function.
- L1-regularization:
  - Simultaneous regularization and feature selection.
  - Robust to having lots of irrelevant features.

- Next time: are we really going to use regression for classification?

Why doesn't L2-Regularization set variables to 0?

- Consider an L2-regularized least squares problem with 1 feature:

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

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

Set equal to 0: $\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda w = 0 \xrightarrow{\text{re-arrange}} w\Big(\underbrace{\sum_{i=1}^{n} x_i^2}_{\|x\|^2} + \lambda\Big) = \underbrace{\sum_{i=1}^{n} x_i y_i}_{y^Tx}$

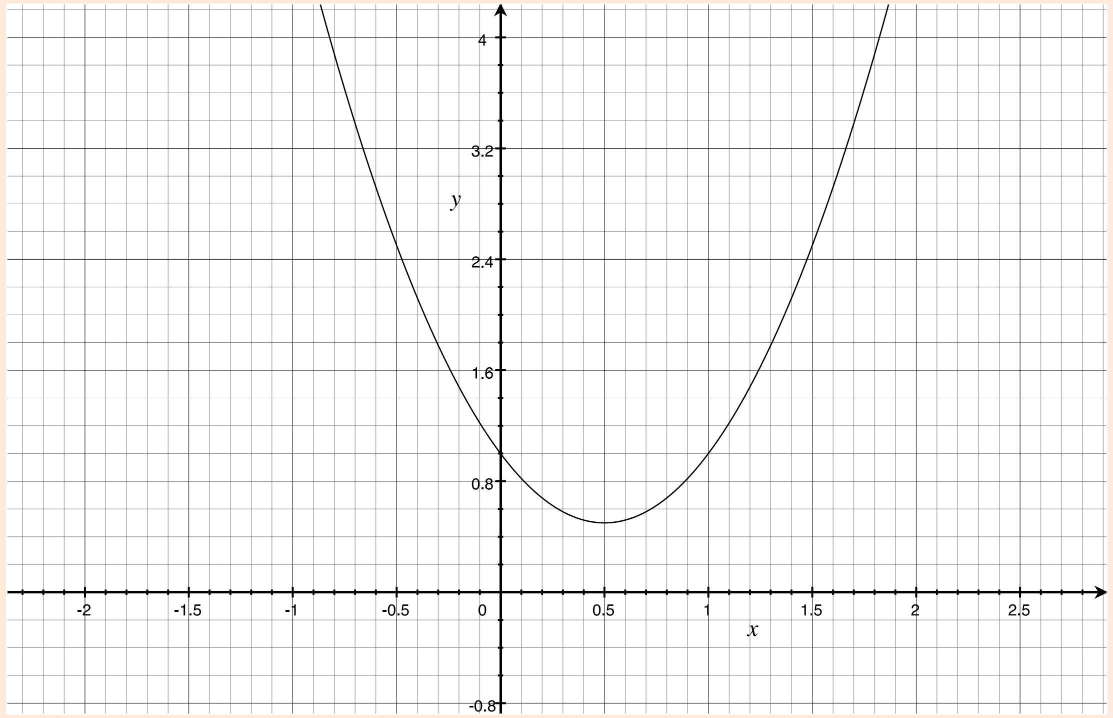
or $w = \frac{y^Tx}{\|x\|^2 + \lambda}$

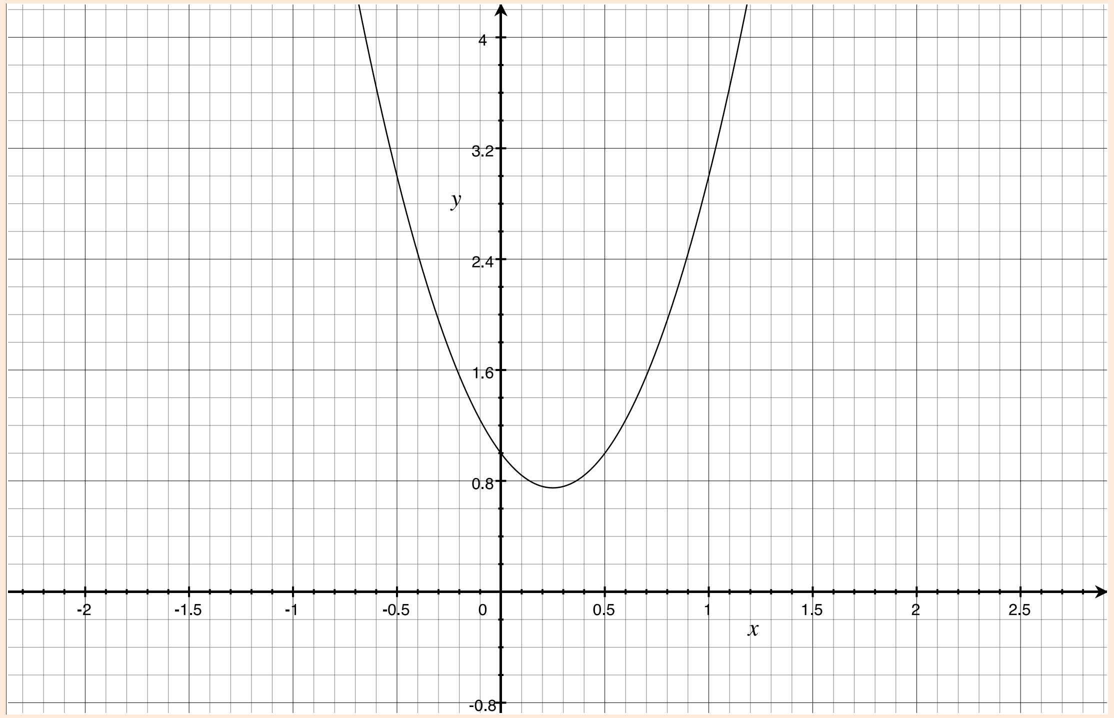
- So as λ gets bigger, 'w' converges to 0.
- However, for all finite λ 'w' will be non-zero unless $y^Tx = 0$.
  - But it's very unlikely that y^Tx will be exactly zero.

Why doesn’t L2-Regularization set variables to 0?

- Small λ

Big λ





- Solution further from zero

Solution closer to zero (but not exactly 0)

Why does L1-Regularization set things to 0?

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then "left" limit and "right" limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda \qquad f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

- So what should gradient descent do if (w=0)?

$$-f^-(0) = \underbrace{-y^Tx + \lambda}_{} \quad -f^+(0) = \underbrace{-y^Tx - \lambda}_{}$$

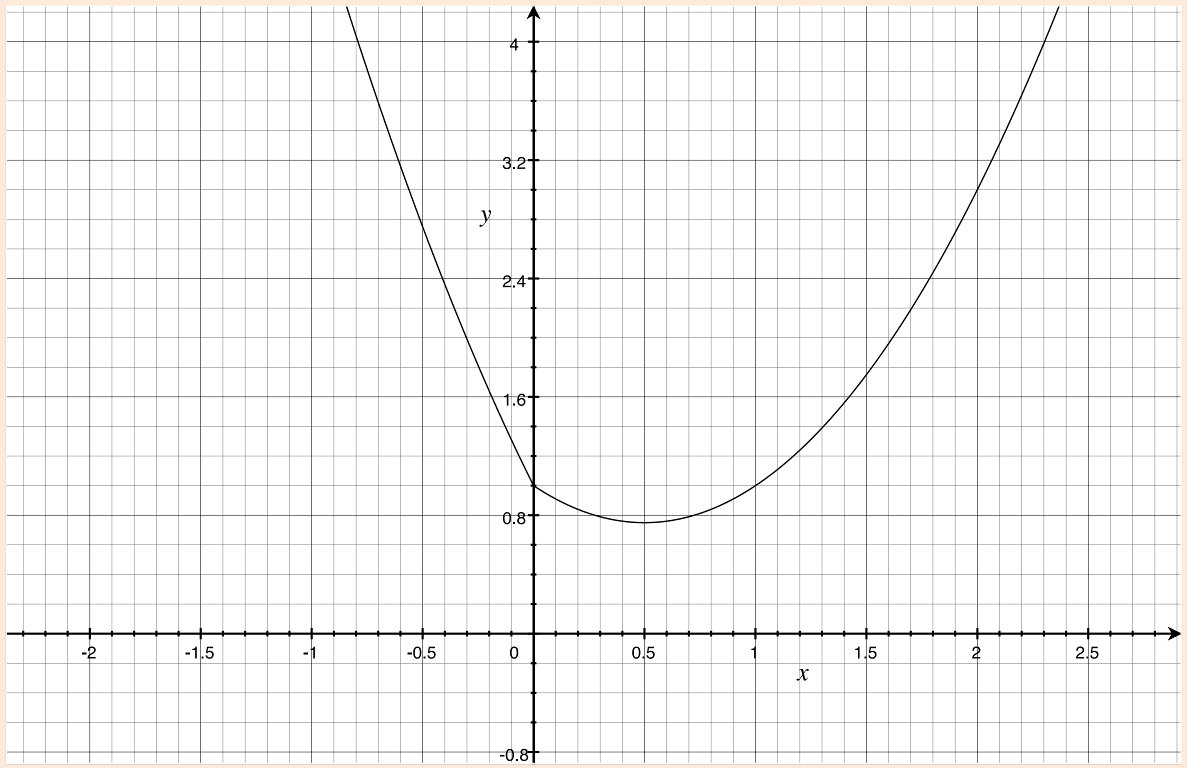
If these are positive $(-y^Tx > \lambda)$, we can improve by increasing 'w'.

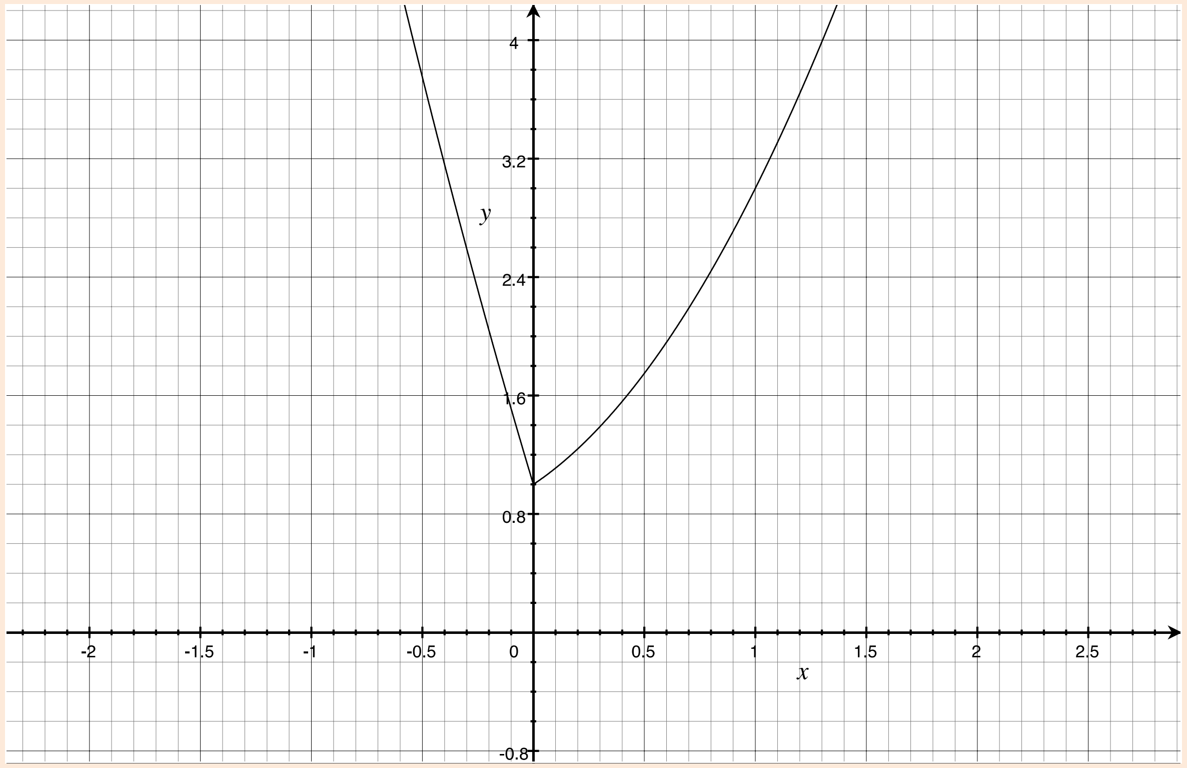
If these are negative $(y^Tx > \lambda)$, we can improve by decreasing 'w'.

But if left and right "gradient descent" directions point in opposite directions $(|y^Tx| \leq \lambda)$, minimum is 0.

Why does L1-Regularization set things to 0?

- Small λ Big λ





- Solution nonzero Solution exactly zero

(minimum of left parabola is past origin, but right parabola is not) (minimum of both parabola are past the origin)

L2-regularization vs. L1-regularization

- So with 1 feature:
  - L2-regularization only sets ‘w’ to 0 if $y^T x = 0$.
    - There is a only a single possible $y^T x$ value where the variable gets set to zero.
    - And λ has nothing to do with the sparsity.
  - L1-regularization sets ‘w’ to 0 if $|y^T x| \leq \lambda$.
    - There is a range of possible $y^T x$ values where the variable gets set to zero.
    - And increasing λ increases the sparsity since the range of $y^T x$ grows.
- Not that it’s really important that the function is non-differentiable:
  - If we used “Huber regularization”, it would select all variables.

L1-Loss vs. Huber Loss

- The same reasoning tells us the difference between the L1 *loss* and the Huber loss. They are very similar in that they both grow linearly far away from 0. So both are both robust but…
  - With the L1 loss the model often passes exactly through some points.
  - With Huber the model doesn’t necessarily pass through any points.

- Why? With L1-regularization we were causing the elements of ’w’ to be exactly 0. Analogously, with the L1-loss we cause the elements of ‘r’ (the residual) to be exactly zero. But zero residual for an example means you pass through that example exactly.

Non-Uniqueness of L1-Regularized Solution

- How can L1-regularized least squares solution not be unique?
  - Isn’t it convex?
- Convexity implies that minimum value of f(w) is unique (if exists), but there may be multiple ‘w’ values that achieve the minimum.

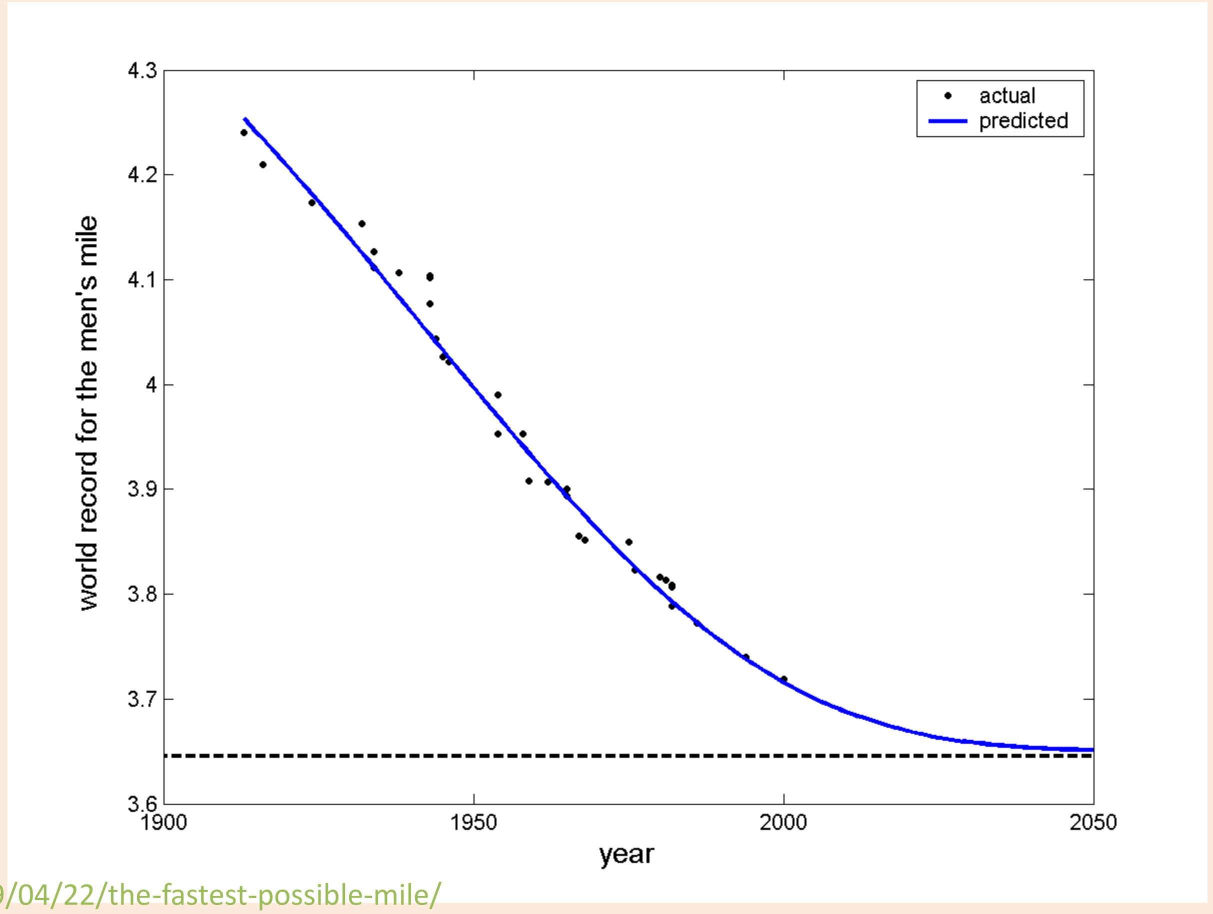
- Consider L1-regularized least squares with d=2, where feature 2 is a copy of a feature 1. For a solution (w_1,w_2) we have:

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} = w_1 x_{i1} + w_2 x_{i1} = (w_1 + w_2) x_{i1}$$

- So we can get the same squared error with different w_1 and w_2 values that have the same sum. Further, if neither w_1 or w_2 changes sign, then $|w_1| + |w_2|$ will be the same so the new w_1 and w_2 will be a solution.

Predicting the Future

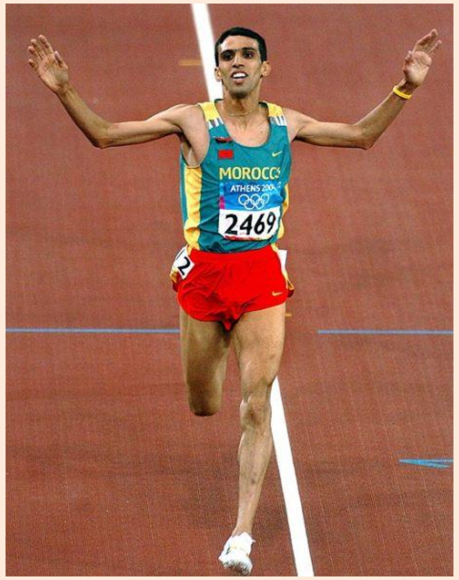
- In principle, we can use any features x_i that we think are relevant.
- This makes it tempting to use time as a feature, and predict future.

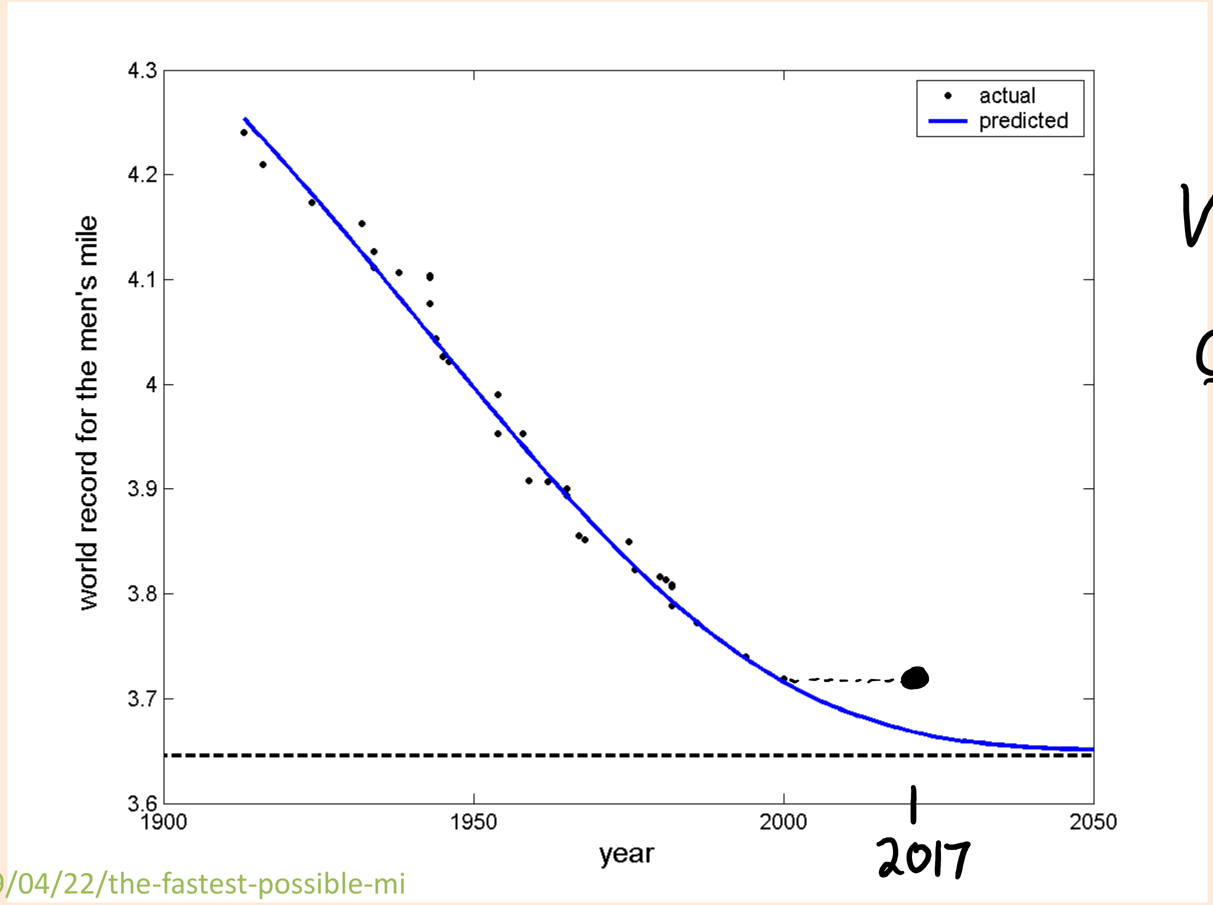


https://gravityandlevity.wordpress.com/2009/04/22/the-fastest-possible-mile/

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.

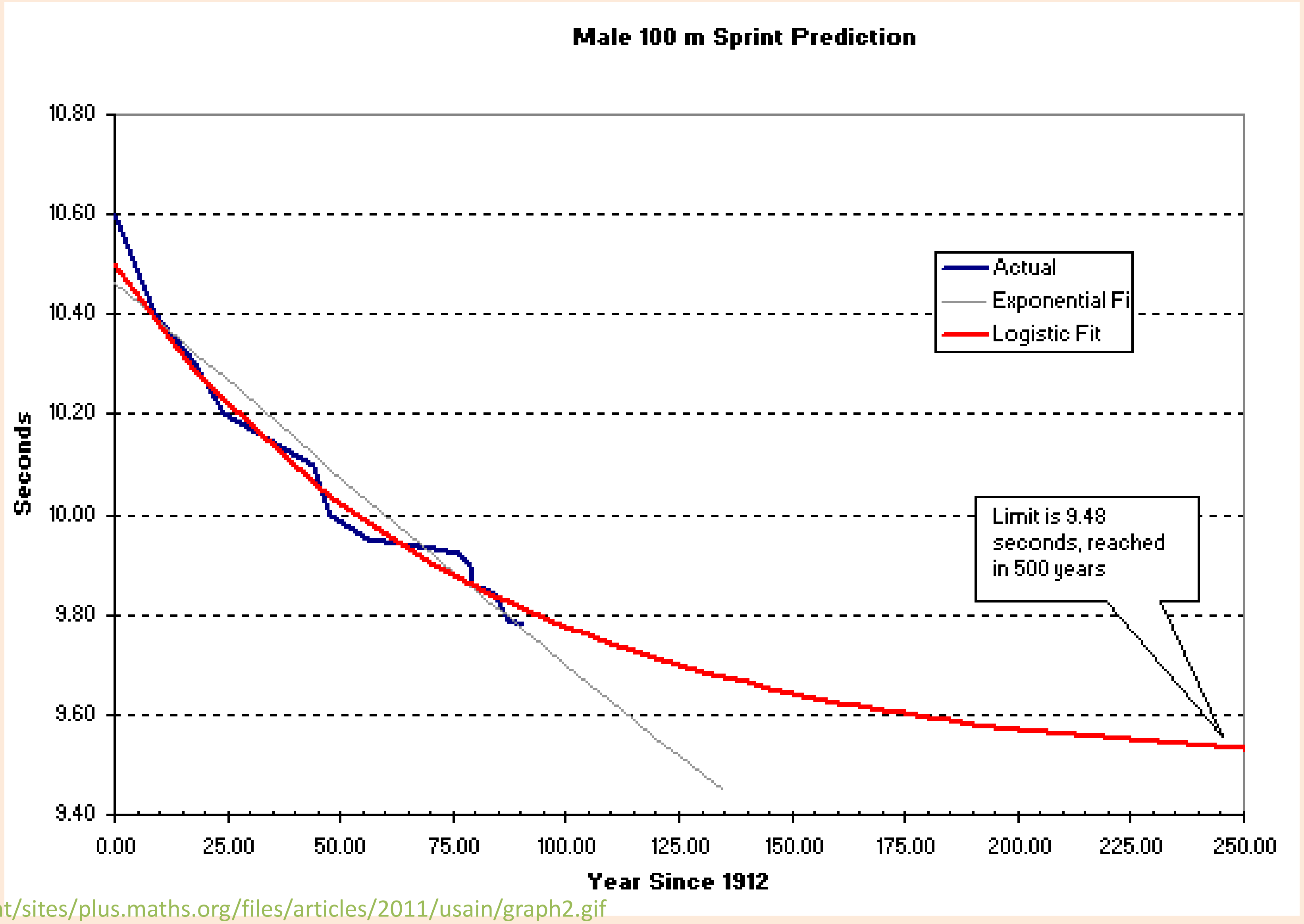




We need to be cautious about doing this.

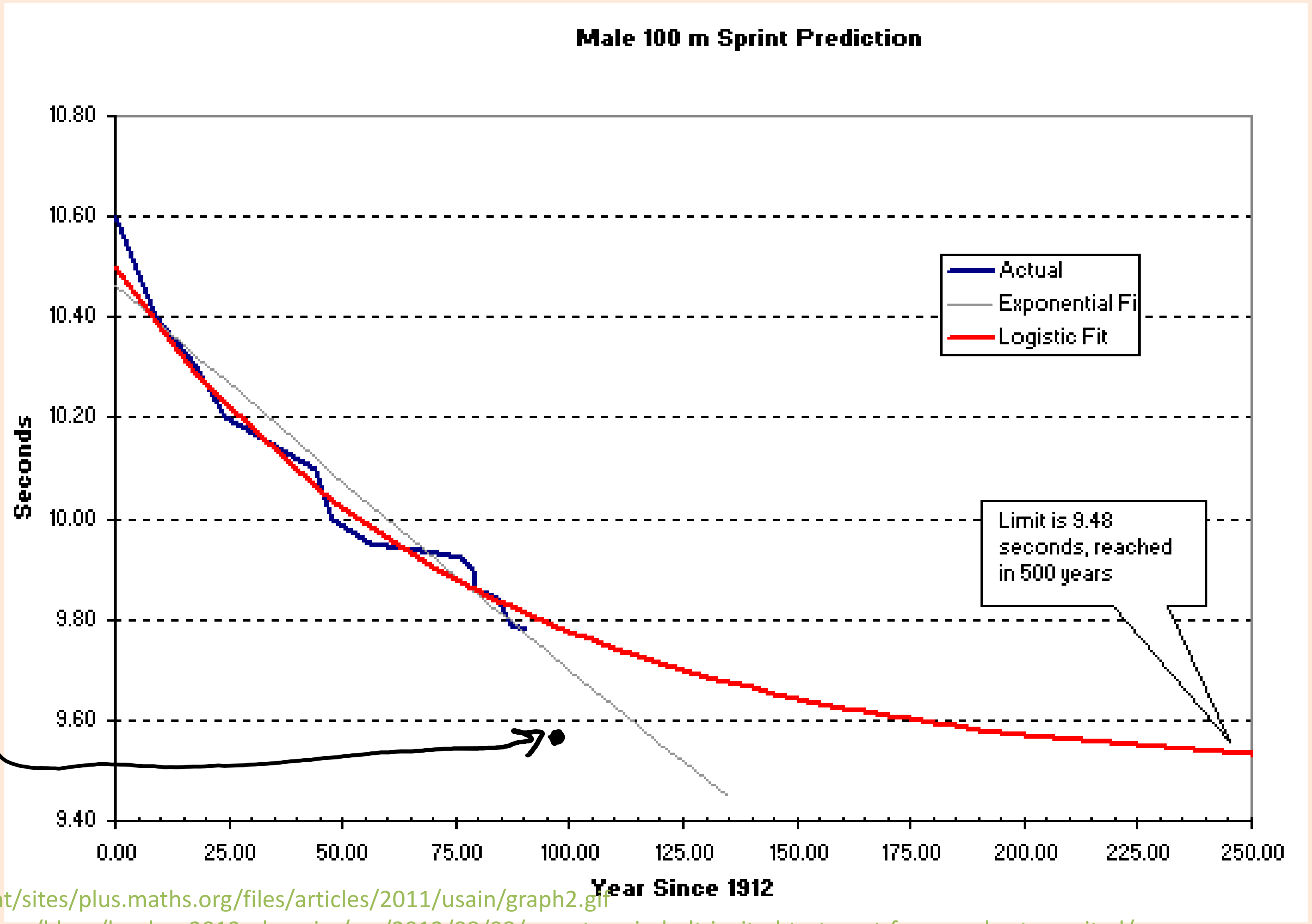
https://gravityandlevity.wordpress.com/2009/04/22/the-fastest-possible-mi
https://overthehillsports.wordpress.com/tag/hicham-el-guerrouj/le/

Predicting 100m times 400 years in the future?



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

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/

Interpolation vs Extrapolation

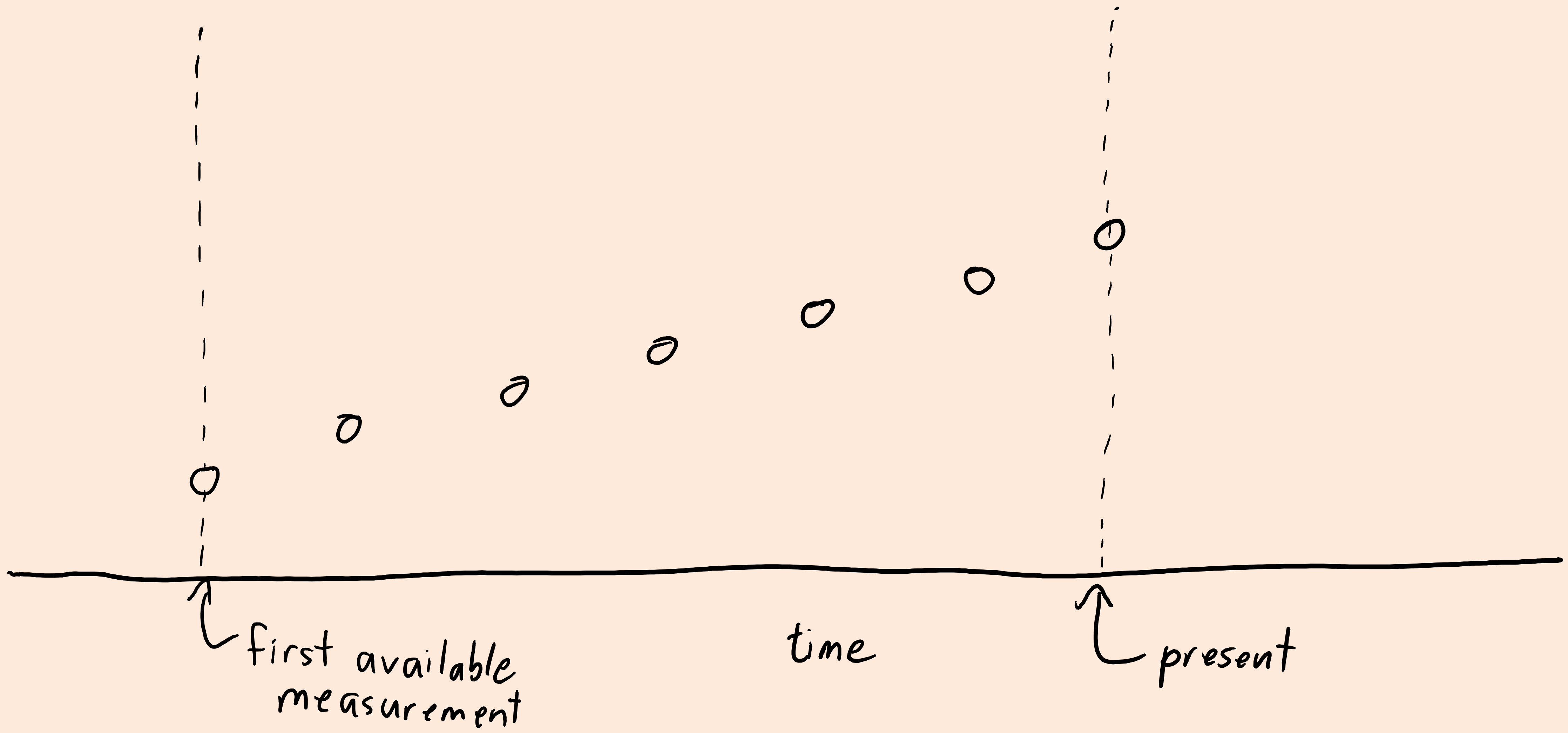
- Interpolation is task of predicting "between the data points".
  - Regression models are good at this if you have enough data and function is smooth.
- Extrapolation is task of prediction outside the range of the data points.
  - Without assumptions, regression models can be embarrassingly-bad at this.

- If you run the 100m regression models backwards in time:
  - They predict that humans used to be really really slow!

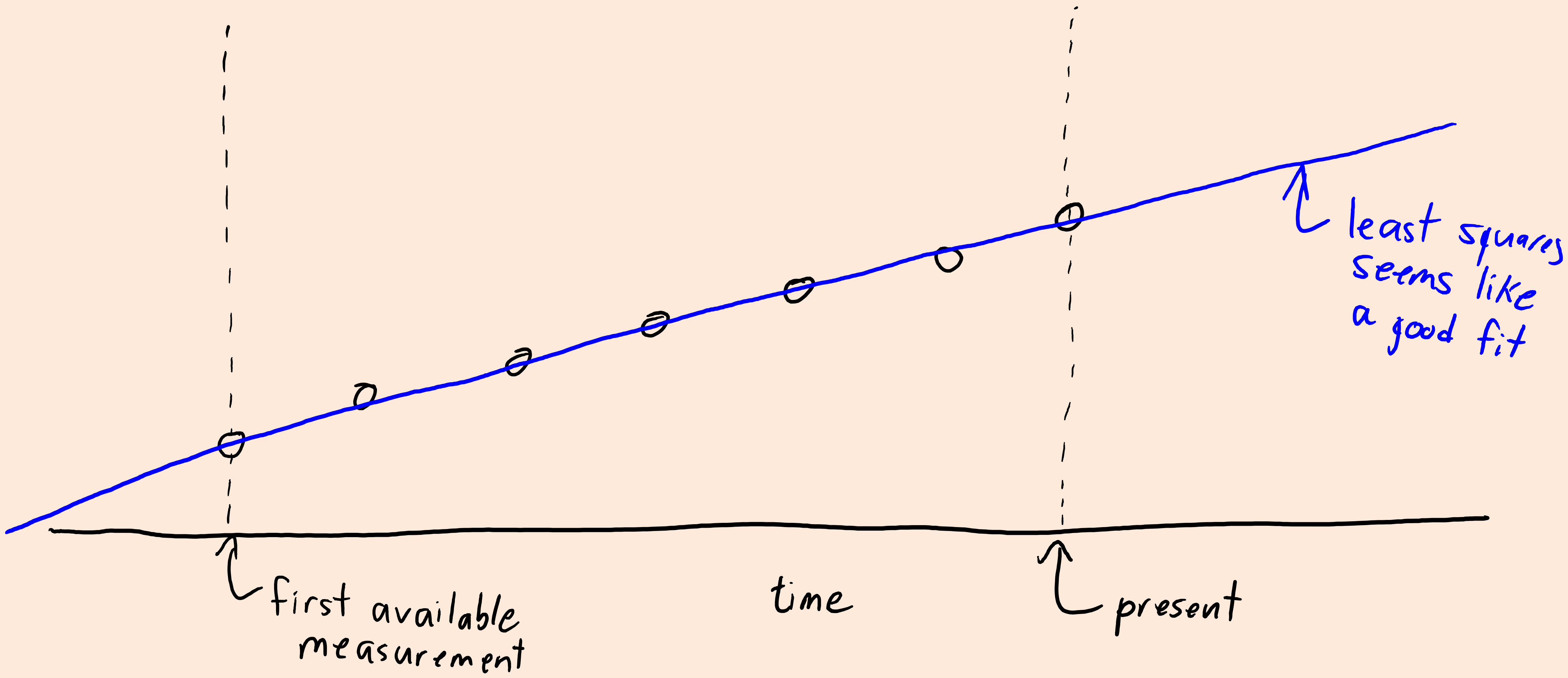
- If you run the 100m regression models forwards in time:
  - They might eventually predict arbitrarily-small 100m times.
  - The linear model actually predicts negative times in the future.
    - These time traveling races in 2060 should be pretty exciting!

- Some discussion here:
  - http://callingbullshit.org/case_studies/case_study_gender_gap_running.html

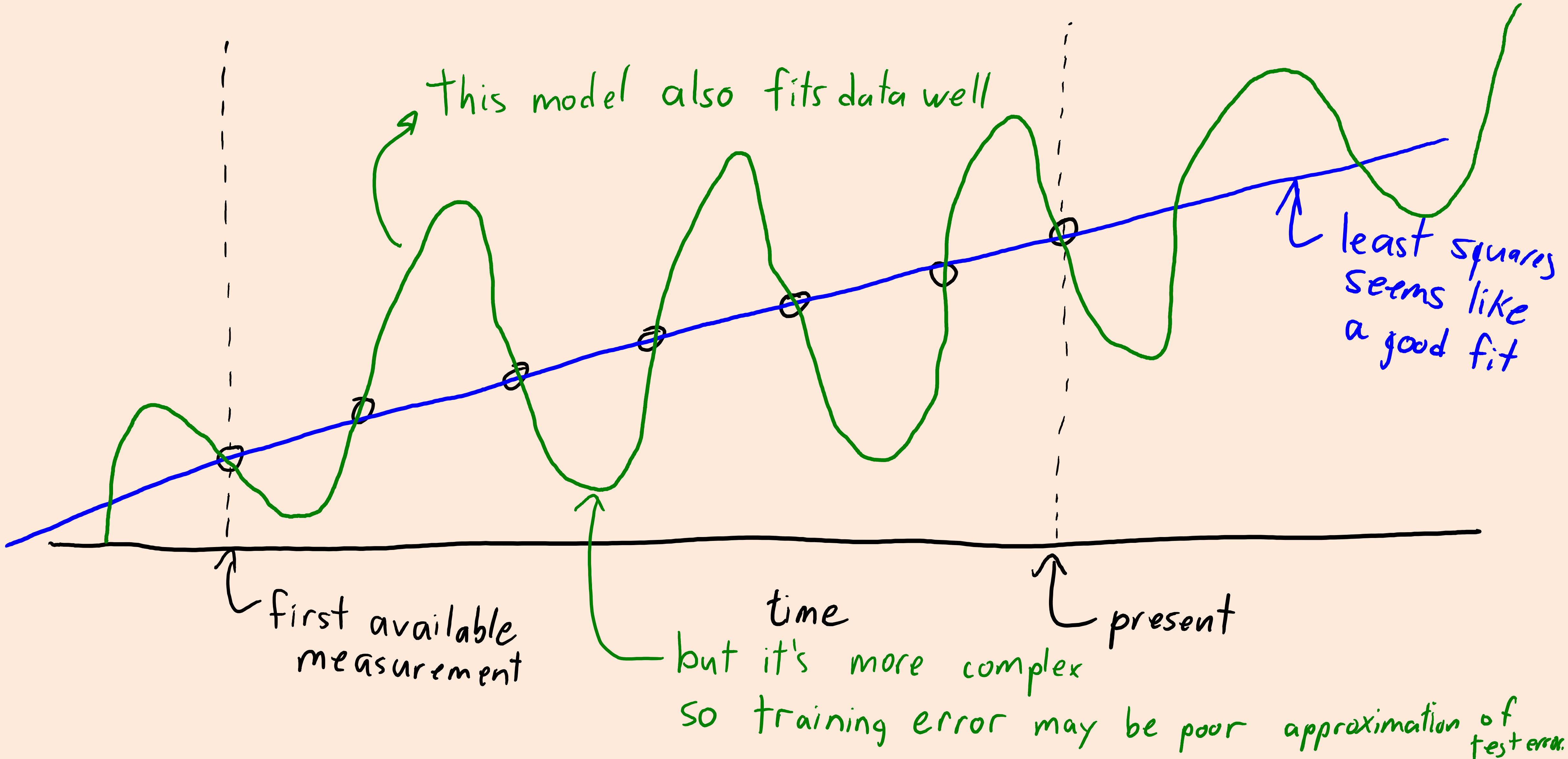
No Free Lunch, Consistency, and the Future



No Free Lunch, Consistency, and the Future

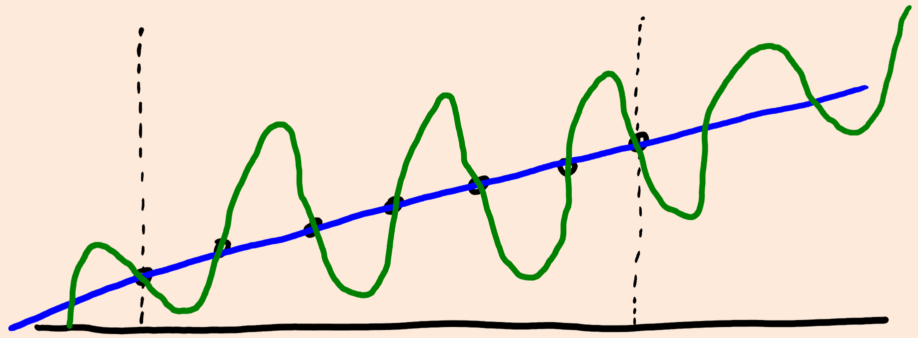


No Free Lunch, Consistency, and the Future


this model also fits data well
least squares seems like a good fit
first available measurement
time
present
but it's more complex
so training error may be poor approximation of test error.

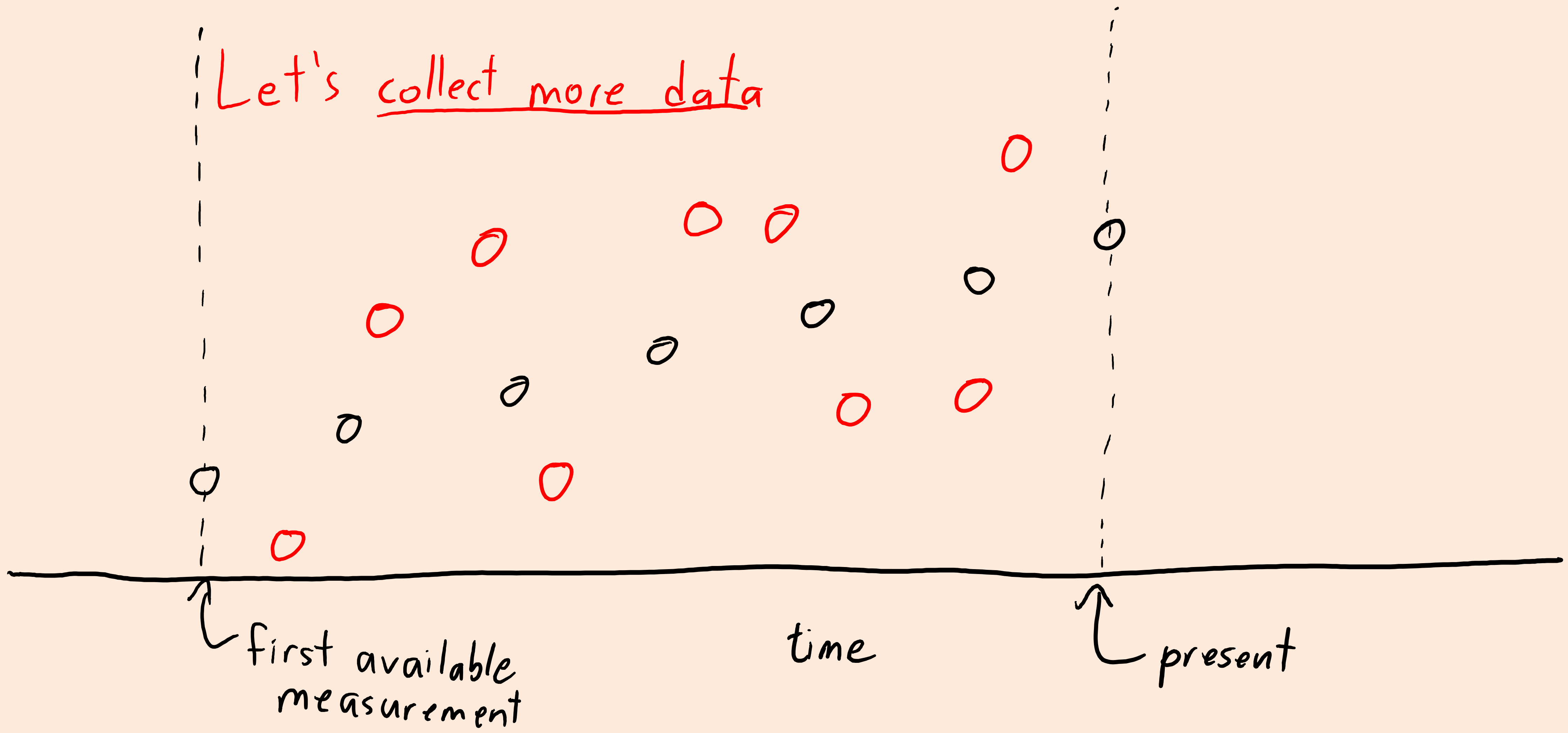
Ockham's Razor vs. No Free Lunch

- 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.

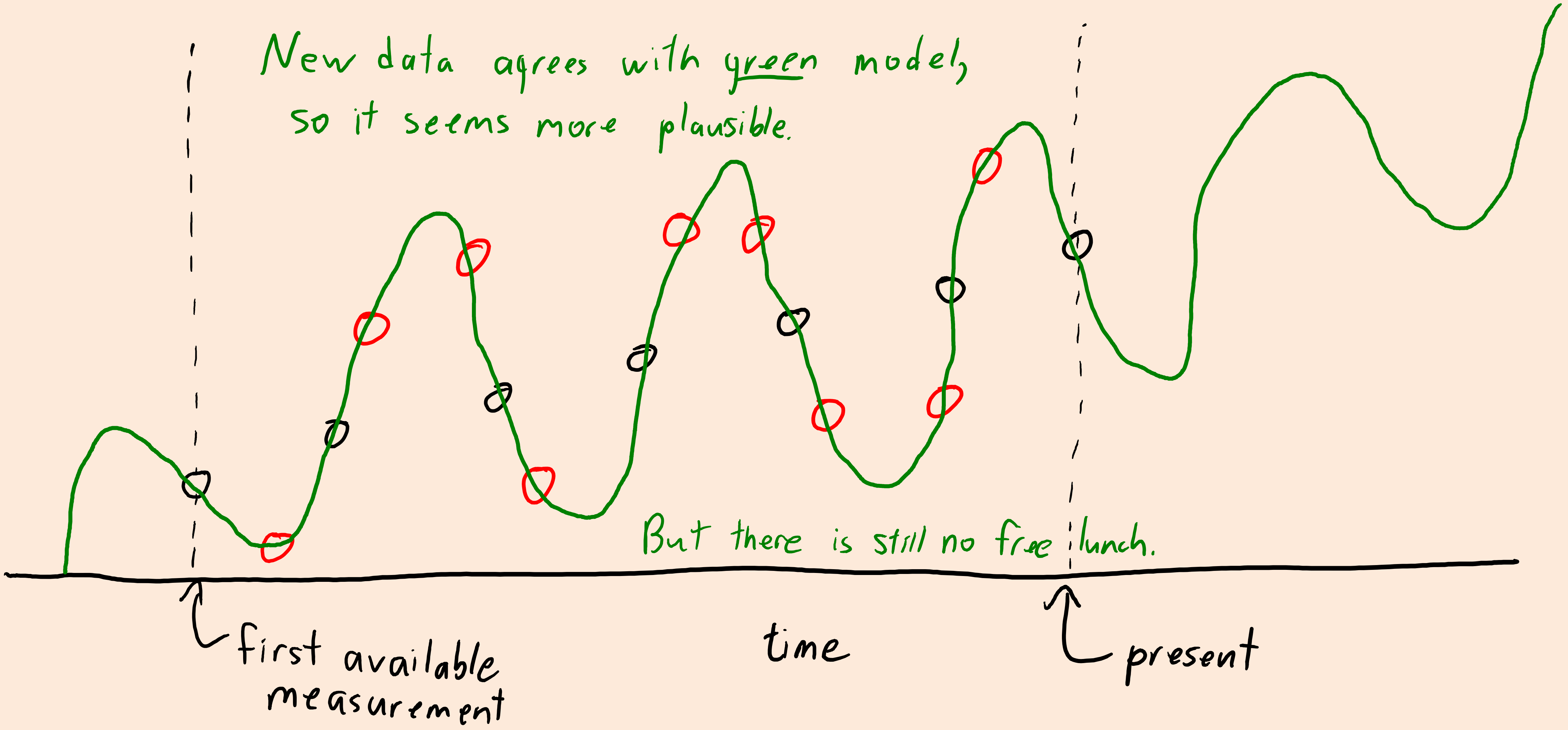


- Fundamental trade-off:
  - If same training error, pick model less likely to overfit.
  - Formal version of Occam's problem-solving principle.
  - Also suggests we should select linear model.
- No free lunch theorem:
  - There *exists possible datasets* where you should select the green model.

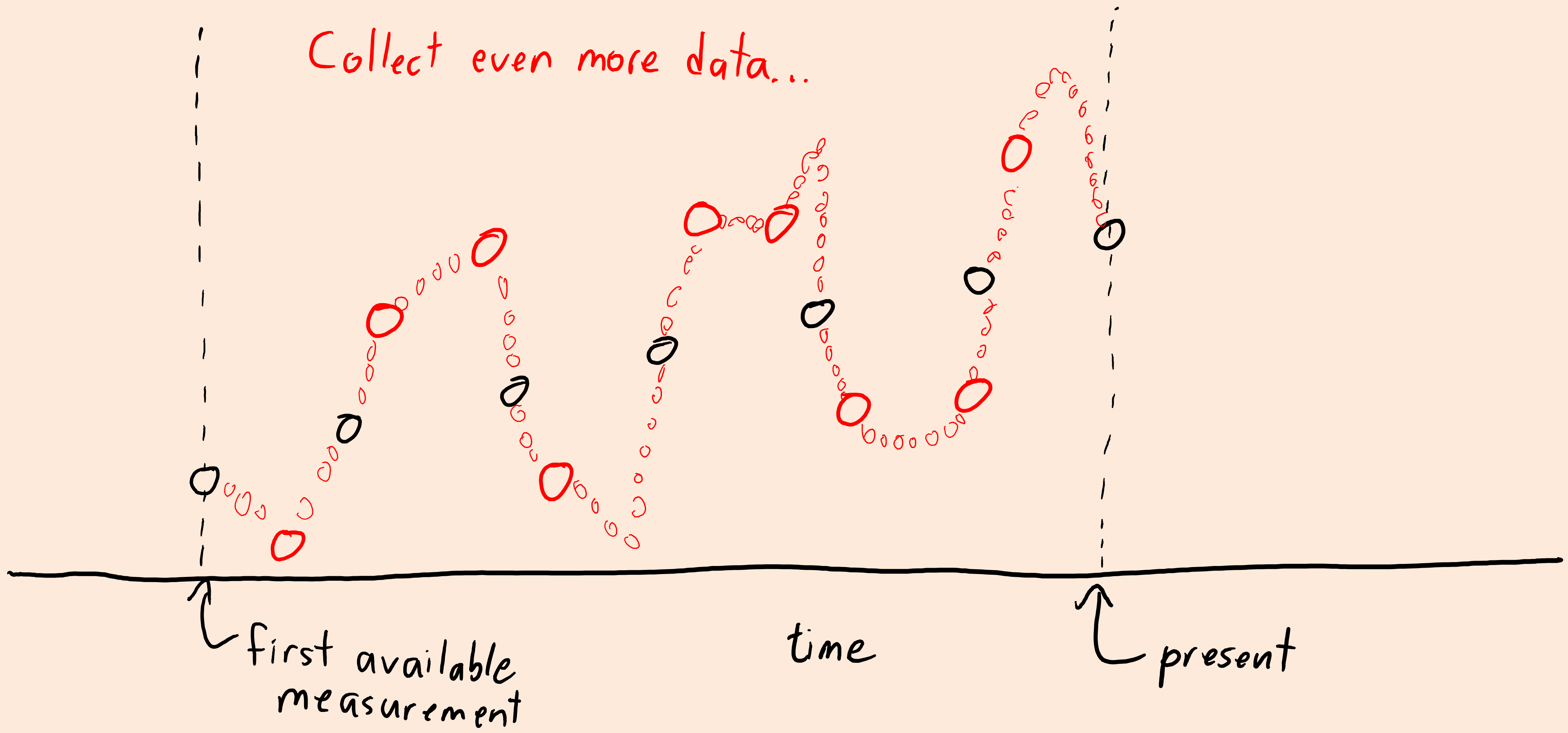
No Free Lunch, Consistency, and the Future



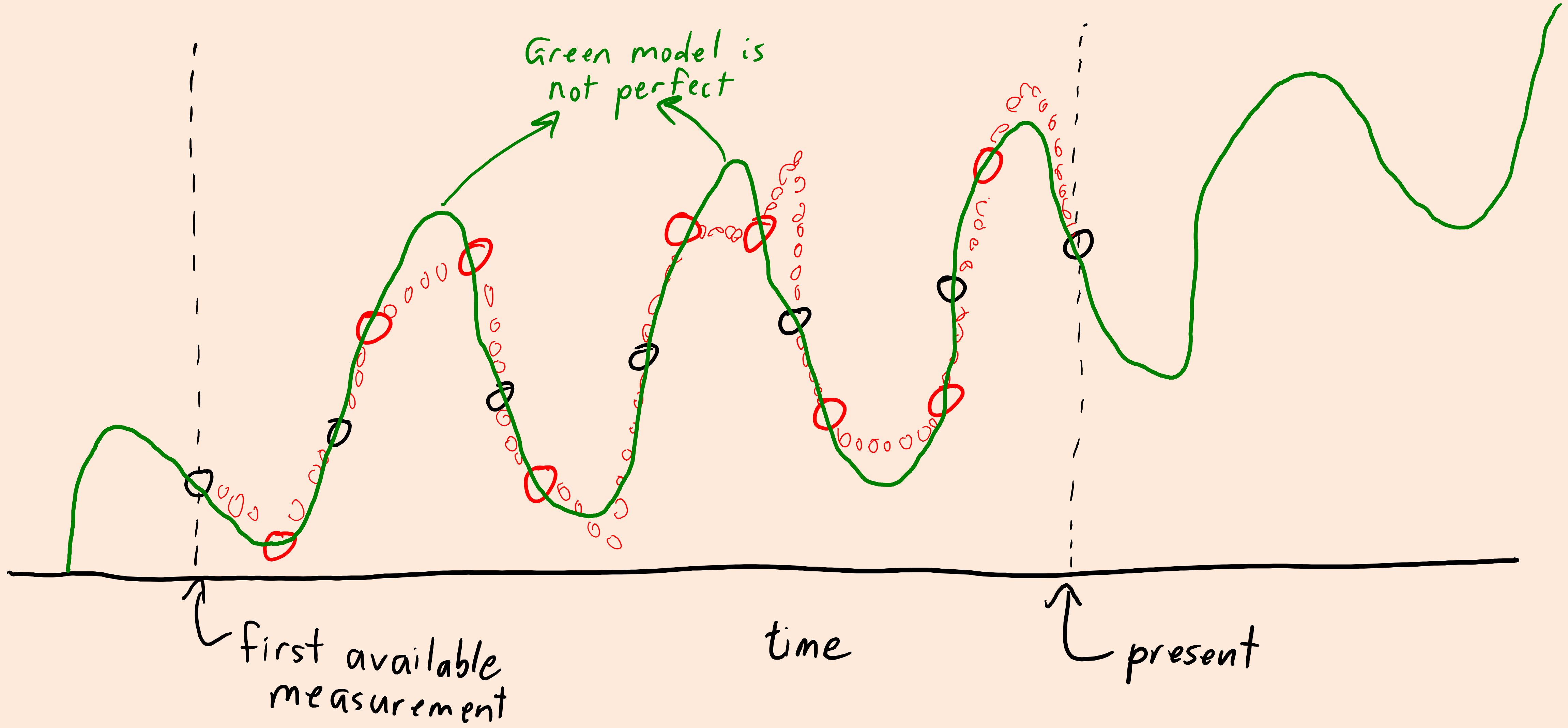
No Free Lunch, Consistency, and the Future


New data agrees with green model, so it seems more plausible.
But there is still no free lunch.
first available measurement
time
present

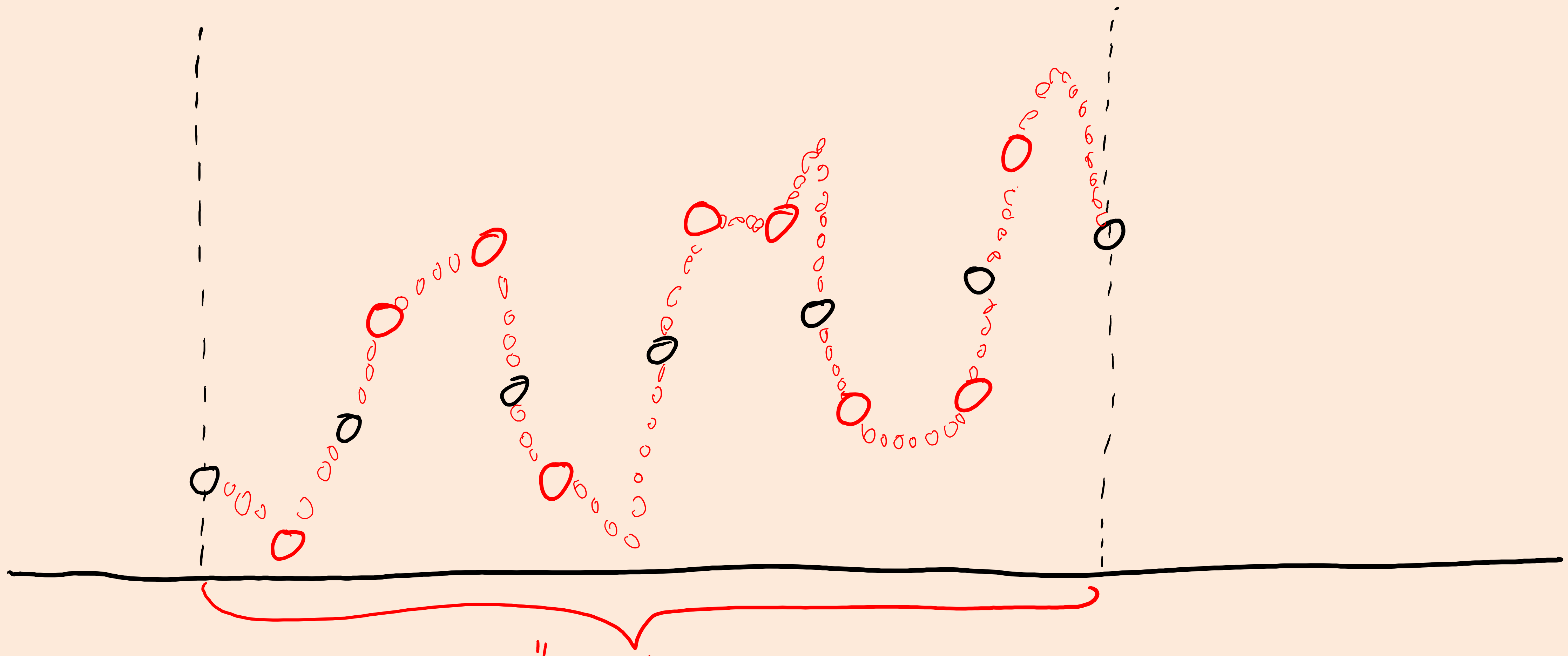
No Free Lunch, Consistency, and the Future



No Free Lunch, Consistency, and the Future



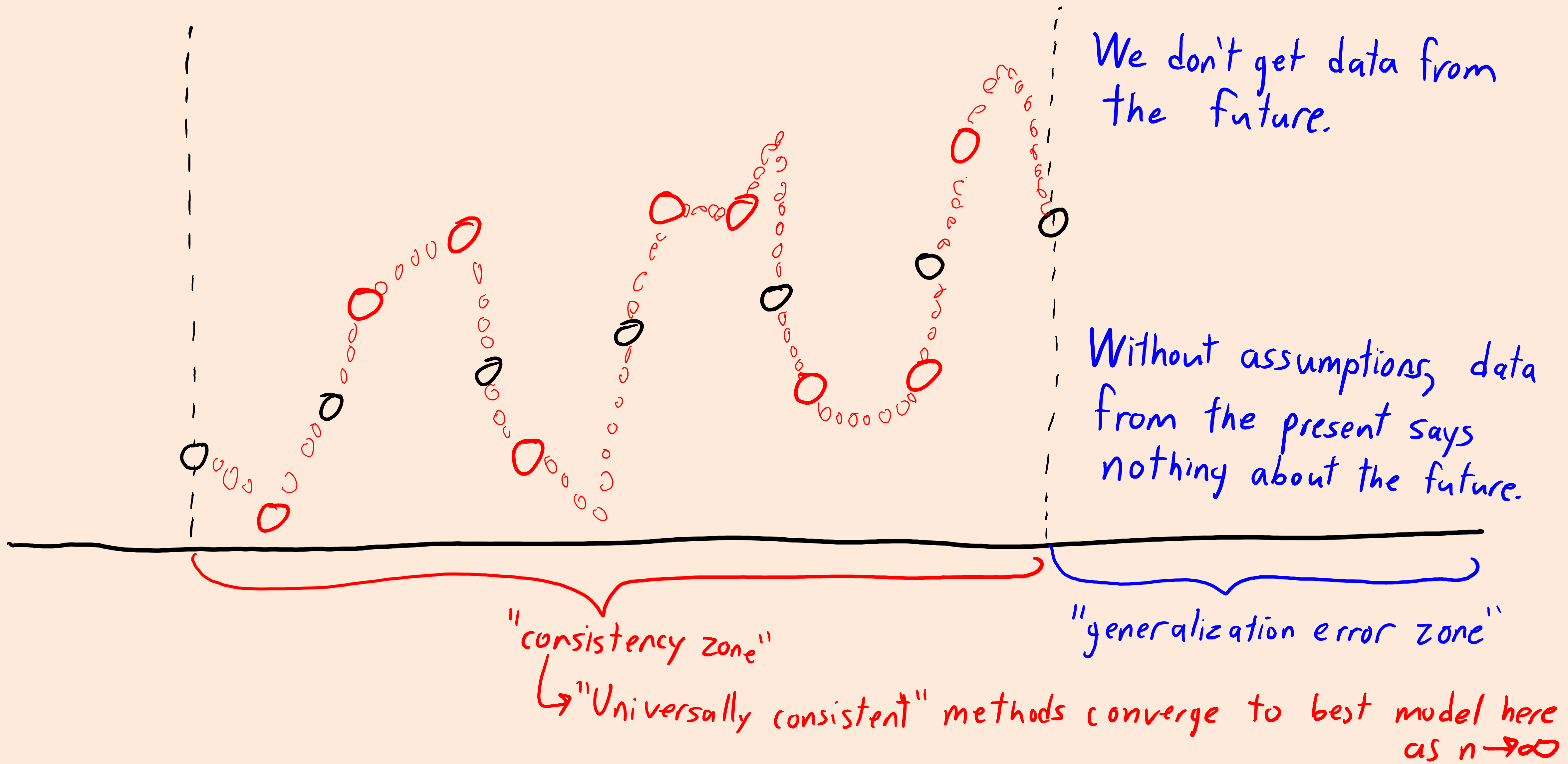
No Free Lunch, Consistency, and the Future



"consistency zone"

↳ "Universally consistent" methods converge to best model here as $n \to \infty$

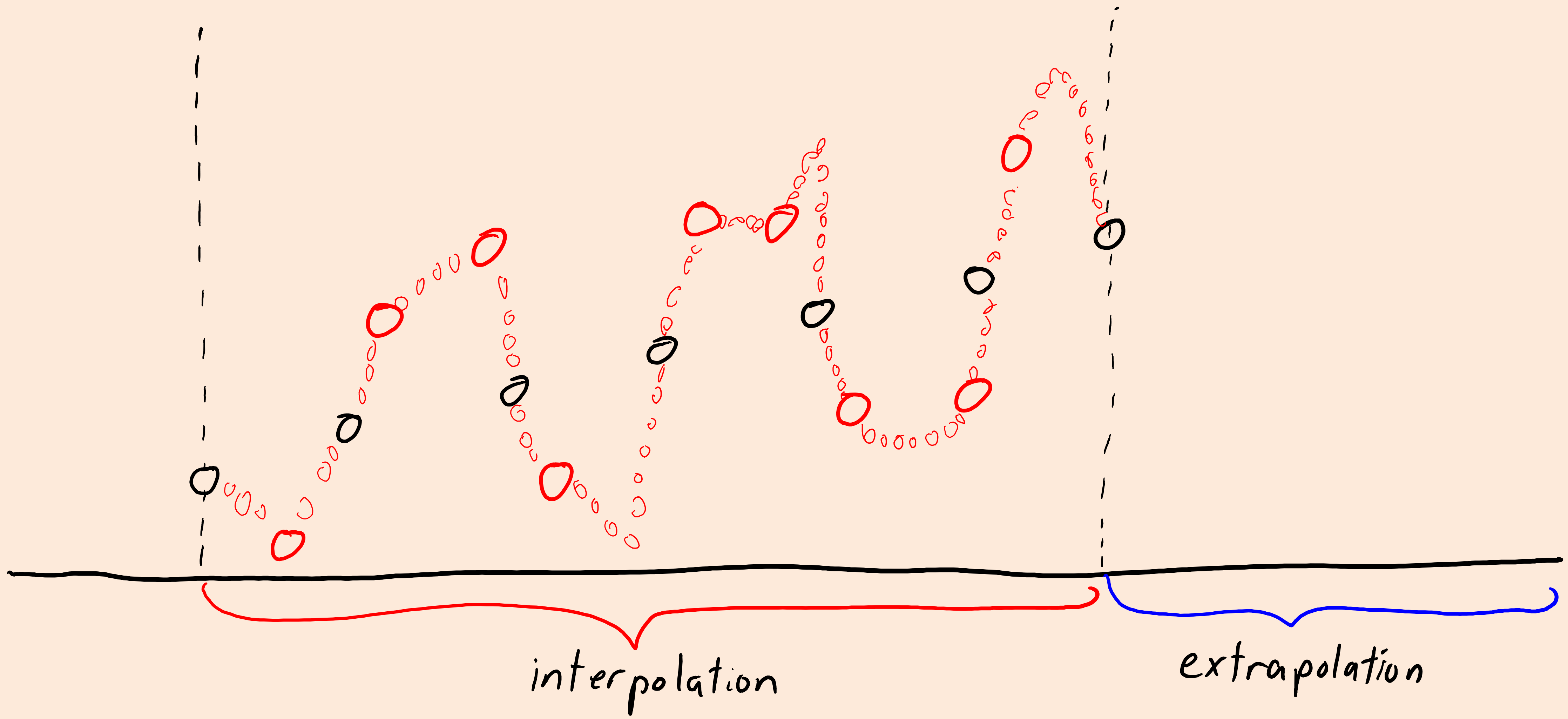
No Free Lunch, Consistency, and the Future



We don't get data from the future.
Without assumptions, data from the present says nothing about the future.
"consistency zone"
"generalization error zone"

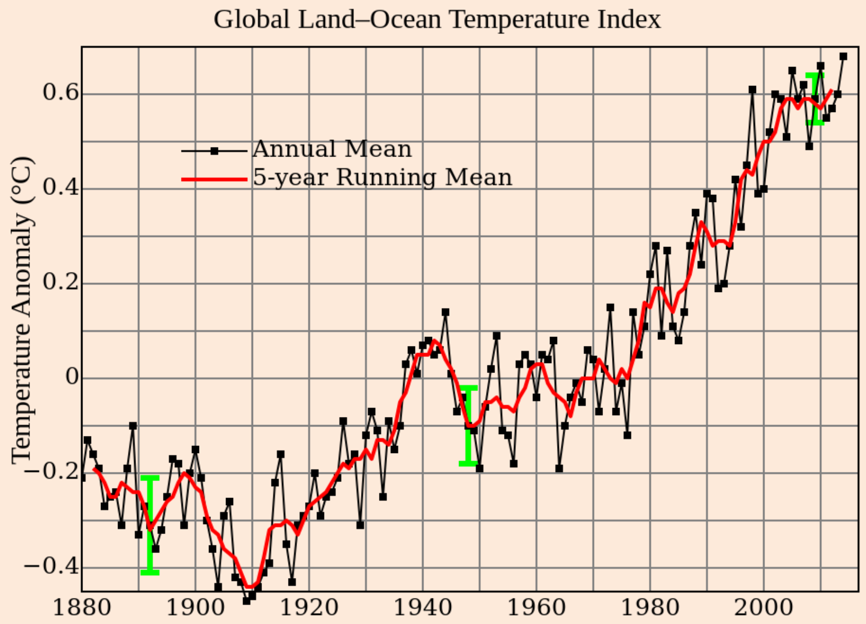
"Universally consistent" methods converge to best model here as $n \to \infty$

No Free Lunch, Consistency, and the Future



Discussion: Climate Models

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

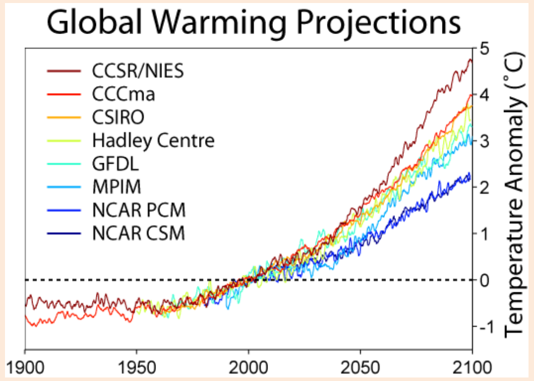


- Will Earth continue to warm over next 100 years? (generalization error)
  - We should be more skeptical about models that predict future events.

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

Discussion: Climate Models

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

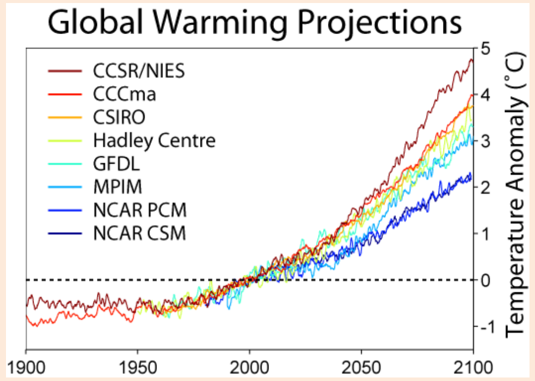


  – We should be skeptical of individual models, but agreeing predictions made by models with different data/assumptions are more likely be true.
- All the near-future predictions agree, so they are likely to be accurate.
- Variance is higher further into future, so predictions are less reliable.

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

Discussion: Climate Models

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

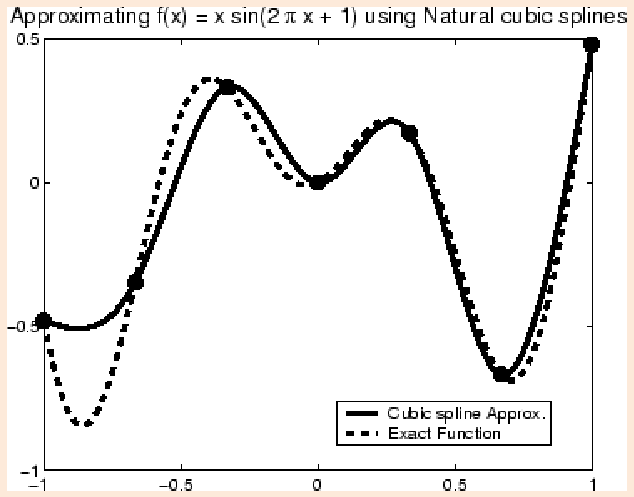


- Process is probably continuous:
  - If so, near-future predictions would be "close enough" to consistency zone.
  - As we go further in the future, we enter "no free lunch" zone where we start to need to reliable more and more on our assumptions.

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

Splines in 1D

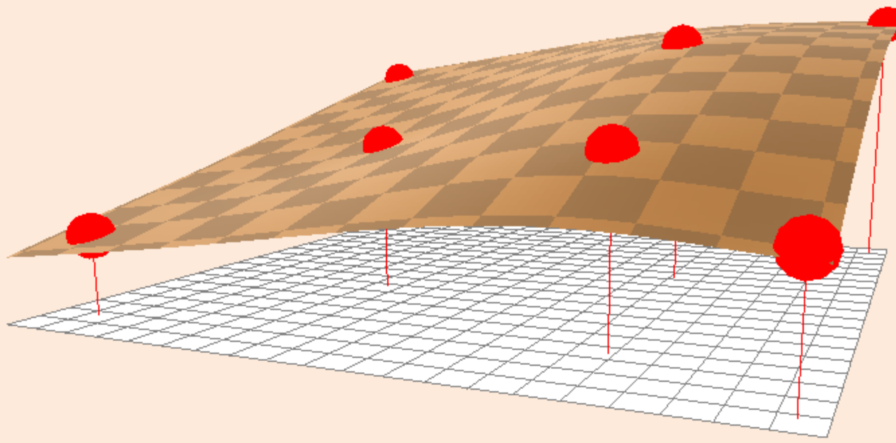
- For 1D interpolation, alternative to polynomials/RBFs are splines:
  - Use a polynomial in the region between each data point.
  - Constrain some derivatives of the polynomials to yield a unique solution.
- Most common example is cubic spline:
  - Use a degree-3 polynomial between each pair of points.
  - 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 $g(\varepsilon) = \varepsilon^3$.
    - Different than Gaussian RBF because it *increases with distance*.



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

Splines in Higher Dimensions

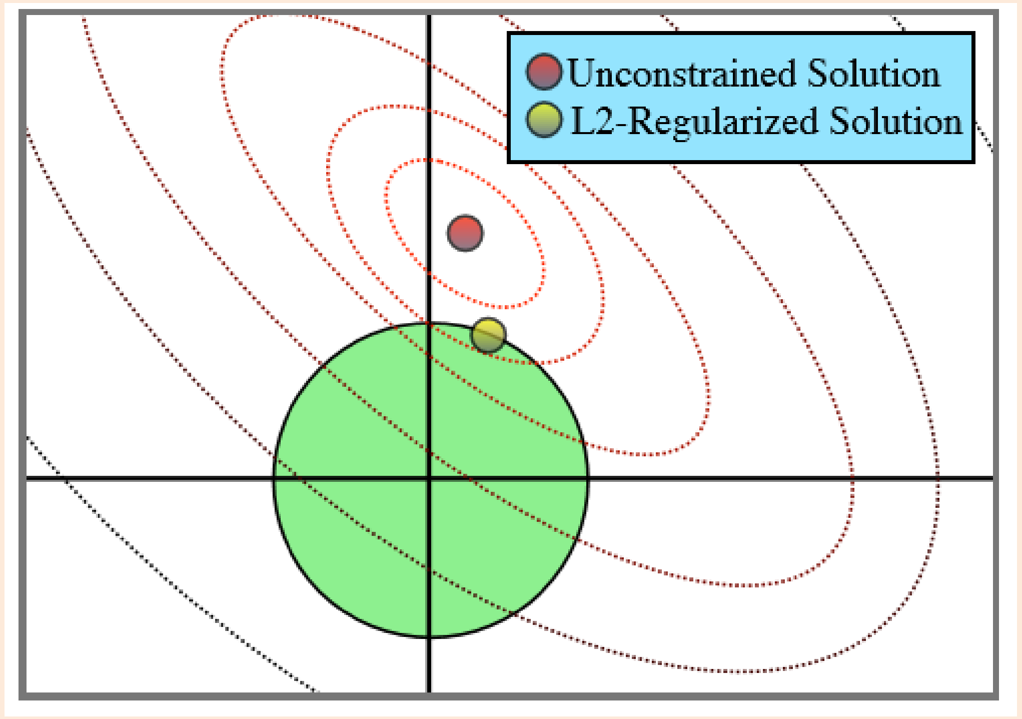
- Splines generalize to higher dimensions if data lies on a grid.
  - For more general (“scattered”) data, there isn’t a natural generalization.
- Common 2D “scattered” data interpolation is thin-plate splines:
  - Based on curve made when bending sheets of metal.
  - Corresponds to RBFs with $g(\varepsilon)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



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

L2-Regularization vs. L1-Regularization

- L2-regularization conceptually restricts 'w' to a ball.

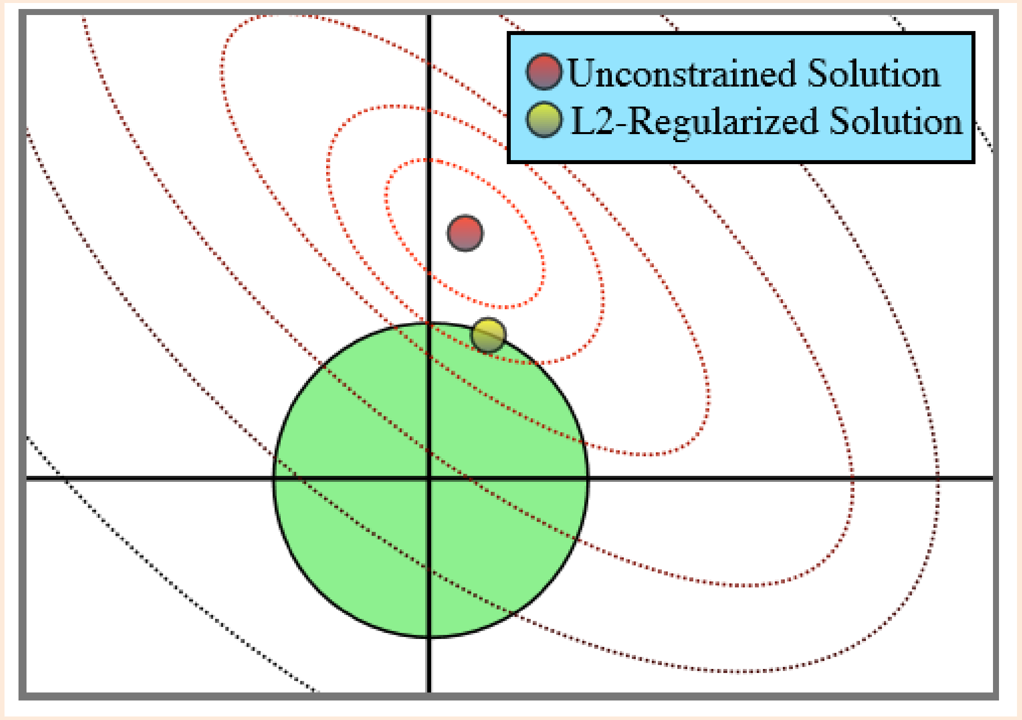


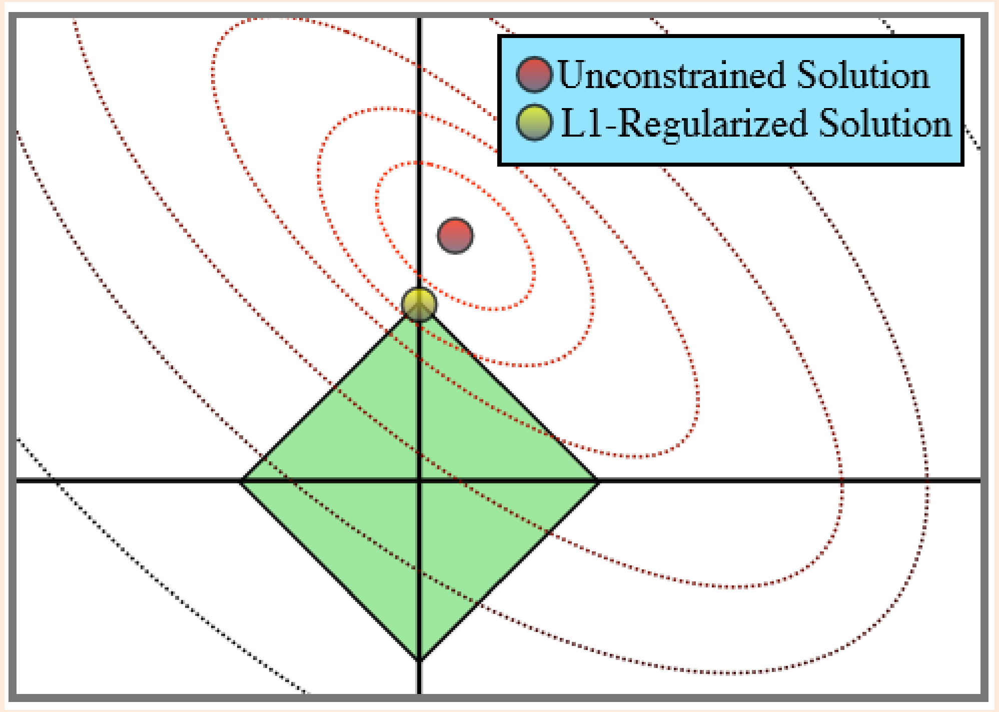
Minimizing $\frac{1}{2}||Xw - y||^2 + \frac{\lambda}{2}||w||^2$

is equivalent to minimizing $\frac{1}{2}||Xw - y||^2$ subject to the constraint that $||w|| \leq \tau$ for some value 'τ'

L2-Regularization vs. L1-Regularization

- L2-regularization conceptually restricts ‘w’ to a ball.





- L1-regularization restricts to the L1 “ball”:
  - Solutions tend to be at corners where w_j are zero.

Related Infinite Series video