CPSC 340:
Machine Learning and Data Mining

Nonlinear Regression

"One of the most surprising and important stories of our time."
—Ashlee Vance, author of *Elon Musk*

Genius Makers

The Mavericks Who Brought AI to Google, Facebook, and the World

CADE METZ

Last Time: Linear Regression

- We discussed linear models:

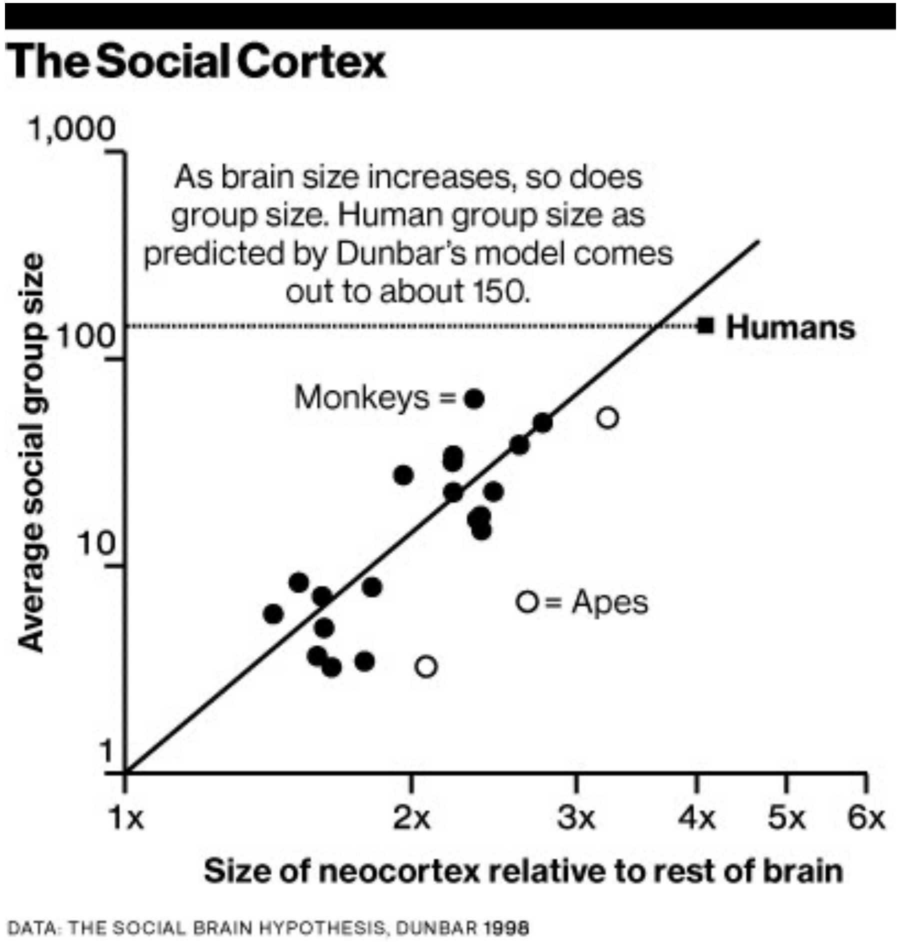
$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= \sum_{j=1}^{d} w_j x_{ij}$$

- "Multiply feature x_{ij} by weight w_j, add them to get y_i".
- We discussed squared error function:

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

Predicted value ($w^T x_i$)   True value (y_i)

  - Minimize 'f' by equating gradient of 'f' with zero.
- Interactive demo:
  - http://setosa.io/ev/ordinary-least-squares-regression



To predict on test case $\tilde{x}_i$
use $\hat{y}_i = w^T \tilde{x}_i$

http://www.bloomberg.com/news/articles/2013-01-10/the-dunbar-number-from-the-guru-of-social-networks

Matrix/Norm Notation (MEMORIZE/STUDY THIS)

- Parts 1&2: we used lists (not vectors): e.g. x_i was a 1D list of length d
- From now on: we use vectors, and typically assume that vectors are column-vectors
  - We use 'w' as a "d times 1" vector containing weight 'w_j' in position 'j'.
  - We use 'y' as an "n times 1" vector containing target 'y_i' in position 'i'.
  - We use 'x_i' as a "d times 1" vector containing features 'j' of example 'i'.
    - We're now going to be careful to make sure these are column vectors.
  - So 'X' is a matrix with x_i^T in row 'i'. (note the latter x is lowercase: it is not the ith row of X)
- Recommended: Course Notation Guide (on website)

lowercase x's

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix} \quad X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix} = \begin{bmatrix} \text{---} & x_1^T & \text{---} \\ \text{---} & x_2^T & \text{---} \\ & \vdots & \\ \text{---} & x_n^T & \text{---} \end{bmatrix}$$

Matrix/Norm Notation (MEMORIZE/STUDY THIS)

- We showed how to express various quantities in matrix notation:
  - Linear regression prediction for one example: $\hat{y}_i = w^T x_i$
  - Linear regression prediction for all 'n' examples: $\hat{y} = Xw$
  - Linear regression residual vector: $r = Xw - y$
  - Sum of residuals squared in linear regression model:

$$f(w) = \sum_{i=1}^{n} \left(\sum_{j=1}^{d} w_j x_{ij} - y_i \right)^2 = \|Xw - y\|^2$$

  - Today: derive gradient and least squares solution in matrix notation.

Digression: Matrix Algebra Review

- Quick review of linear algebra operations we'll use:
  - If 'a' and 'b' are vectors, and 'A' and 'B' are matrices then:

$$a^T b = b^T a$$

$$\|a\|^2 = a^T a$$

$$(A+B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

$$(A+B)(A+B) = AA + BA + AB + BB$$

$$a^T \underbrace{Ab}_{\text{vector}} = \underbrace{b^T A^T}_{\text{vector}} a$$

Sanity check:

ALWAYS CHECK THAT DIMENSIONS MATCH

(if not, you did something wrong)

Linear and Quadratic Gradients

- From these rules we have

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 = \frac{1}{2}\|Xw - y\|^2 = \frac{1}{2}w^T \underbrace{X^T X}_{\text{matrix 'A'}} w - w^T \underbrace{X^T y}_{\text{vector 'b'}} + \underbrace{\frac{1}{2}y^T y}_{\text{scalar 'c'}}$$

see post-lecture slide for black-to-blue steps

$$= \frac{1}{2}w^T \underbrace{A}_{X^T X} w + w^T \underbrace{b}_{-X^T y} + \underbrace{c}_{\frac{1}{2}y^T y}$$

These are scalars so dimensions match.

Note the "–" added to the definition of b, flipping the sign

- How do we compute gradient?

Let's first do it with $d=1$:

$$f(w) = \frac{1}{2}waw + wb + c$$

$$= \frac{1}{2}aw^2 + wb + c$$

$$f'(w) = aw + b + 0$$

Here are the generalizations to 'd' dimensions:

$\nabla[c] = 0$ (zero vector)

$\nabla[w^T b] = b$

$\nabla[\frac{1}{2}w^T A w] = Aw$ (if A is symmetric)

Which it is because it is $X^T X$, which is symmetric

Full derivations are on webpage in notes on linear and quadratic gradients.

Linear and Quadratic Gradients

- From these rules we have (see post-lecture slide for steps):

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 = \frac{1}{2}\|Xw - y\|^2 = \frac{1}{2}w^T\underbrace{X^TX}_{\text{matrix 'A'}}w - w^T\underbrace{X^Ty}_{\text{vector 'b'}} + \underbrace{\frac{1}{2}y^Ty}_{\text{scalar 'c'}}$$

$$= \frac{1}{2}w^T\underset{X^TX}{A}w + w^T\underset{-X^Ty}{b} + \underset{\frac{1}{2}y^Ty}{c}$$

- Gradient is given by: $\nabla f(w) = Aw + b + 0$

- Using definitions of 'A' and 'b': $= \underbrace{X^TXw - X^Ty}_{\substack{\text{Sanity check: all dimensions match} \\ (d\times n)(n\times d)(d\times 1) - (d\times n)(n\times 1)}}$

Normal Equations for Least Squares Solution

- Set gradient equal to zero to find the "critical" points:

$$X^T X w - X^T y = 0$$

- We now move terms not involving 'w' to the other side:

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

From last time

For linear least squares we have:

$$\nabla f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \left(\sum_{j=1}^{d} w_j x_{ij} - y_i \right) x_{i1} \\ \sum_{i=1}^{n} \left(\sum_{j=1}^{d} w_j x_{ij} - y_i \right) x_{i2} \\ \vdots \\ \sum_{i=1}^{n} \left(\sum_{j=1}^{d} w_j x_{ij} - y_i \right) x_{id} \end{bmatrix}$$

Normal Equations for Least Squares Solution

- Set gradient equal to zero to find the "critical" points:

$$X^T X w - X^T y = 0$$

- We now move terms not involving 'w' to the other side:

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

Virtually only ML alg where you just get the optimal w (and with one line of code!)

- This is a set of 'd' linear equations called the normal equations.
  - This a linear system like "Ax = b" from Math 152 (A is $X^T X$, x is w confusingly)
    - You can use Gaussian elimination to solve for 'w'.
  - In Julia, the "\" command can be used to solve linear systems:

In linear algebra, the variables you adjust are x (in ML, w)

In Python: numpy.linalg.solve

Train: $w = (X'X)\backslash(X'y)$ Predict: $yhat = X_{test} * w$

Incorrect Solutions to Least Squares Problem

The least squares objective is $f(w) = \frac{1}{2}\|Xw - y\|^2$

The minimizers of this objective are solutions to the linear system:

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

The following are not the solutions to the least squares problem:

$w = (X^T X)^{-1}(X^T y)$ (only true if $X^T X$ is invertible) We'll talk about when this is true later

$w X^T X = X^T y$ (matrix multiplication is not commutative, dimensions don't even match)

$w = \frac{X^T y}{X^T X}$ (you cannot divide by a matrix)

Least Squares Cost

- Cost of solving "normal equations" $X^TXw = X^Ty$?
- Forming X^Ty vector costs $O(nd)$.
  – It has 'd' elements, and each is an inner product between 'n' numbers.
- Forming matrix X^TX costs $O(nd^2)$.
  – It has d^2 elements, and each is an inner product between 'n' numbers.
- Solving a d x d system of equations costs $O(d^3)$.
  – Cost of Gaussian elimination on a d-variable linear system.
  – Other standard methods have the same cost.
- Overall cost is $O(nd^2 + d^3)$.
  – Which term dominates depends on 'n' and 'd'.

Least Squares Issues

- Issues with least squares model:
  - Solution might not be unique.
  - It is sensitive to outliers.
  - It always uses all features.
  - Data might so big we cannot store X^TX.
    - requires $O(d^2)$, which is bad (e.g. for 10 million features)
    - Or you cannot afford the $O(nd^2 + d^3)$ cost.
  - It might predict outside range of y_i values.
    - For some applications, only positive y_i values are valid.
  - It assumes a linear relationship between x_i and y_i.

X is $n \times d$
so X^T is $d \times n$
and X^TX is $d \times d$.

Non-Uniqueness of Least Squares Solution

- Why is the solution vector 'w' not unique?
  - Imagine having two features that are identical for all examples.
  - I can increase weight on one feature, and decrease it on the other, without changing predictions.

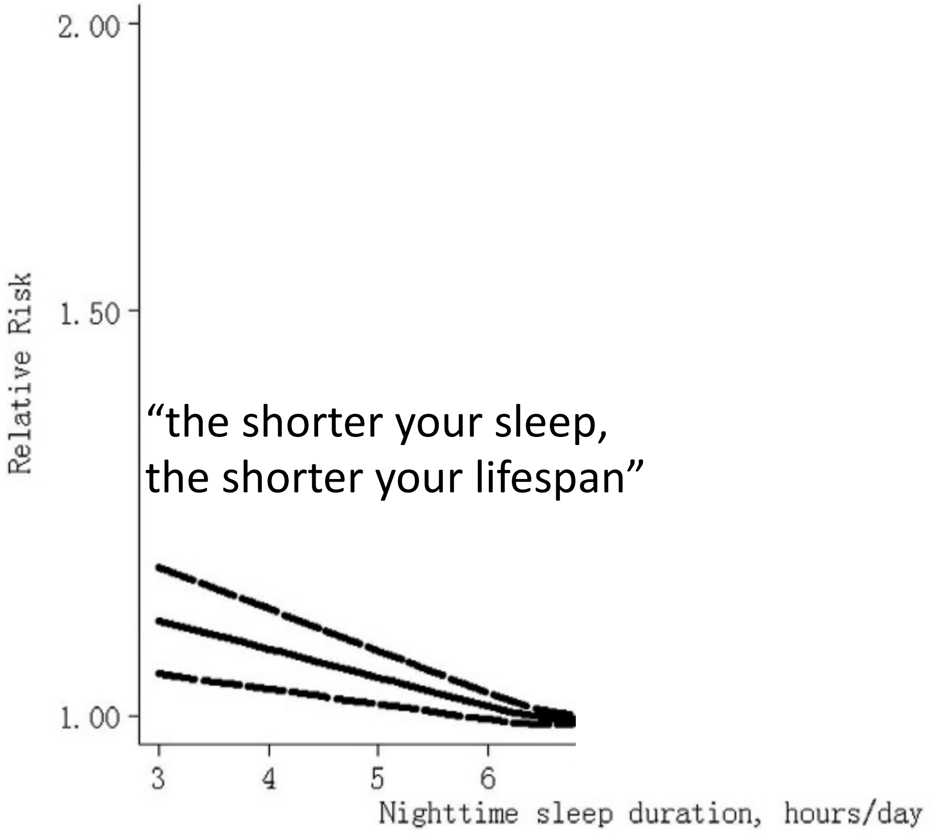
$$\hat{y}_i = w_1 x_{i1} + w_2 \underbrace{x_{i1}}_{\text{copy}} = (w_1 + w_2) x_{i1} + 0 x_{i1}$$

  - In this setting, if (w_1, w_2) is a solution then $(w_1+w_2, 0)$ is another solution.
  - This is special case of features being "collinear":
    - One feature is a linear function of the others.
- But, any 'w' where $\nabla f(w) = 0$ is a global minimizer of 'f'.
  - This is due to convexity of 'f', which we will discuss later.

Next Topic: Non-Linear Regression

Motivation: Non-Linear Regression

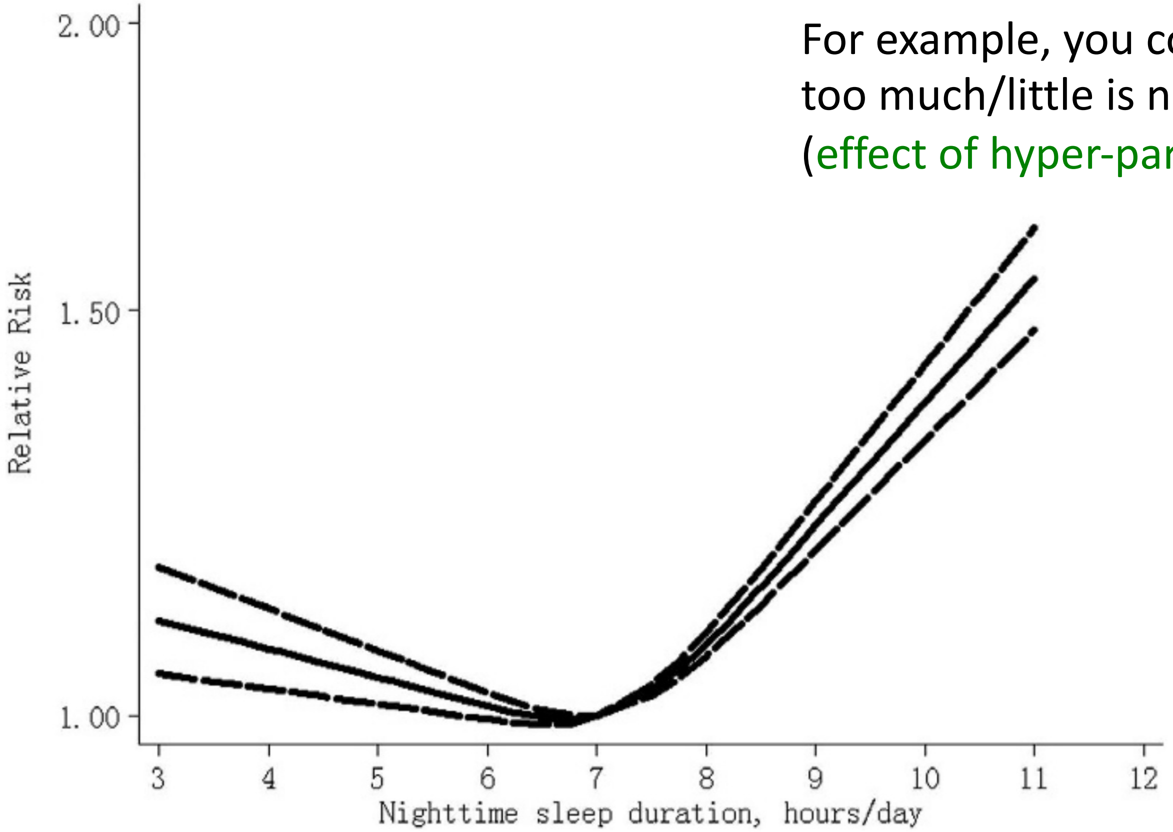
- Many relationships are approximated well by linear function.



https://www.nature.com/articles/srep21480.pdf

Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  - But many are also highly non-linear.

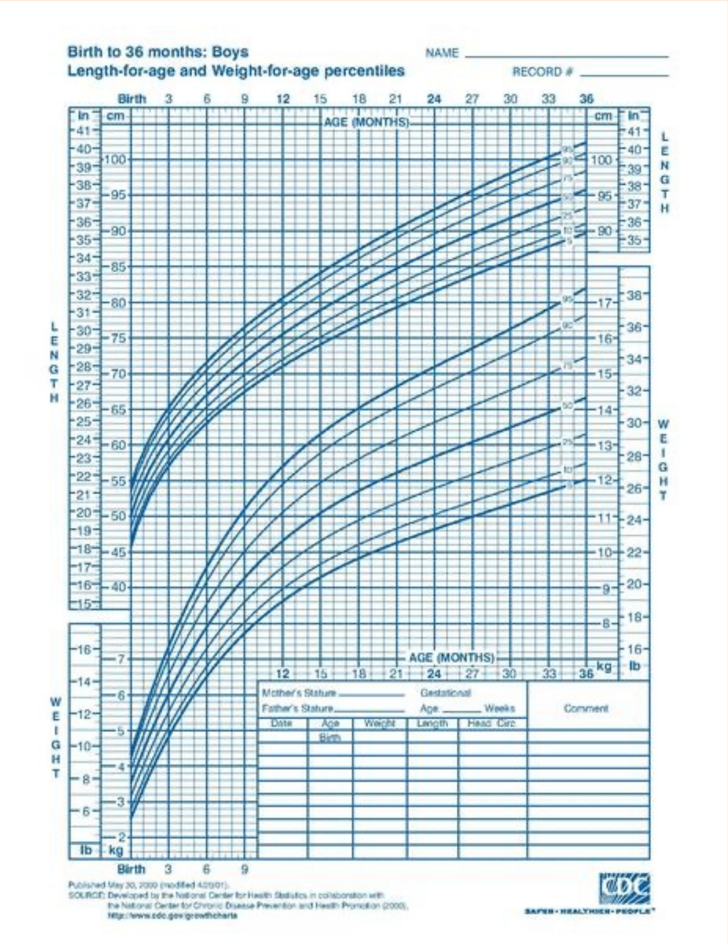


For example, you could have a “u-shape” when too much/little is not good.
(effect of hyper-parameters usually looks like this)

https://www.nature.com/articles/srep21480.pdf

Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  – But many are also highly non-linear.



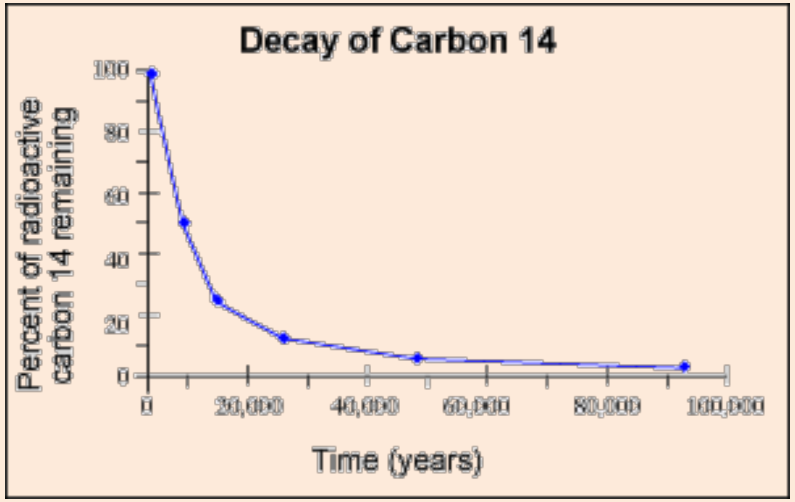
Slope could slowly change
or reach asymptote.

https://commons.wikimedia.org/wiki/File:CDC_growth_chart_b

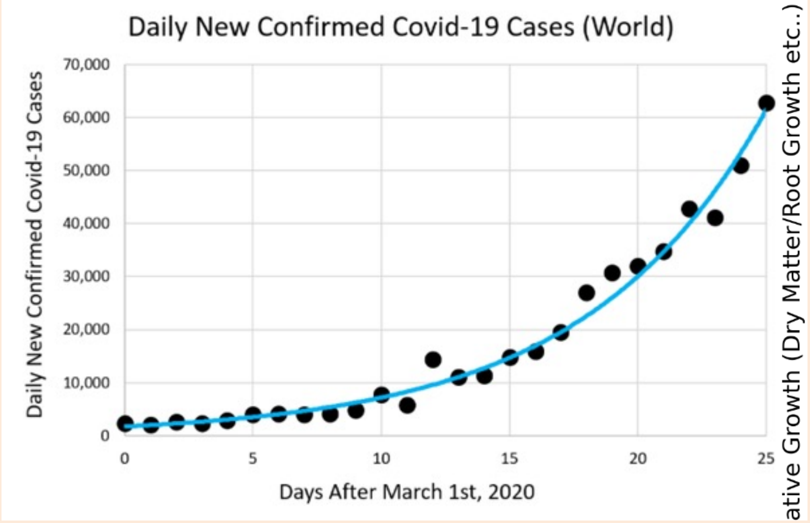
Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  - But many are also highly non-linear.

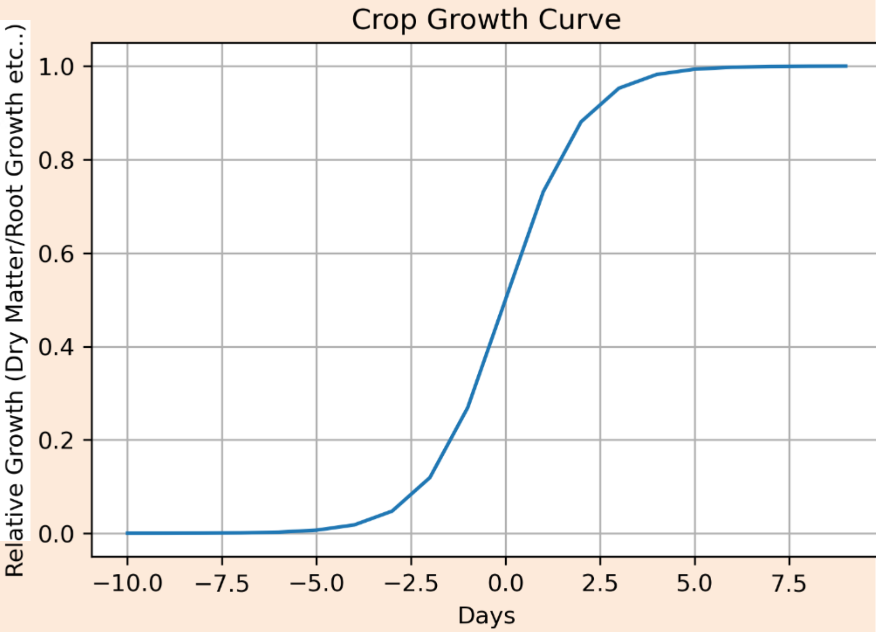
“geometric decay”



“exponential growth”



“sigmoidal growth”.

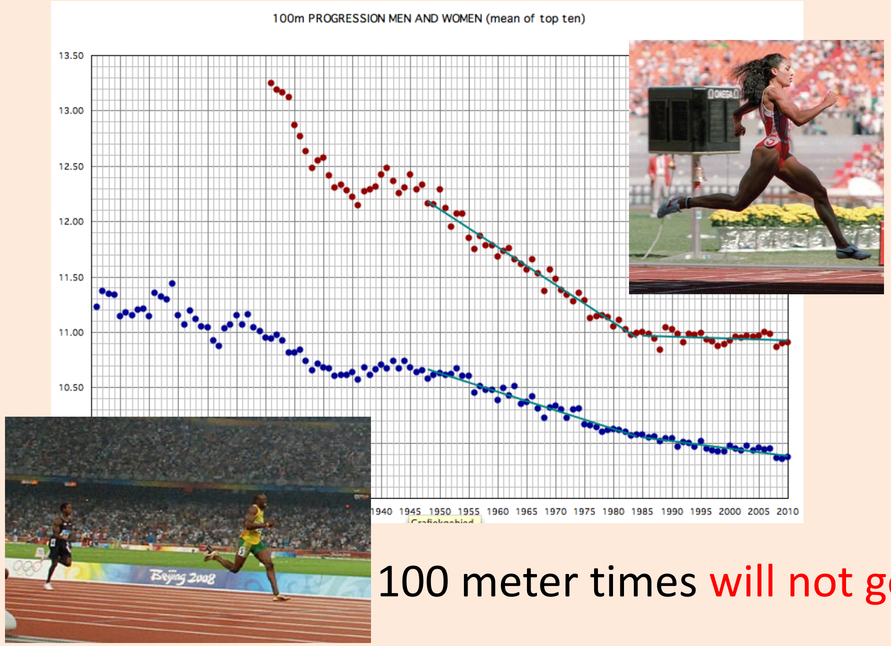


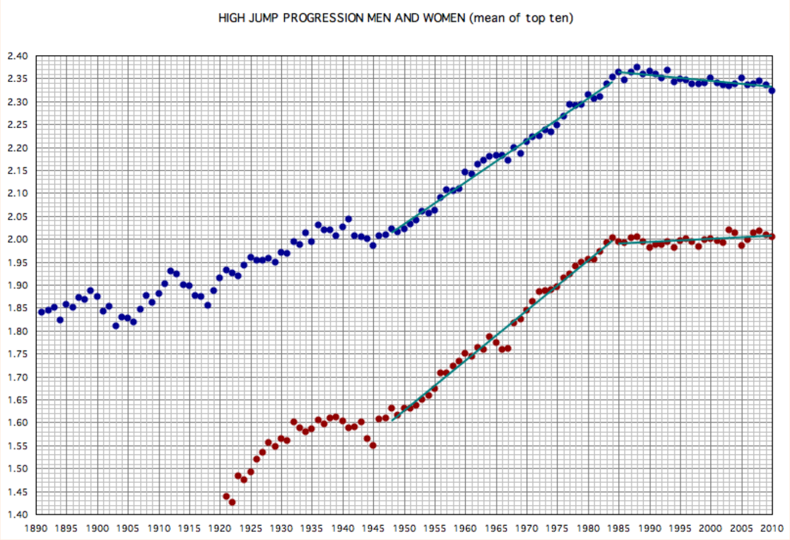
http://www.biology.arizona.edu/biomath/tutorials/Applications/Carbon.html
https://www.nytimes.com/2021/05/14/learning/the-math-of-ending-the-pandemic-exponential-growth-and-decay.html
https://stackoverflow.com/questions/69535905/how-to-dynamically-fit-sigmoid-growth-curve-for-crop-plants

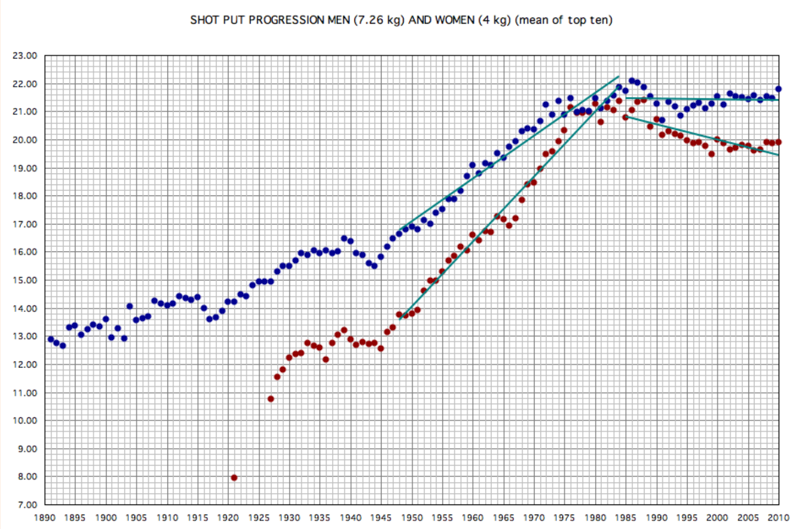
Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  - But many are also highly non-linear.

"Piecewise linear": different pieces follow different linear functions.
(or be linear up to asymptote or phase transition)







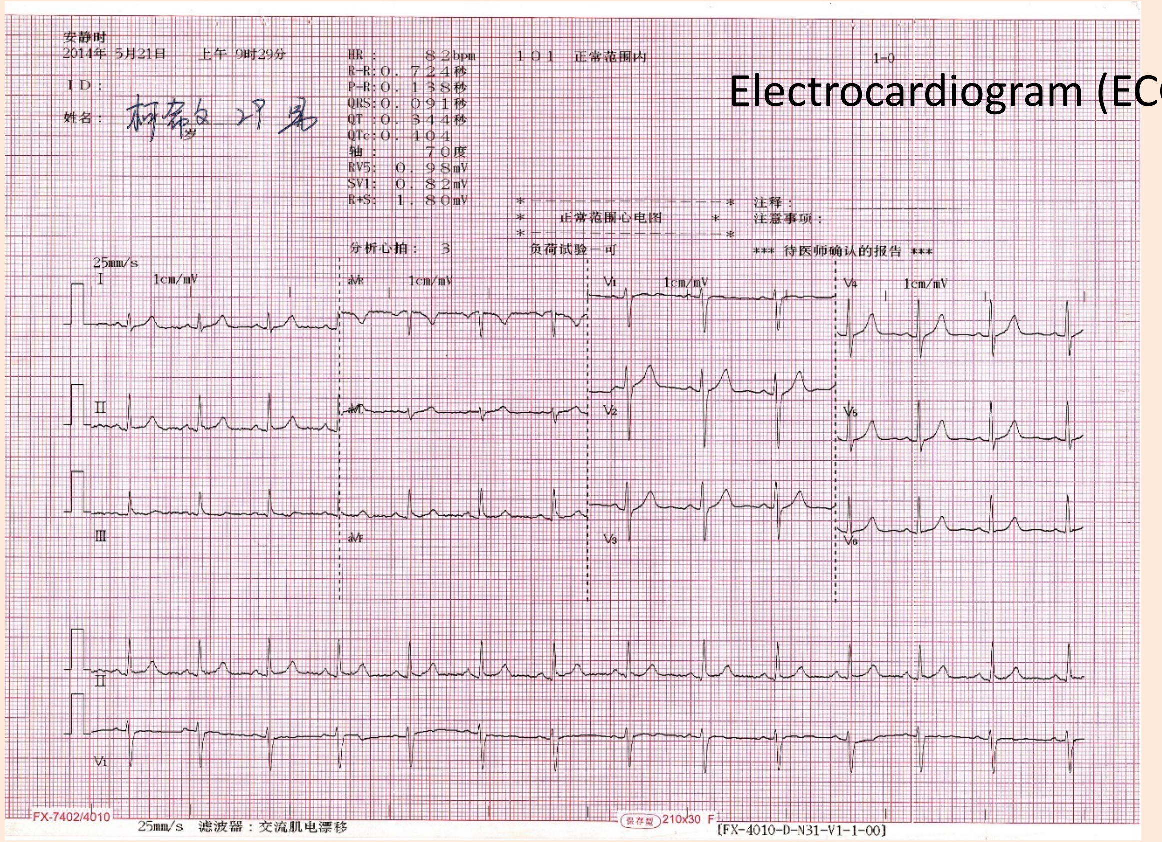
100 meter times will not go negative!

http://www.at-a-lanta.nl/weia/Progressie.html
https://en.wikipedia.org/wiki/Usain_Bolt
http://www.britannica.com/biography/Florence-Griffith-Joyner

Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  – But many are also highly non-linear.

"Periodic" signals.



Electrocardiogram (ECG)

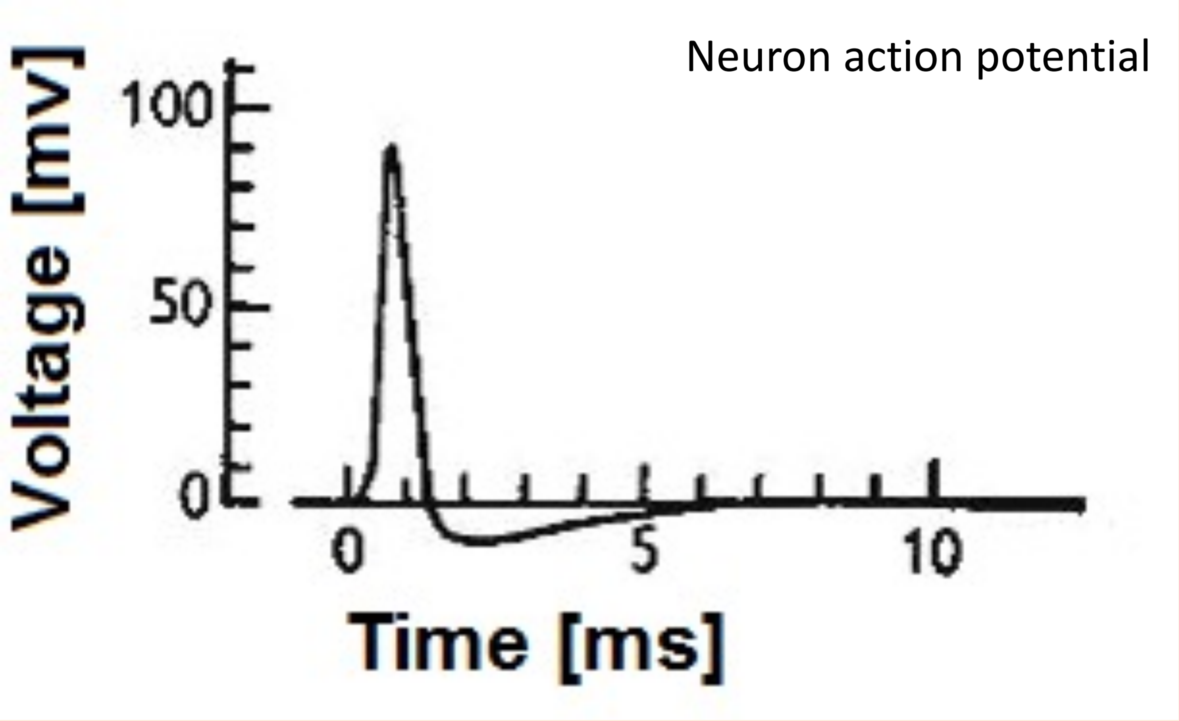
https://en.wikipedia.org/wiki/Electrocardiography#/media/File:Normal_12_lead_EKG.jpg

Electroencephalography (EEG)

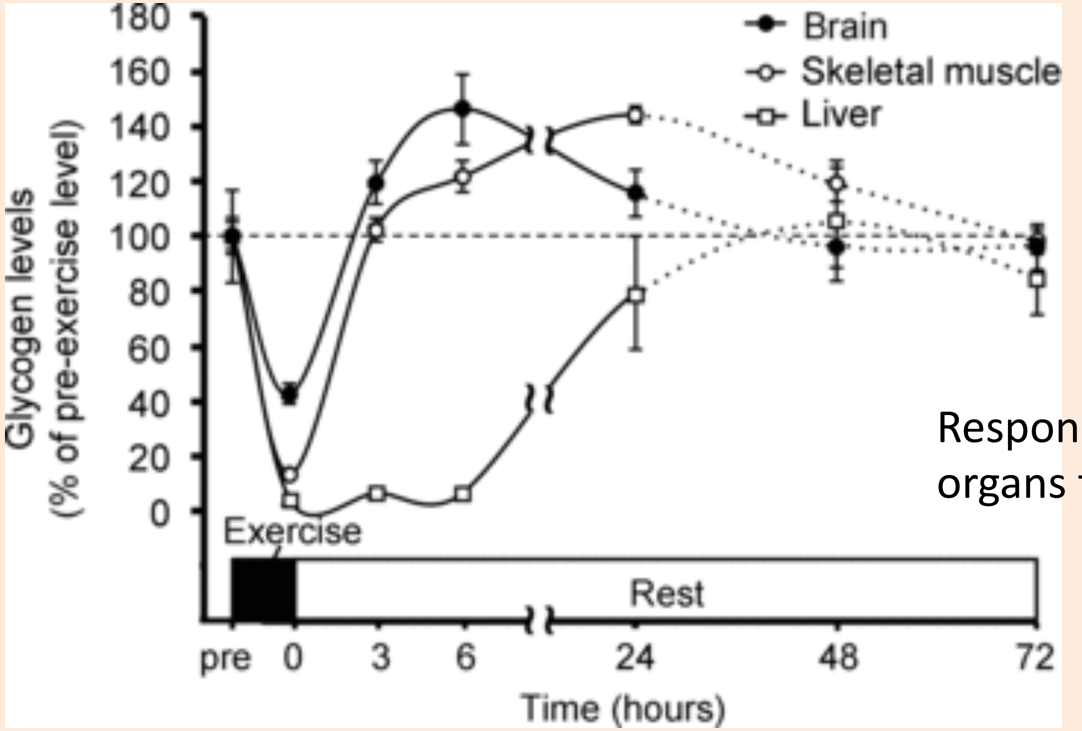
Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  - But many are also highly non-linear.

"Spike then recover"



Neuron action potential



Response of organs to exercise

https://en.wikipedia.org/wiki/Biological_neuron_model
https://physoc.onlinelibrary.wiley.com/doi/full/10.1113/jphysiol.2011.217919

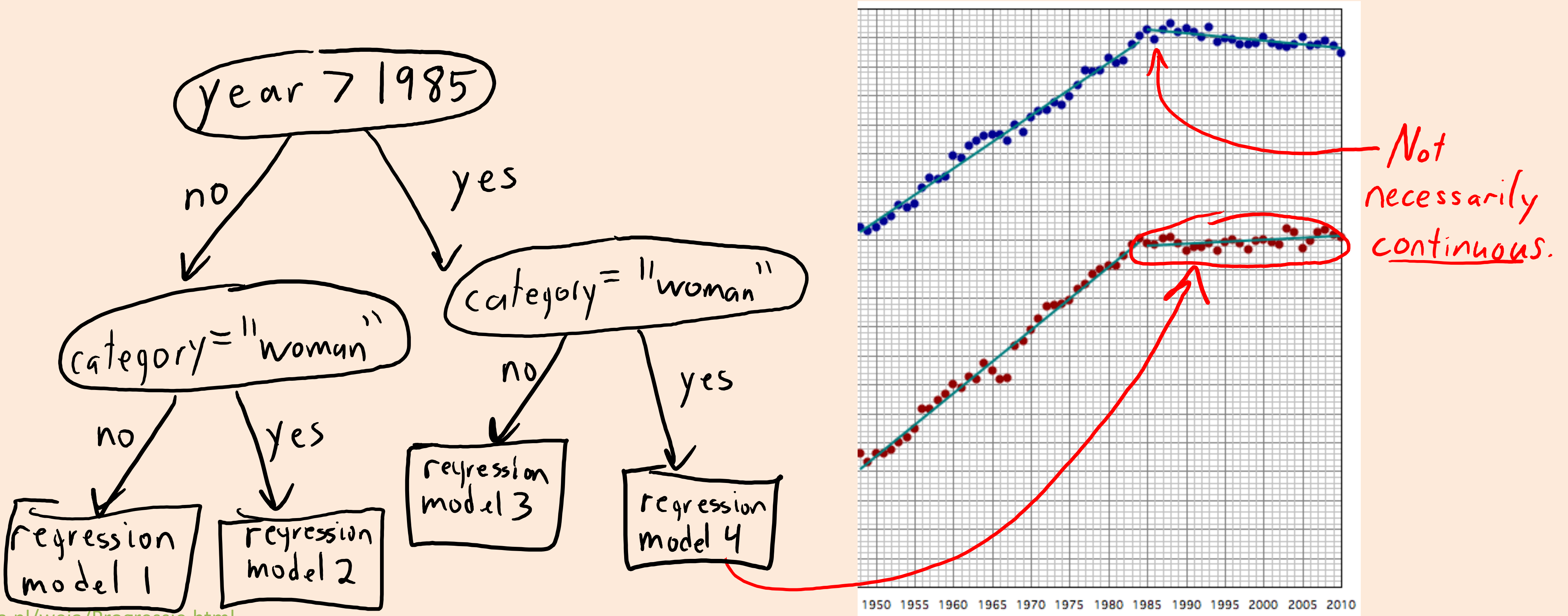
Adapting Counting/Distance-Based Methods

- Can adapt classification methods to perform non-linear regression:

http://www.at-a-lanta.nl/weia/Progressie.html

Adapting Counting/Distance-Based Methods

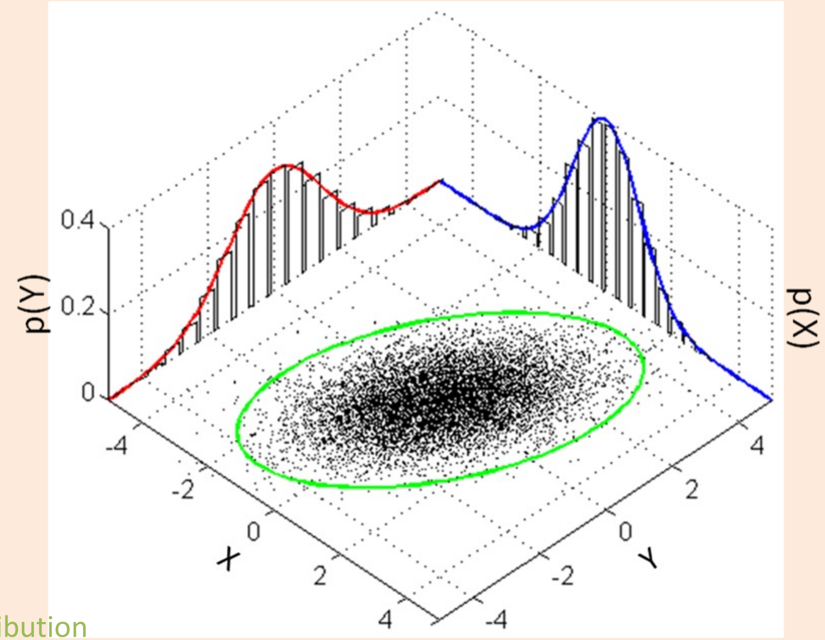
- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.



http://www.at-a-lanta.nl/weia/Progressie.html

Adapting Counting/Distance-Based Methods

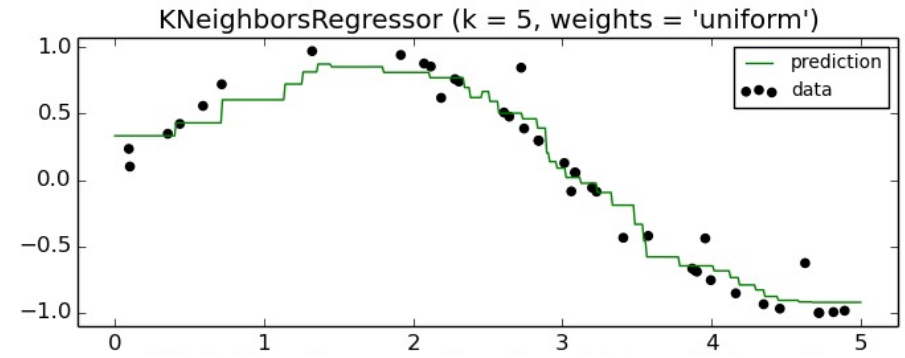
- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
    - Take CPSC 440.



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

Adapting Counting/Distance-Based Methods

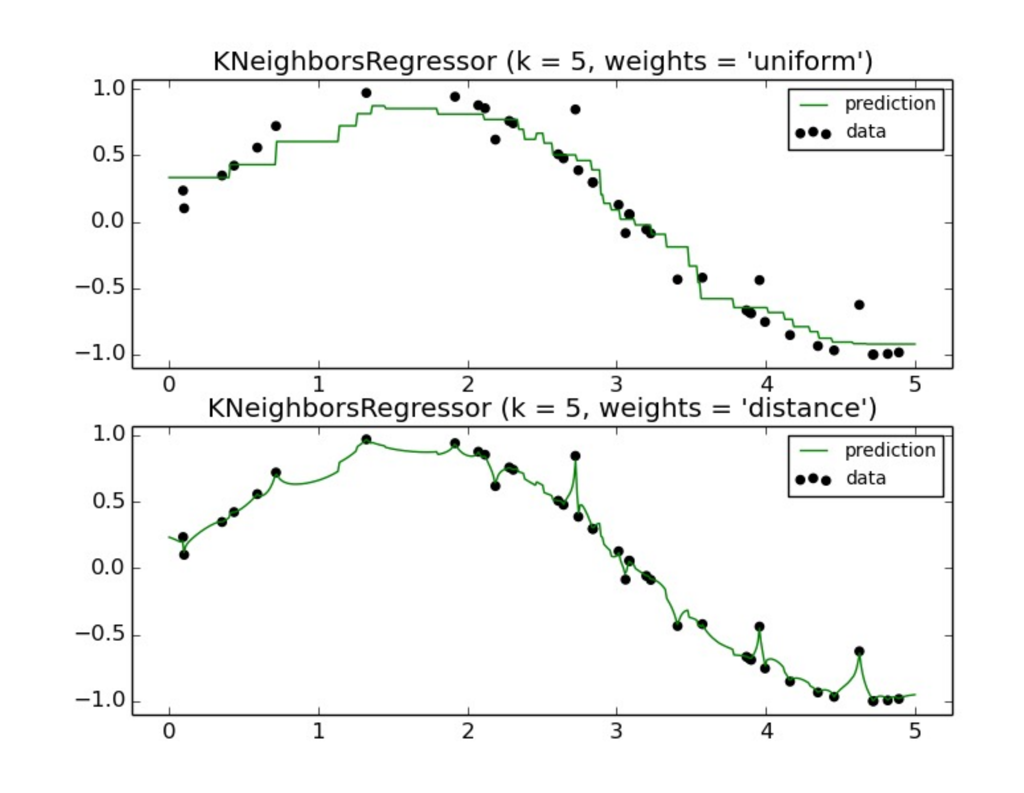
- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression:
      - Find 'k' nearest neighbours of $\tilde{x}_i$.
      - Return the mean of the corresponding y_i.



http://scikit-learn.org/stable/modules/neighbors.html

Adapting Counting/Distance-Based Methods

- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
      - Close points 'j' get more "weight" w_{ij}.

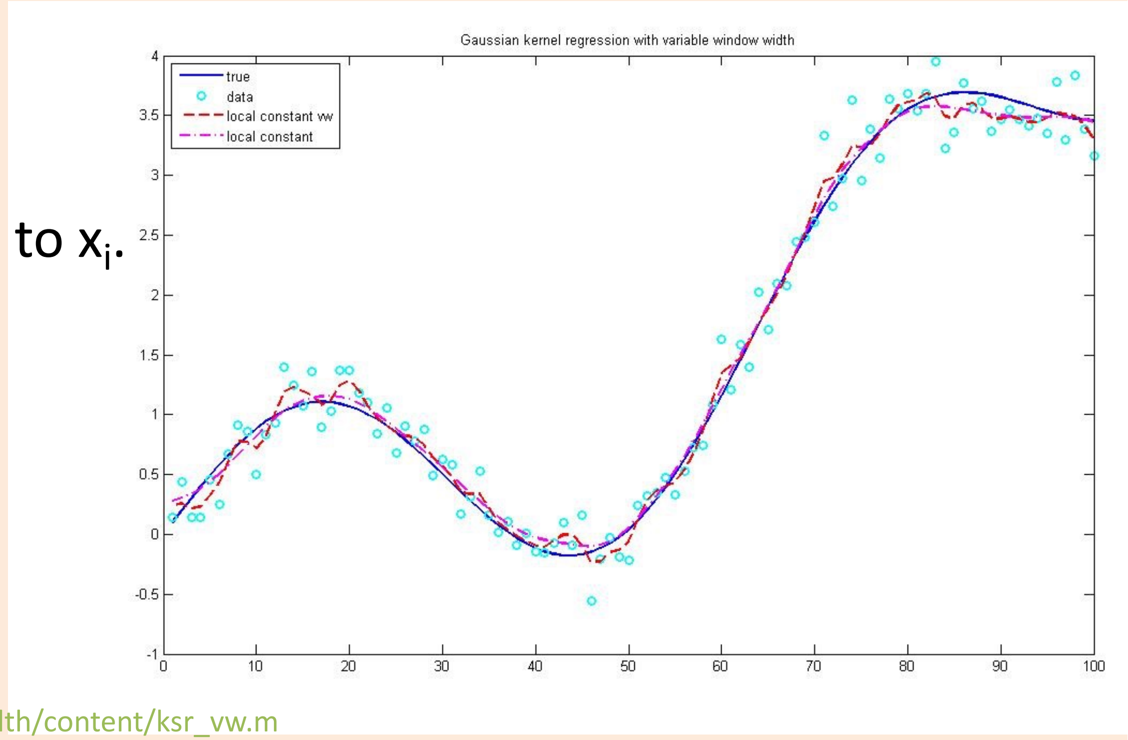


http://scikit-learn.org/stable/modules/neighbors.html

Adapting Counting/Distance-Based Methods

- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - 'Nadaraya-Watson': weight *all* y_i by distance to x_i.

$$\hat{y}_i = \frac{\sum_{j=1}^{n} v_{ij} y_j}{\sum_{j=1}^{n} v_{ij}}$$

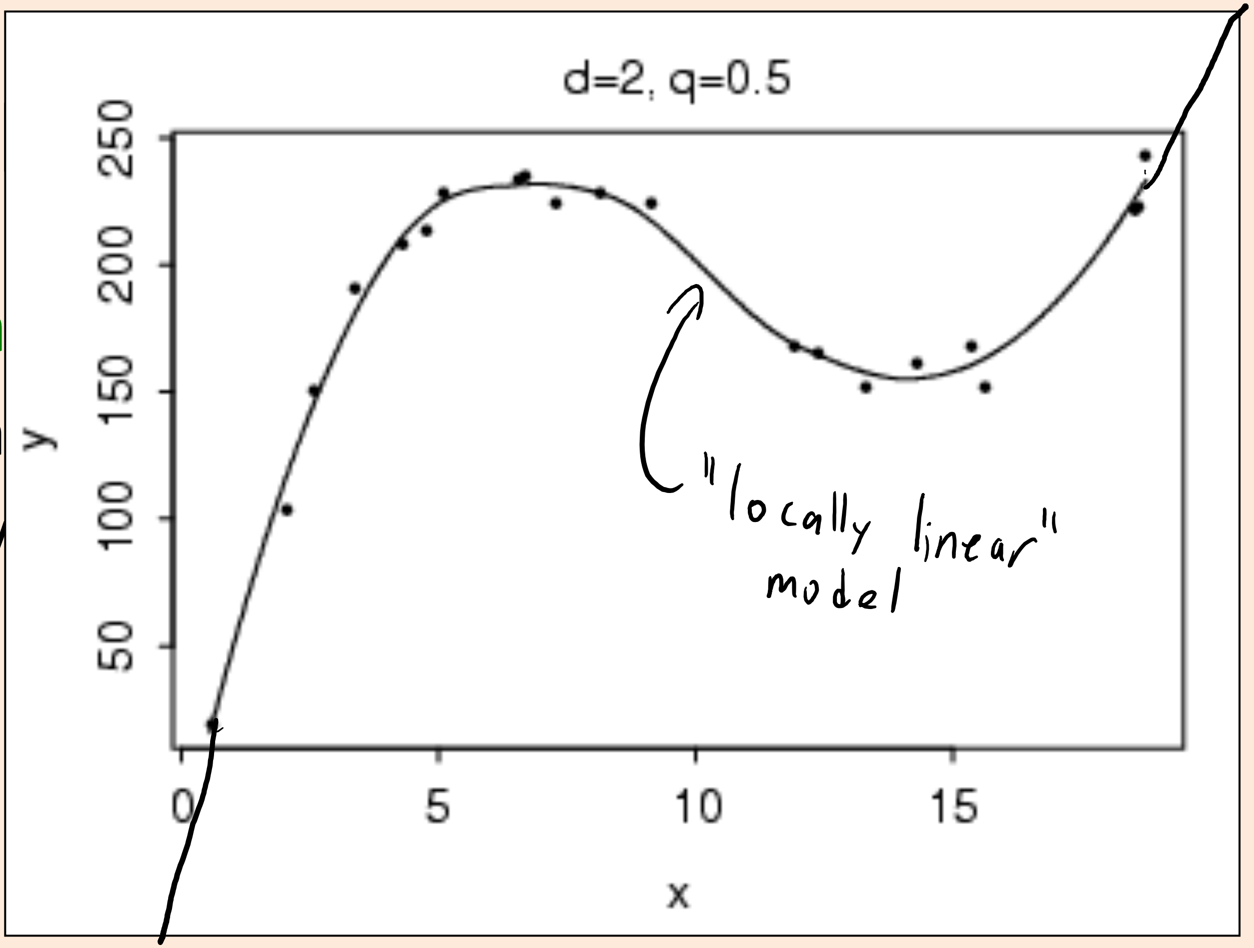


http://www.mathworks.com/matlabcentral/fileexchange/35316-kernel-regression-with-variable-window-width/content/ksr_vw.m

Adapting Counting/

- Can adapt classification meth
  - Regression tree: tree with mea
  - Probabilistic models: fit $p(x_i \mid y$
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - 'Nadaraya-Watson': weight *all* y_i
    - 'Locally linear regression': for each x_i, fit a linear model weighted by distance.

(Better than KNN and NW at boundaries.)



http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd423.htm

Adapting Counting/Distance-Based Methods

- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - 'Nadaraya-Watson': weight *all* y_i by distance to x_i.
    - 'Locally linear regression': for each x_i, fit a linear model weighted by distance.
      (Better than KNN and NW at boundaries.)
  - Ensemble methods:
    - Can improve performance by averaging predictions across regression models.

Adapting Counting/Distance-Based Methods

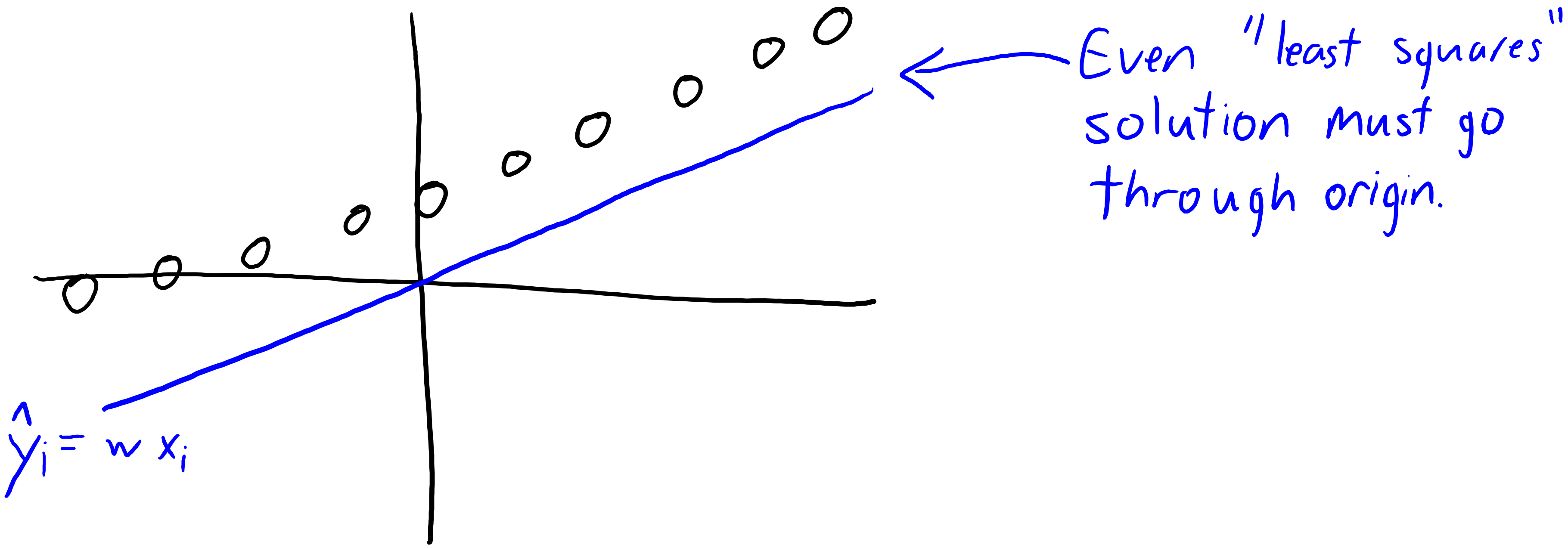
- Applications of non-linear regression (we will see many more):
  - Regression forests for fluid simulation:
  - KNN for image completion:
    - Combined with "graph cuts" and "Poisson blending".
    - See also "PatchMatch".
  - KNN regression for "voice photoshop":
    - Combined with "dynamic time warping" and "Poisson blending".

- We will first focus on linear models with non-linear transforms.
  - These are the building blocks for more advanced methods.

http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd423.htm

Why don't we have a y-intercept?

- Linear model is $\hat{y}_i = wx_i$ instead of $\hat{y}_i = wx_i + w_0$ with y-intercept w_0.
- Without an intercept, if $x_i = 0$ then we must predict $\hat{y}_i = 0$.



Why don't we have a y-intercept?

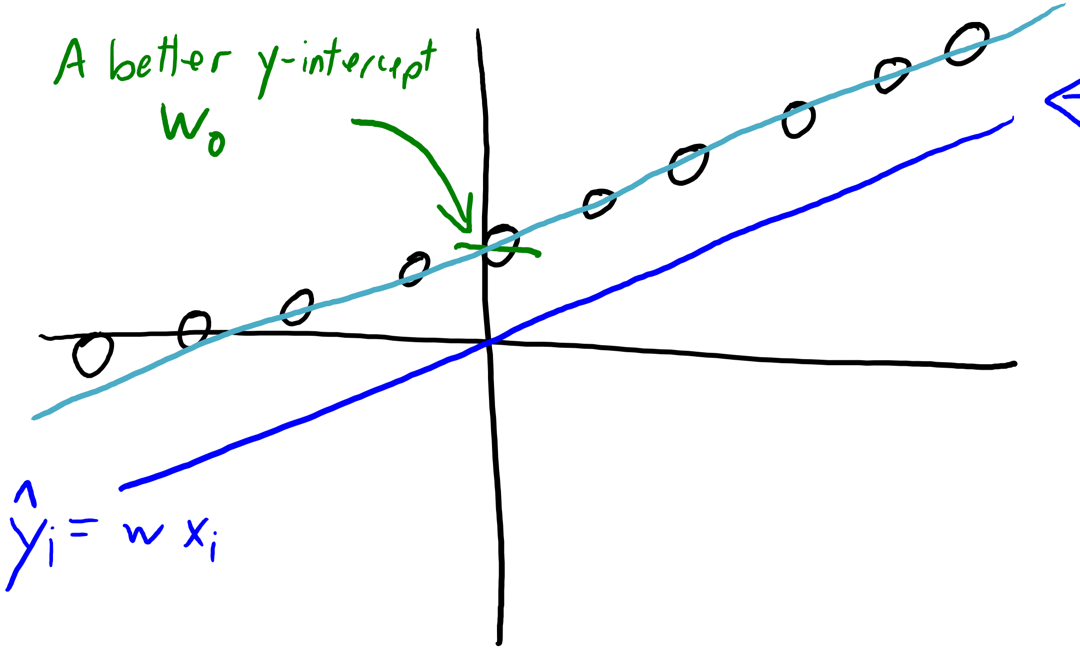
– Linear model is $\hat{y}_i = wx_i$ instead of $\hat{y}_i = wx_i + w_0$ with y-intercept w_0.

– Without an intercept, if $x_i = 0$ then we must predict $\hat{y}_i = 0$.

Adding y-intercept fixes this.

$\hat{y}_i = w^T x_i + w_0$

Even "least squares" solution must go through origin.



Adding a Y-Intercept ("Bias") Variable

- Simple trick to add a y-intercept ("bias") variable:
  - Make a new matrix "Z" with an extra feature that is always "1".

$$X = \begin{bmatrix} -0.1 \\ 0.3 \\ 0.2 \end{bmatrix} \qquad Z = \begin{bmatrix} \underbrace{\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}}_{\text{"always 1"}} & \underbrace{\begin{matrix} -0.1 \\ 0.3 \\ 0.2 \end{matrix}}_{X} \end{bmatrix}$$

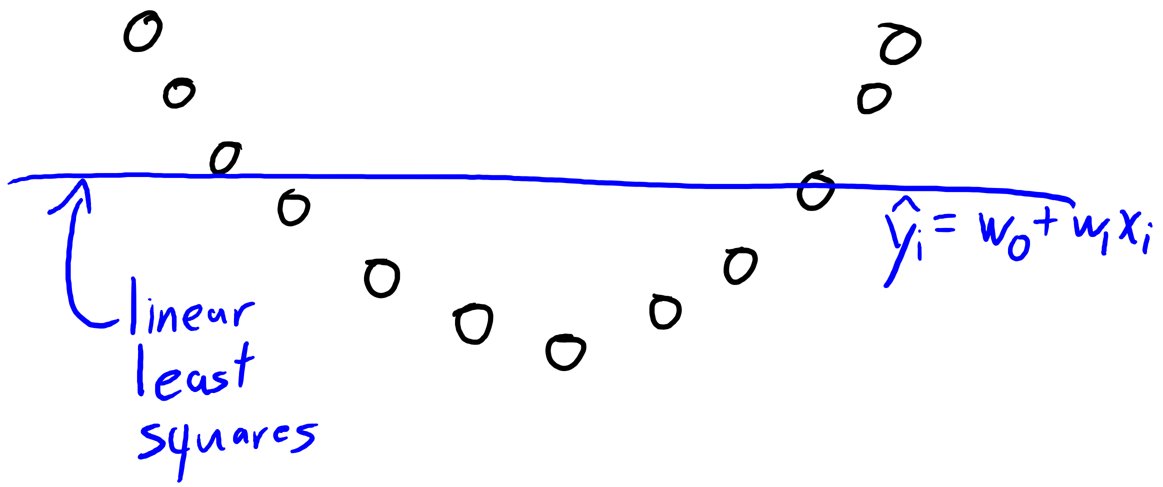
- Now use "Z" as your features in linear regression.
  - We will use 'v' instead of 'w' as regression weights when we use features 'Z'.

$$\hat{y}_i = \underset{\downarrow\, w_0}{v_1}\, \underset{\downarrow\, 1}{z_{i1}} + \underset{\downarrow\, w_1}{v_2}\, \underset{\downarrow\, x_{i1}}{z_{i2}} = w_0 + w_1 x_{i1}$$

- So we can have a non-zero y-intercept by changing features.
  - This means we can ignore the y-intercept to make cleaner derivations/code.

Motivation: Limitations of Linear Models

- On many datasets, y_i is not a linear function of x_i.



- A quadratic function would be a better fit for this dataset.

Non-Linear Feature Transforms

- Can we use linear least squares to fit a quadratic model?

$$\hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2$$

  - Notice that this is a non-linear function of x_i but a linear function of 'w'.

- So you can implement this by changing the features:

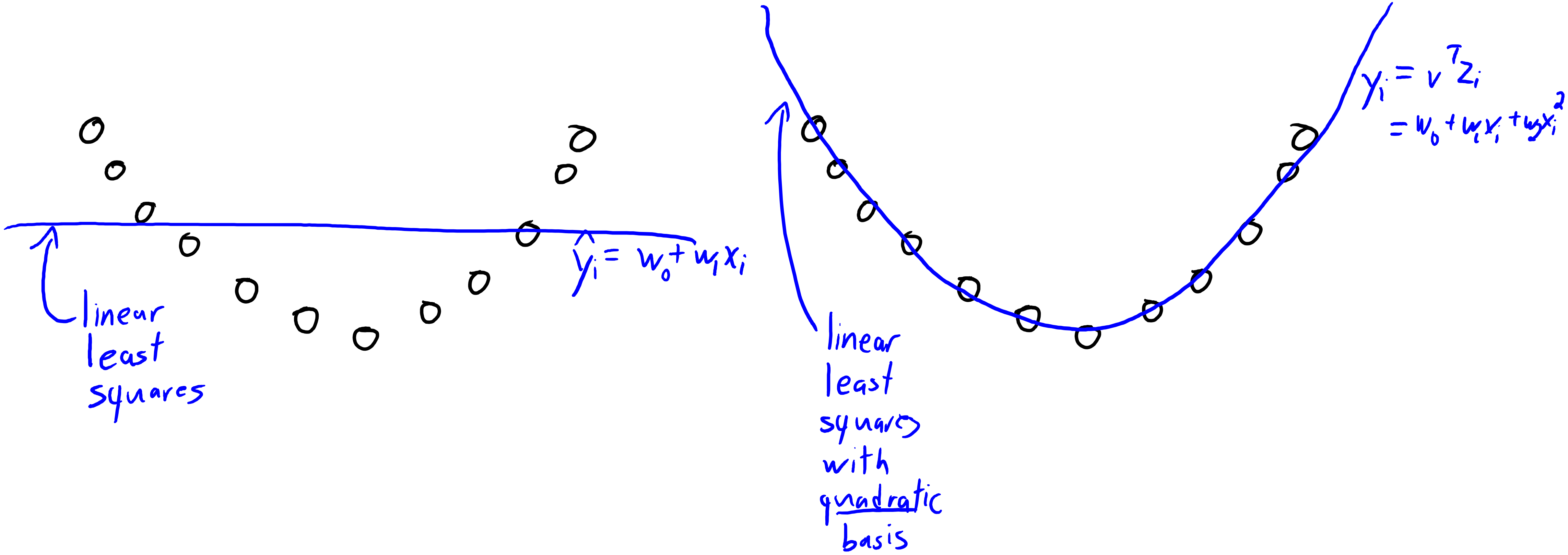
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad Z = \begin{array}{ccc} \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix} \\ \text{y-int} \quad\quad x \quad\quad x^2 \end{array}$$

  - Fit new parameters 'v' under "change of basis": solve $Z^TZv = Z^Ty$.

- It's a linear function of w, but a quadratic function of x_i.

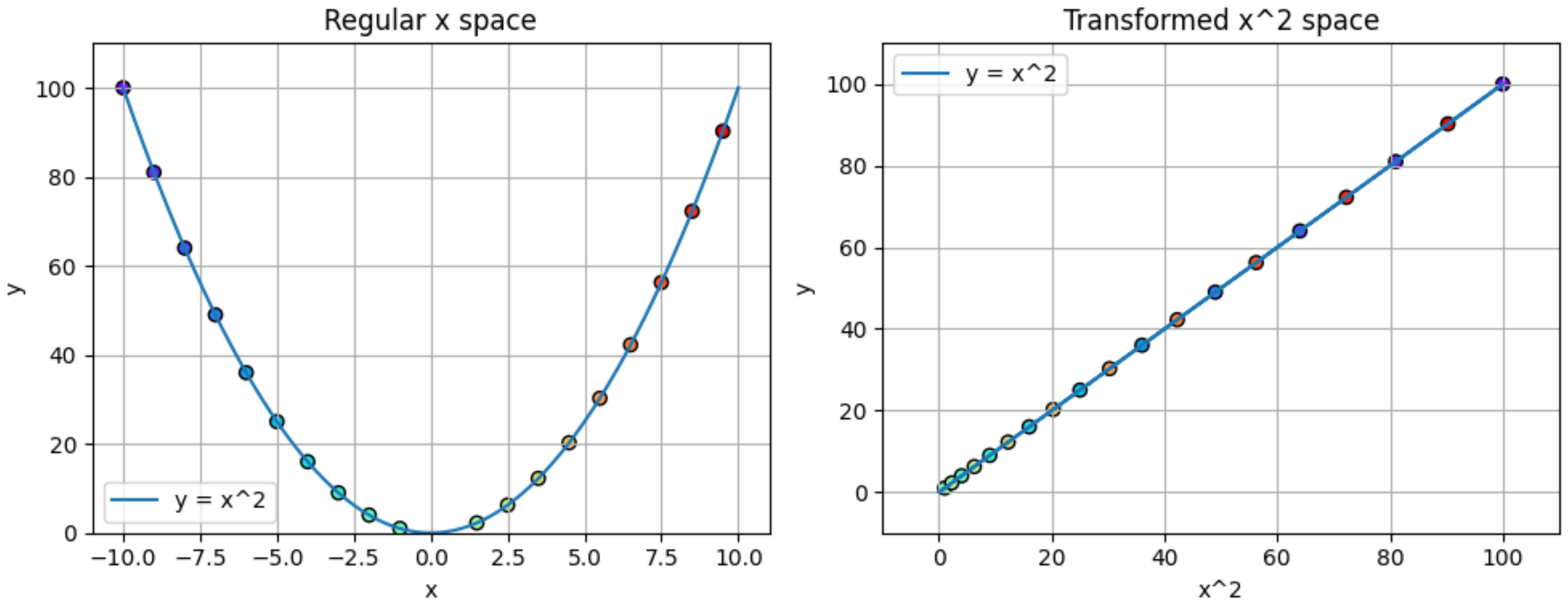
$$\hat{y}_i = v^T z_i = \underbrace{v_1}_{w_0} \underbrace{z_{i1}}_{1} + \underbrace{v_2}_{w_1} \underbrace{z_{i2}}_{x_i} + \underbrace{v_3}_{w_2} \underbrace{z_{i3}}_{x_i^2}$$

Non-Linear Feature Transforms



To predict on new data $\tilde{X}$, form $\tilde{Z}$ from $\tilde{X}$ and take $y = \tilde{Z}v$

Non-Linear Feature Transforms



- It's a linear function of w, but a quadratic function of x_i.

$$\hat{y}_i = v^T z_i = \underbrace{v_1}_{w_0} \underbrace{z_{i1}}_{1} + \underbrace{v_2}_{w_1} \underbrace{z_{i2}}_{x_i} + \underbrace{v_3}_{w_2} \underbrace{z_{i3}}_{x_i^2}$$

Non-Linear Feature Transforms

To predict on new data $\tilde{X}$, form $\tilde{Z}$ from $\tilde{X}$ and take $y = \tilde{Z}v$

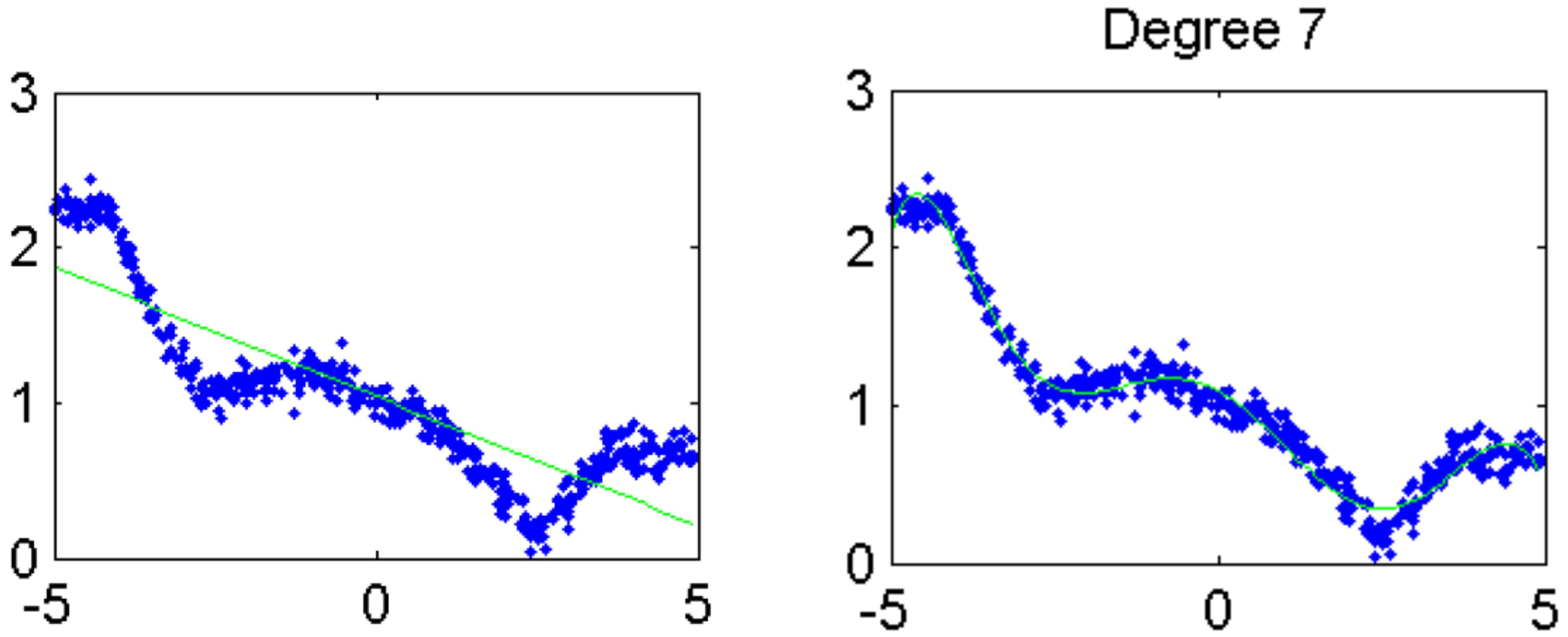
General Polynomial Features (d=1)

- We can have a polynomial of degree 'p' by using these features:

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

- There are polynomial basis functions that are numerically nicer:
  - Such as Lagrange polynomials (see CPSC 303).

General Polynomial Features



- If you have more than one feature, you can include interactions:
  - With p=2, in addition to $(x_{i1})^2$ and $(x_{i2})^2$ you could include $x_{i1}x_{i2}$.

"Change of Basis" Terminology

- Instead of "nonlinear feature transform", in machine learning it is common to use the expression "change of basis".
  - The z_i are the "coordinates in the new basis" of the training example.

- "Change of basis" means something different in math:
  - Math: basis vectors must be linearly independent (in ML we don't care).
  - Math: change of basis must span the same space (in ML we change space).

- Unfortunately, saying "change of basis" in ML is common.
  - When I say "change of basis", just think "nonlinear feature transform".

Linear Basis vs. Nonlinear Basis

Usual linear Regression

Train:

- Use 'X' and 'y' to find 'w'

Test:

- Use $\tilde{X}$ and 'w' to find $\hat{y}$

Linear regression with change of basis

Train:

- Use 'X' to find 'Z'
- Use 'Z' and 'y' to find 'v'

Test:

- Use '$\tilde{X}$' to find '$\tilde{Z}$'
- Use $\tilde{Z}$ and 'v' to find $\hat{y}$

Change of Basis Notation (MEMORIZE)

- Linear regression with original features:
  - We use 'X' as our "n by d" data matrix, and 'w' as our parameters.
  - We can find d-dimensional 'w' by minimizing the squared error:

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

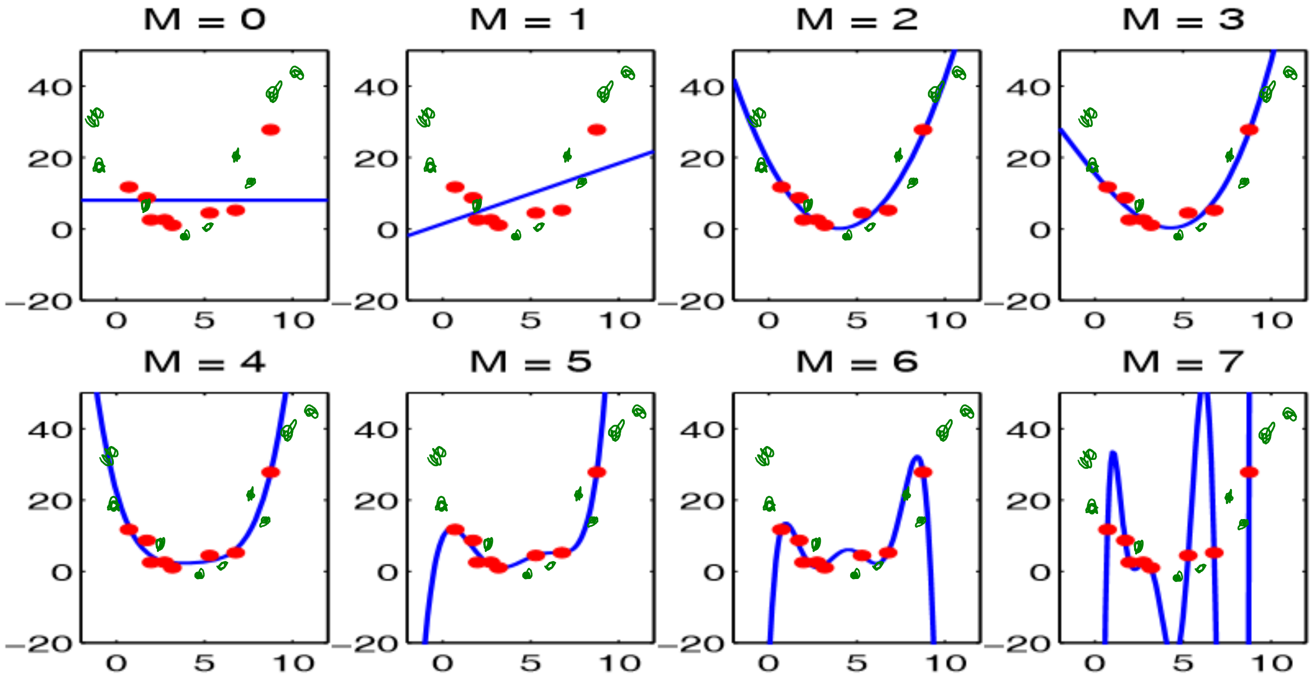
- Linear regression with nonlinear feature transforms:
  - We use 'Z' as our "n by k" data matrix, and 'v' as our parameters.
  - We can find k-dimensional 'v' by minimizing the squared error:

$$f(v) = \frac{1}{2}\|Zv - y\|^2$$

- Notice that in both cases the target is still 'y'.

Degree of Polynomial and Fundamental Trade-Off

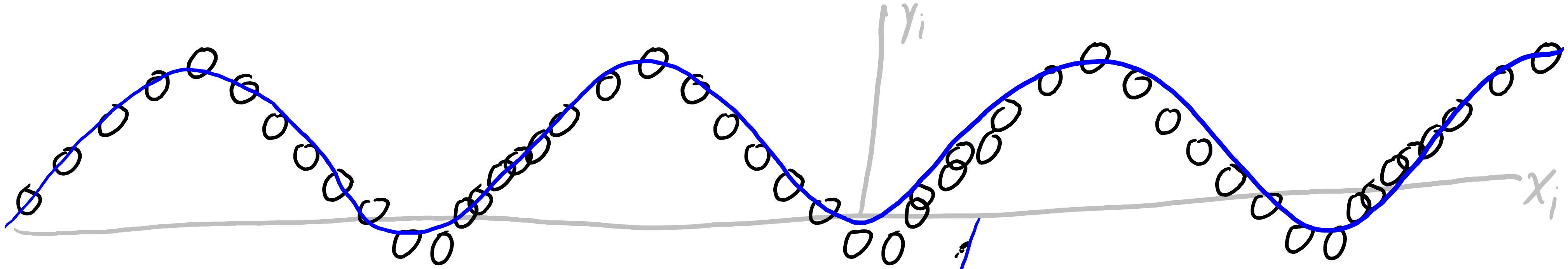
- As the polynomial degree increases, the training error goes down.



- But generalization gap goes up: we start overfitting with large ‘p’.
- Usual approach to selecting degree: validation or cross-validation.

Beyond Polynomial Transformations

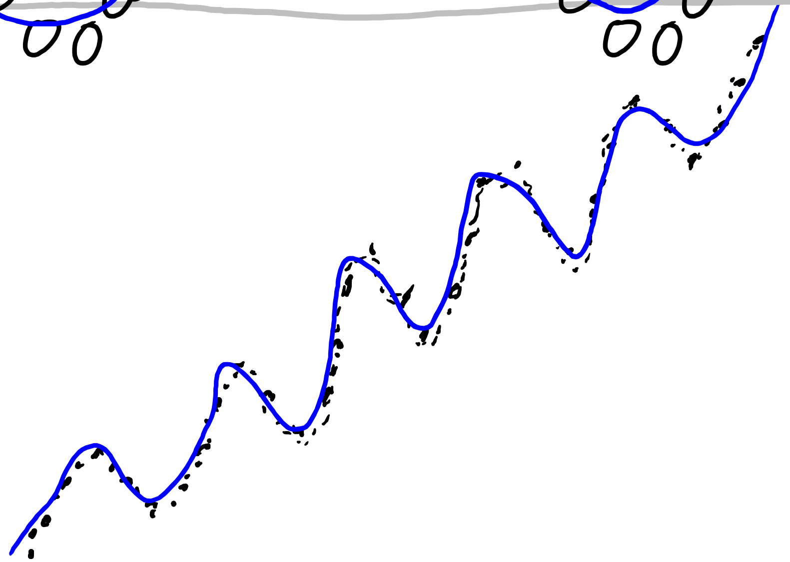
- Polynomials are not the only possible transformation:
  - Exponentials, logarithms, trigonometric functions, and so on.
  - The right non-linear transform will vastly improve performance.
    - Later we will see "deep learning" where you try to learn a transformation.



For periodic data we might use

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

$$\hat{y}_i = v^T z_i$$
$$= w_1 \sin(x_i)$$



You can have different types of bases

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

xkcd

Summary

- Matrix notation for expressing least squares problem.
- Normal equations: solution of least squares as a linear system.
  - Solve $(X^TX)w = (X^Ty)$.
- Solution might not be unique because of collinearity.
  - But any solution is optimal because of "convexity".
- Non-linear transforms:
  - Allow us to model non-linear relationships with linear models.

- Next time: how to do least squares with a million features.

Linear Least Squares: Expansion Step

Want 'w' that minimizes

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 = \underbrace{\frac{1}{2}\|Xw - y\|_2^2}_{\text{Let's expand then compute gradient.}} = \frac{1}{2}(Xw-y)^T(Xw-y)$$

$$= \frac{1}{2}((Xw)^T - y^T)(Xw - y)$$

$$= \frac{1}{2}(w^TX^T - y^T)(Xw - y)$$

$$= \frac{1}{2}(w^TX^T(Xw - y) - y^T(Xw - y))$$

$$= \frac{1}{2}(w^TX^TXw - w^TX^Ty - y^TXw + y^Ty)$$

$$= \underbrace{\frac{1}{2}w^TX^TXw - w^TX^Ty + \frac{1}{2}y^Ty}_{\text{Sanity check: all of these are scalars.}}$$

Rule:

$\|a\|^2 = a^Ta$

$(A+B^T) = (A^T + B^T)$

$(AB)^T = B^TA^T$

$(A+B)C = AC + BC$

$A(B+C) = AB + BC$

$a^TA\underbrace{b}_{\text{vector}} = \underbrace{b^T}_{\text{vector}}A^Ta$

Vector View of Least Squares

- We showed that least squares minimizes:

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

- The ½ and the squaring don't change solution, so equivalent to:

$$f(w) = \|Xw - y\|$$

- From this viewpoint, least square minimizes Euclidean distance between vector of labels 'y' and vector of predictions Xw.

Bonus Slide: Householder(-ish) Notation

- Househoulder notation: set of (fairly-logical) conventions for math.

Use greek letters for scalars: $\alpha = 1$, $\beta = 3.5$, $\gamma = \pi$

Use first/last lowercase letters for vectors: $w = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$, $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $a = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, $b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

Assumed to be column-vectors.

Use first/last uppercase letters for matrices: X, Y, W, A, B

Indices use i, j, k.

Sizes use m, n, d, p, and k

hopefully meaning of 'k' is obvious from context

Sets use S, T, U, V

Functions use f, g, and h.

When I write x_i I mean "grab row 'i' of X and make a column-vector with its values."

Bonus Slide: Householder(-ish) Notation

- Househoulder notation: set of (fairly-logical) conventions for math:

Our ultimate least squares notation:

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

But if we agree on notation we can quickly understand:

$$g(x) = \frac{1}{2}\|Ax - b\|^2$$

If we use random notation we get things like:

$$H(\beta) = \frac{1}{2}\|R\beta - P_n\|^2$$

Is this the same model?

When does least squares have a unique solution?

- We said that least squares solution is not unique if we have repeated columns.
- But there are other ways it could be non-unique:
  - One column is a scaled version of another column.
  - One column could be the sum of 2 other columns.
  - One column could be three times one column minus four times another.

- Least squares solution is unique if and only if all columns of X are "linearly independent".
  - No column can be written as a "linear combination" of the others.
  - Many equivalent conditions (see Strang's linear algebra book):
    - X has "full column rank", X^TX is invertible, X^TX has non-zero eigenvalues, $\det(X^TX) > 0$.
  - Note that we cannot have independent columns if $d > n$.