CPSC 340: Machine Learning and Data Mining

Nonlinear Regression

Fall 2018

Last Time: Linear Regression

- We discussed linear models:

$$y_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= \sum_{j=1}^{d} w_j x_{ij} = w^T x_i$$

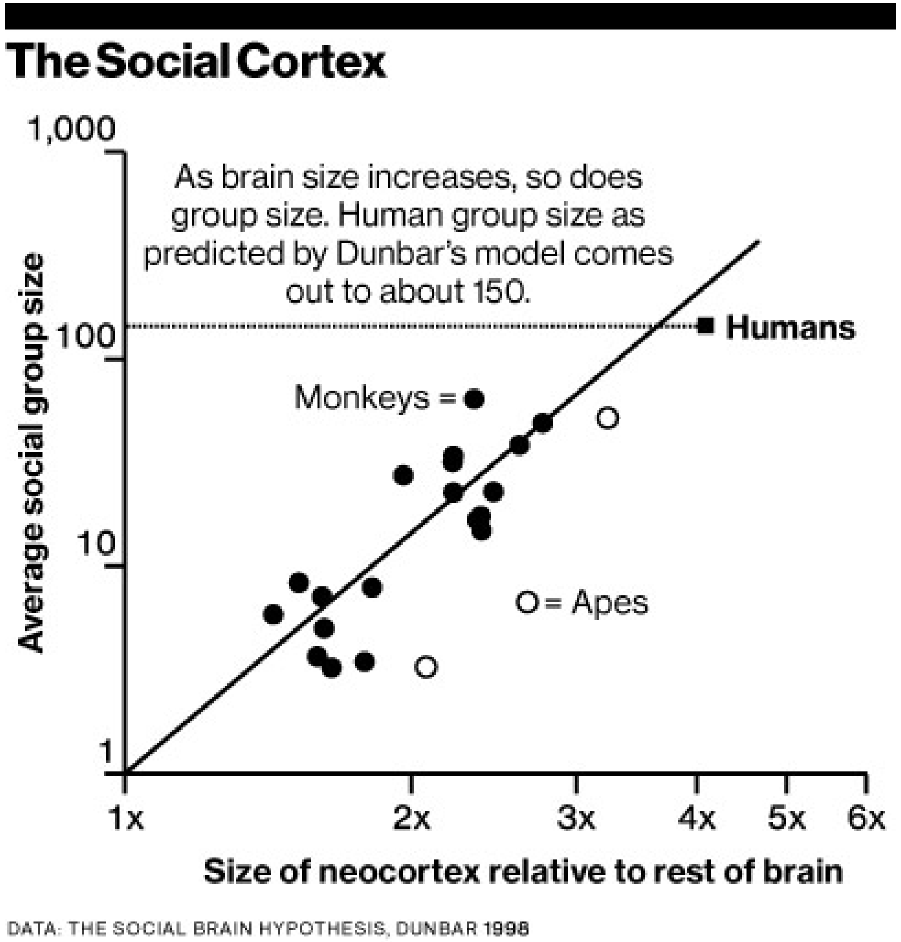
- "Multiply feature x_{ij} by weight w_j, add them to get y_i".
- We discussed squared error function:

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

Predicted value

True value

- 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

Last Time: Supervised Learning Notation

- We're treating 'w', 'y', $\hat{y}_i$, and each x_i as column-vectors:

$$
\underset{d \times 1}{w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}}
\qquad
\underset{n \times 1}{y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}
\qquad
\underset{d \times 1}{x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}}
\qquad
\underset{t \times 1}{\hat{y}_i = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_t \end{bmatrix}}
$$

- So feature matrix 'X' actually has x_i transposed as rows:

$$
X = \begin{bmatrix} — & x_1^T & — \\ — & x_2^T & — \\ & \vdots & \\ — & x_n^T & — \end{bmatrix}
$$

Last Time: Matrix Notation

- We can write vector of predictions $\hat{y}_i$ as a matrix-vector product:

$$\hat{y} = Xw = \begin{bmatrix} w^T x_1 \\ w^T x_2 \\ \vdots \\ w^T x_n \end{bmatrix}$$

- And we can write linear least squares in matrix notation as:

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

- We'll use this notation to derive d-dimensional least squares 'w'…

Digression: Matrix Algebra Review

- Quick review of linear algebra operations we'll use:
  - If 'a' and 'b' be vectors, and 'A' and 'B' be 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 (see post-lecture slide for steps):

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

$$= \frac{1}{2}w^TAw + w^Tb + c$$

These are scalars so dimensions match.

- 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)

$\nabla[w^Tb] = b$

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

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

Linear and Quadratic Gradients

- We've written as a d-dimensional quadratic:

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

$$= \frac{1}{2}w^TAw + w^Tb + c$$

- 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

- 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:

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

- This is a set of 'd' linear equations called the "normal equations".
  - This a linear system like "Ax = b" from Math 152.
    - You can use Gaussian elimination to solve for 'w'.
  - In Python, you solve linear systems in 1 line using numpy.linalg.solve.

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)

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

$w = \dfrac{X^T y}{X^T X}$ (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 a sum of '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 can might so big we can't store X^TX.
    - Or you can't afford the $O(nd^2 + d^3)$ cost.
  - It might predict outside range of y_i values.
  - 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 isn't solution unique?
  - Imagine having two features that are identical for all examples.
  - This is special case of features being "collinear"
    - One feature is a linear function of the others.
  - I can increase weight on one feature, and decrease it on the other, without changing predictions.

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

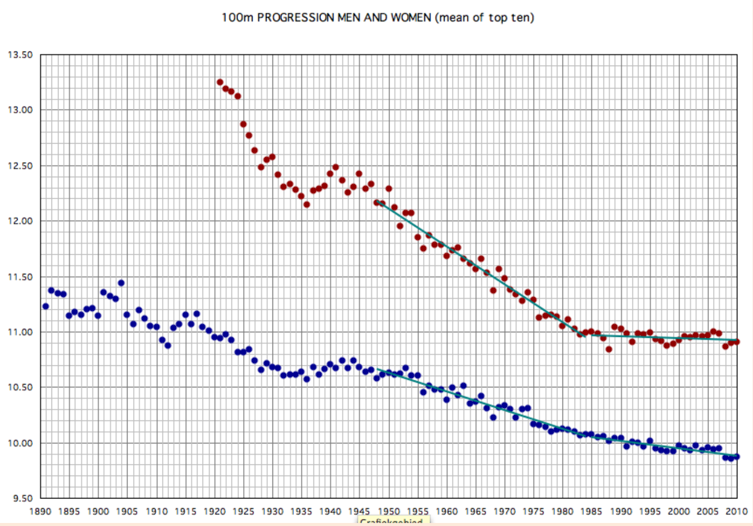
  - Thus, if (w_1,w_2) is a solution then $(w_1+w_2, 0)$ is a solution.

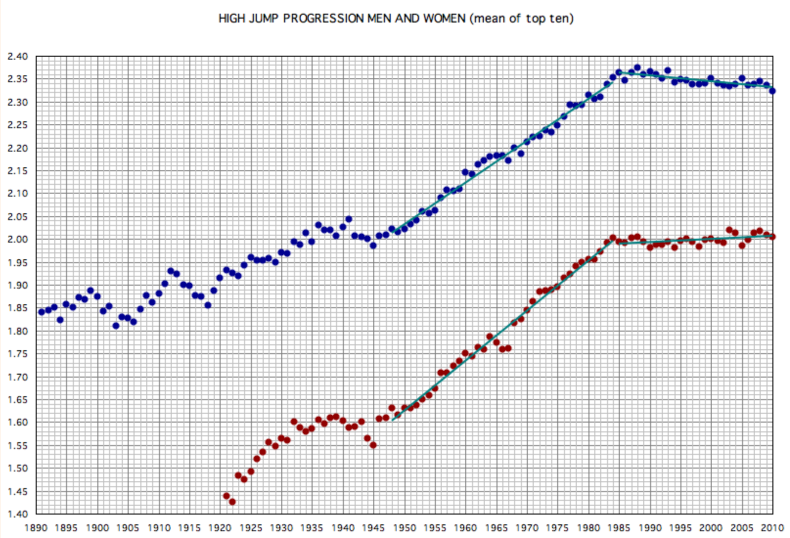
- But, any 'w' where $\nabla f(w) = 0$ is a global optimum.
  - This is due to convexity of 'f', which we'll discuss later.

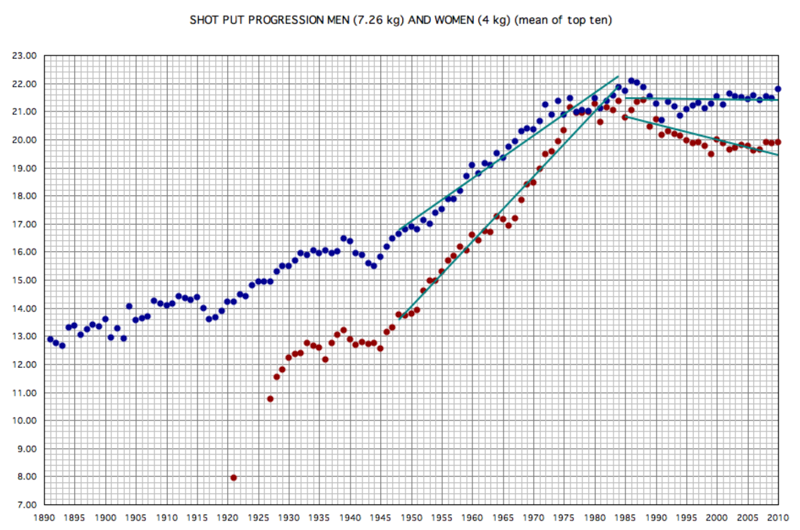
(pause)

Motivation: Non-Linear Progressions in Athletics

- Are top athletes going faster, higher, and farther?







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

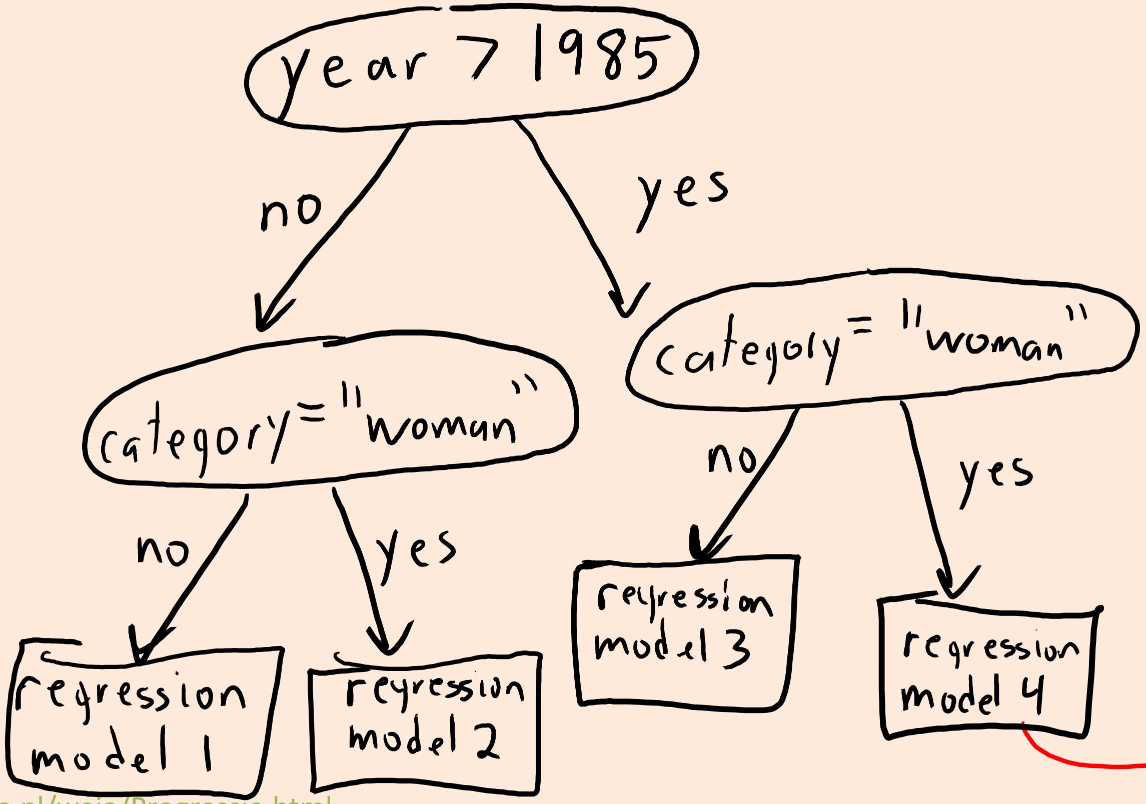
Adapting Counting/Distance-Based Methods

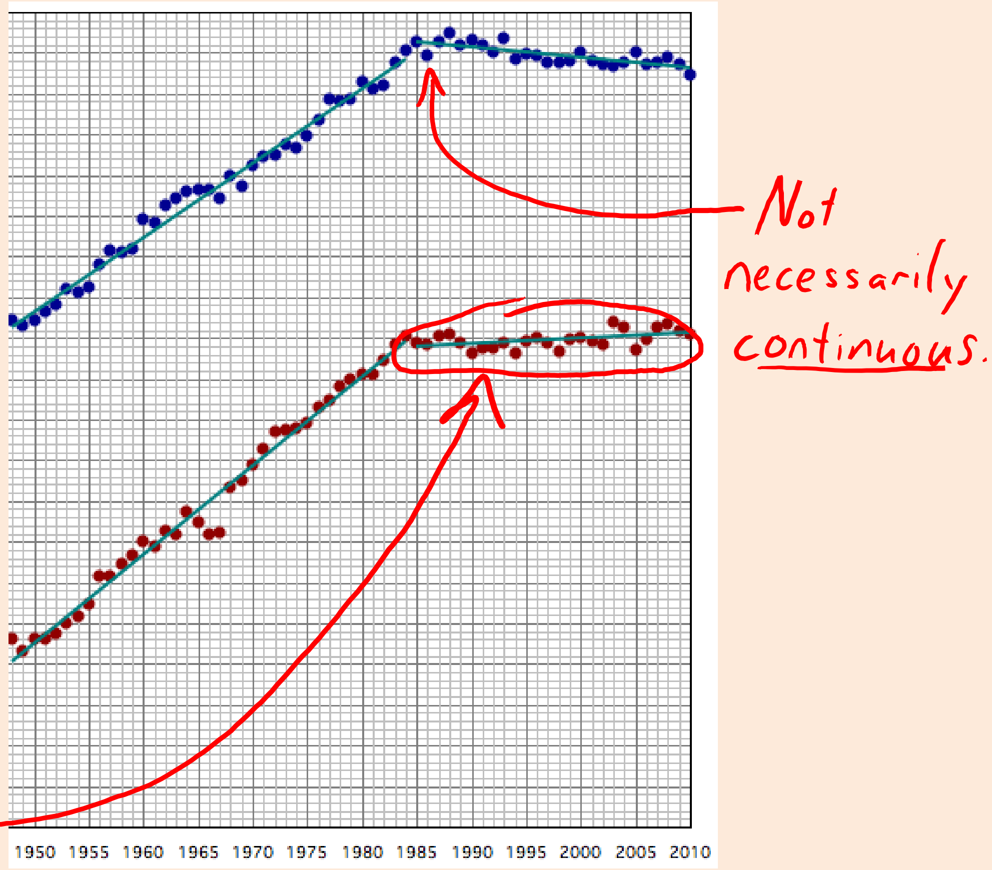
- We can adapt our classification methods to perform regression:

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

Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.

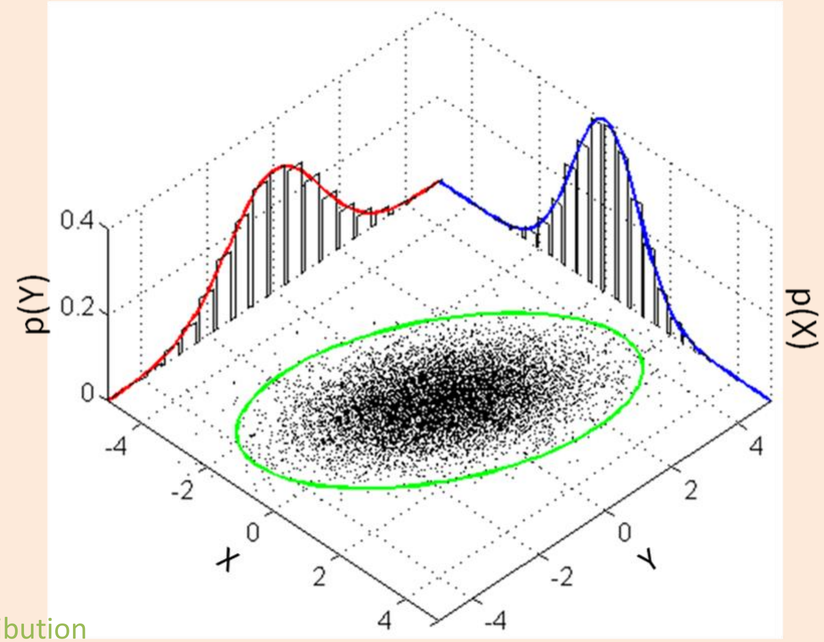




http://www.at-a-lanta.nl/weia/Progressie.html

Adapting Counting/Distance-Based Methods

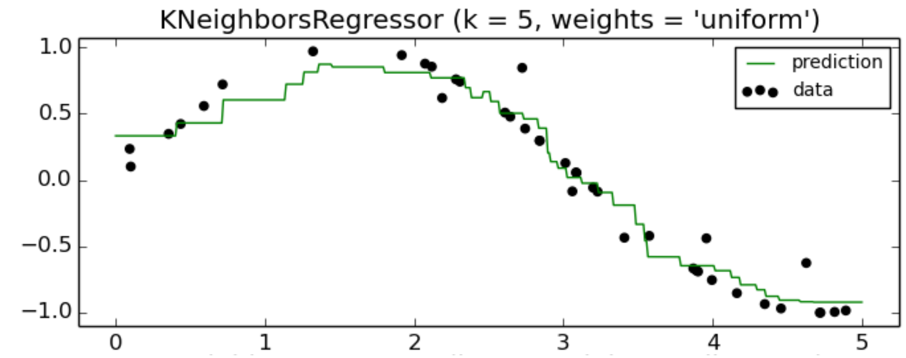
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
    - CPSC 540.



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

Adapting Counting/Distance-Based Methods

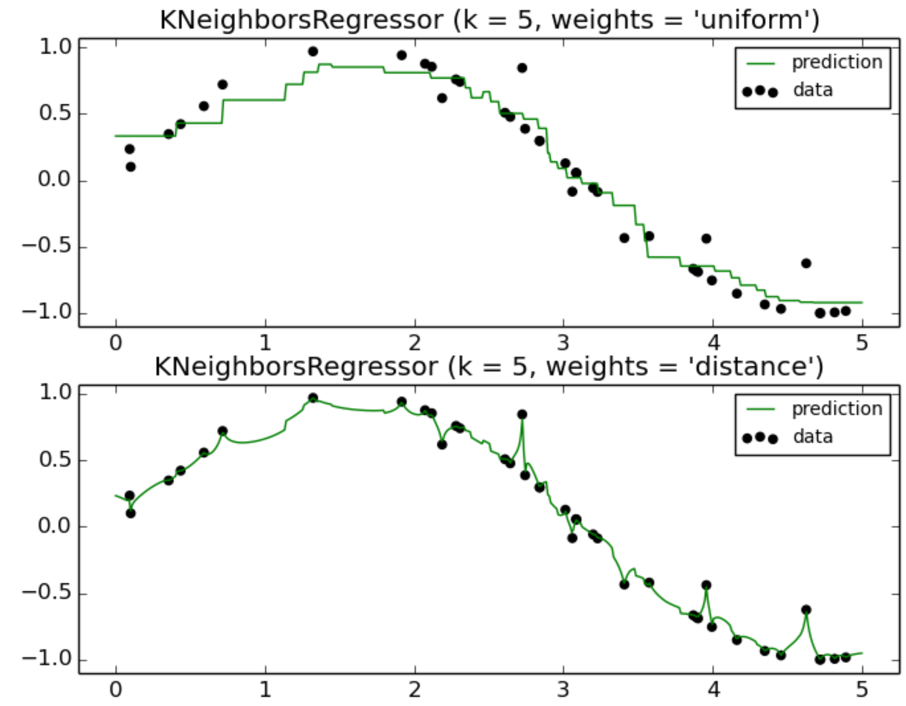
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression:
      - Find 'k' nearest 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

- We can adapt our classification methods to perform 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}.

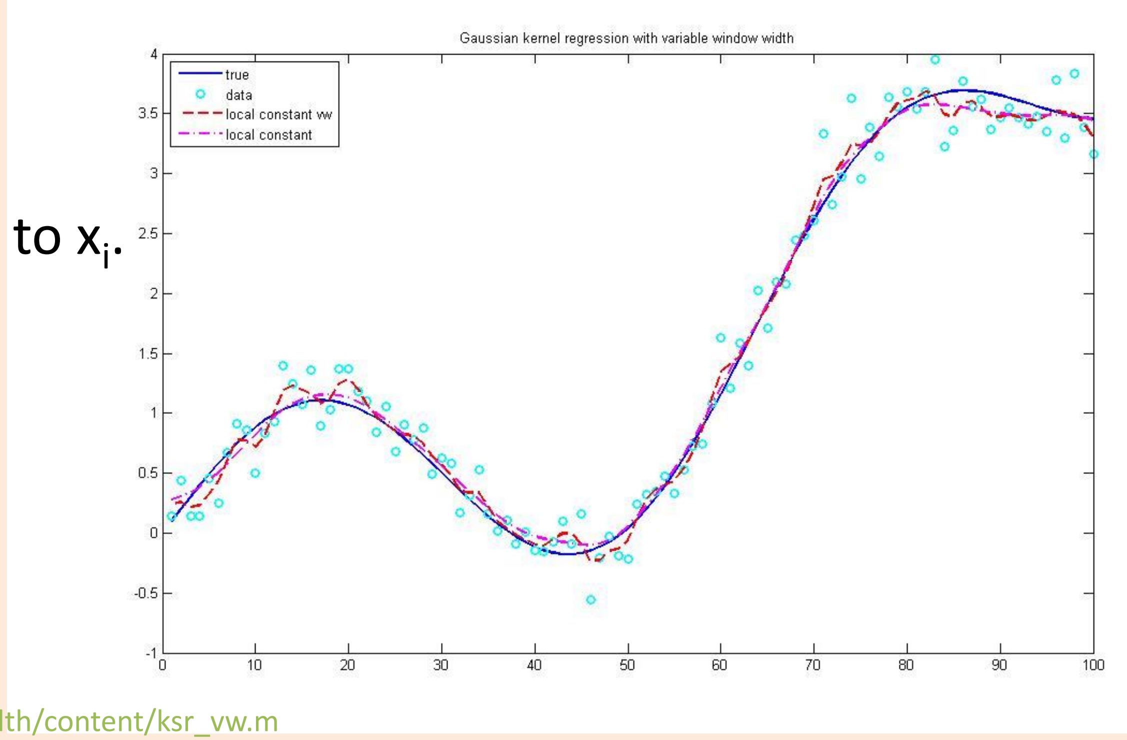


http://scikit-learn.org/stable/modules/neighbors.html

Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform 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-Waston': weight *all* y_i by distance to x_i.

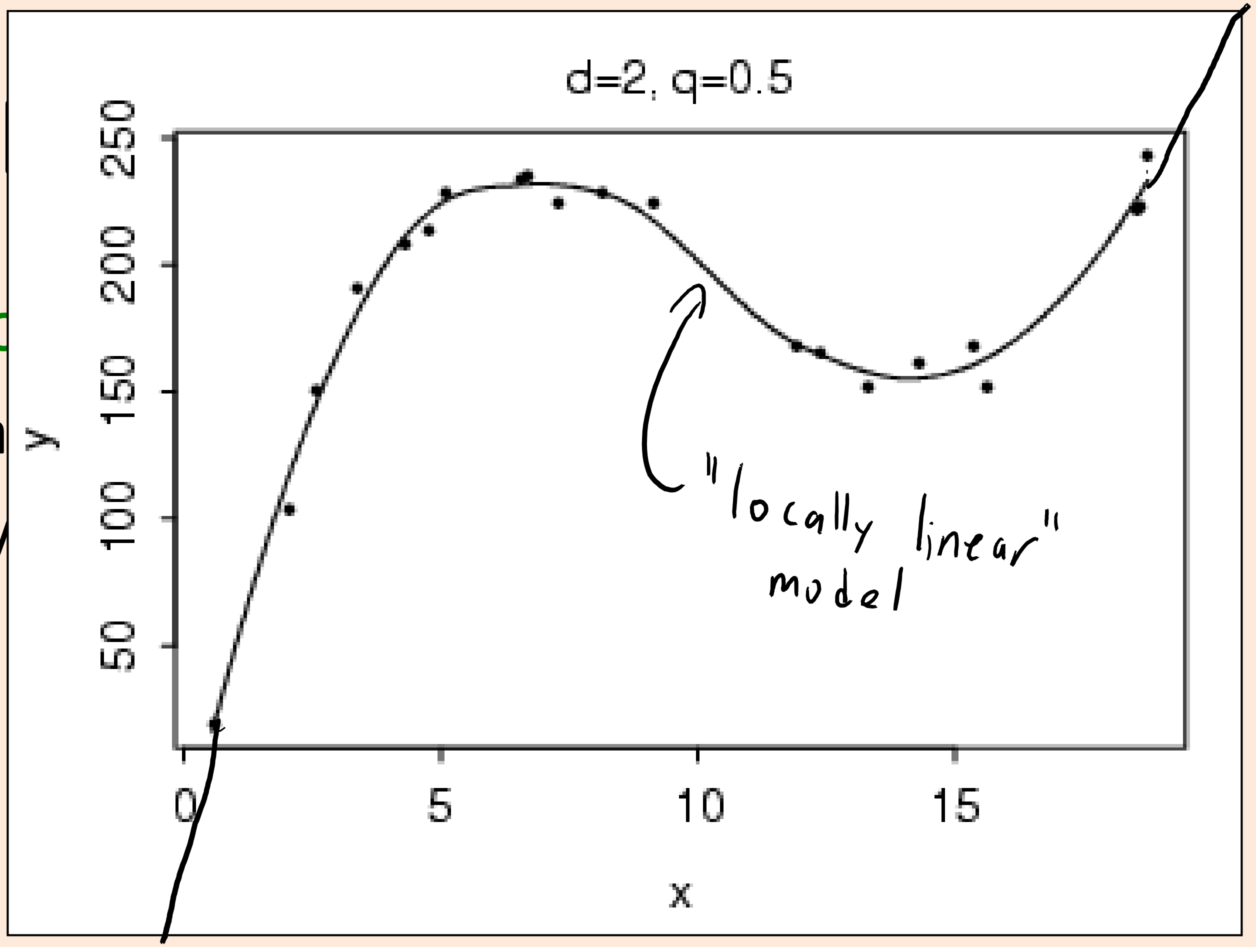
$$\hat{y}_i = \frac{\sum_{j=1}^{n} v_{ij} y_j}{\sum_{j=1}^{n} v_{ij}}$$



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

Adapting Counting/

- We can adapt our classificatio
  - Regression tree: tree with mea
  - Probabilistic models: fit $p(x_i \mid y$
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - 'Nadaraya-Waston': 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.)



Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform 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-Waston': 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 across regression models.

Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression.

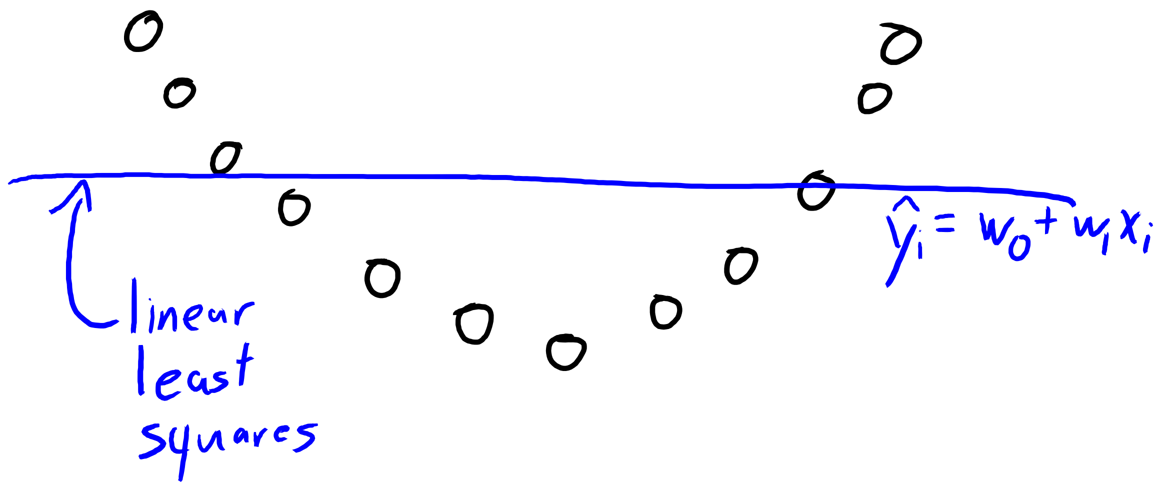
- Applications:
  - Regression forests for fluid simulation:
    - https://www.youtube.com/watch?v=kGB7Wd9CudA
  - KNN for image completion:
    - http://graphics.cs.cmu.edu/projects/scene-completion
    - Combined with “graph cuts” and “Poisson blending”.
  - KNN regression for “voice photoshop”:
    - https://www.youtube.com/watch?v=I3l4XLZ59iw
    - Combined with “dynamic time warping” and “Poisson blending”.

- But we’ll 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

Motivation: Limitations of Linear Models

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



- Can we use least square to fit non-linear models?

Non-Linear Feature Transforms

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

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

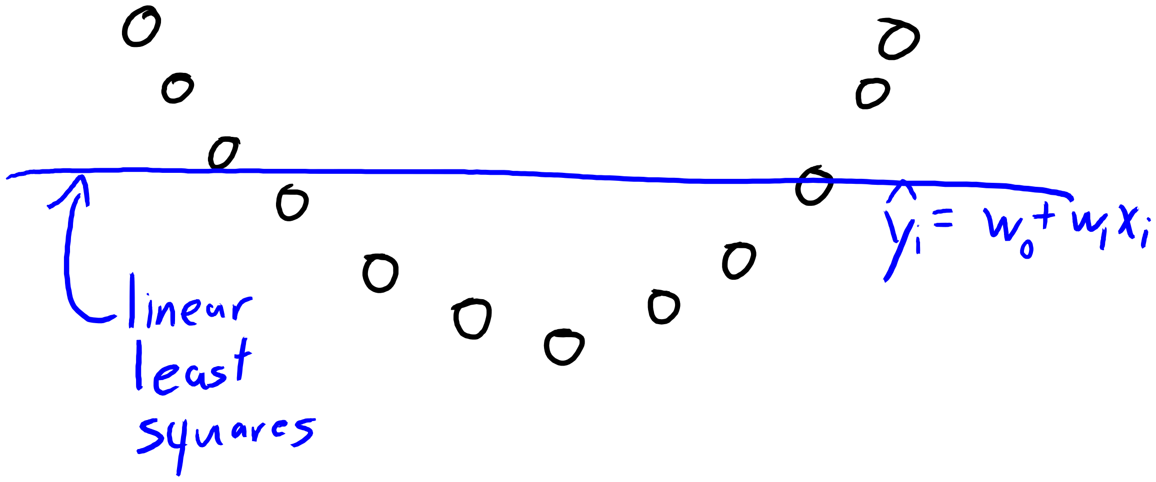
- You can do this by changing the features (change of basis):

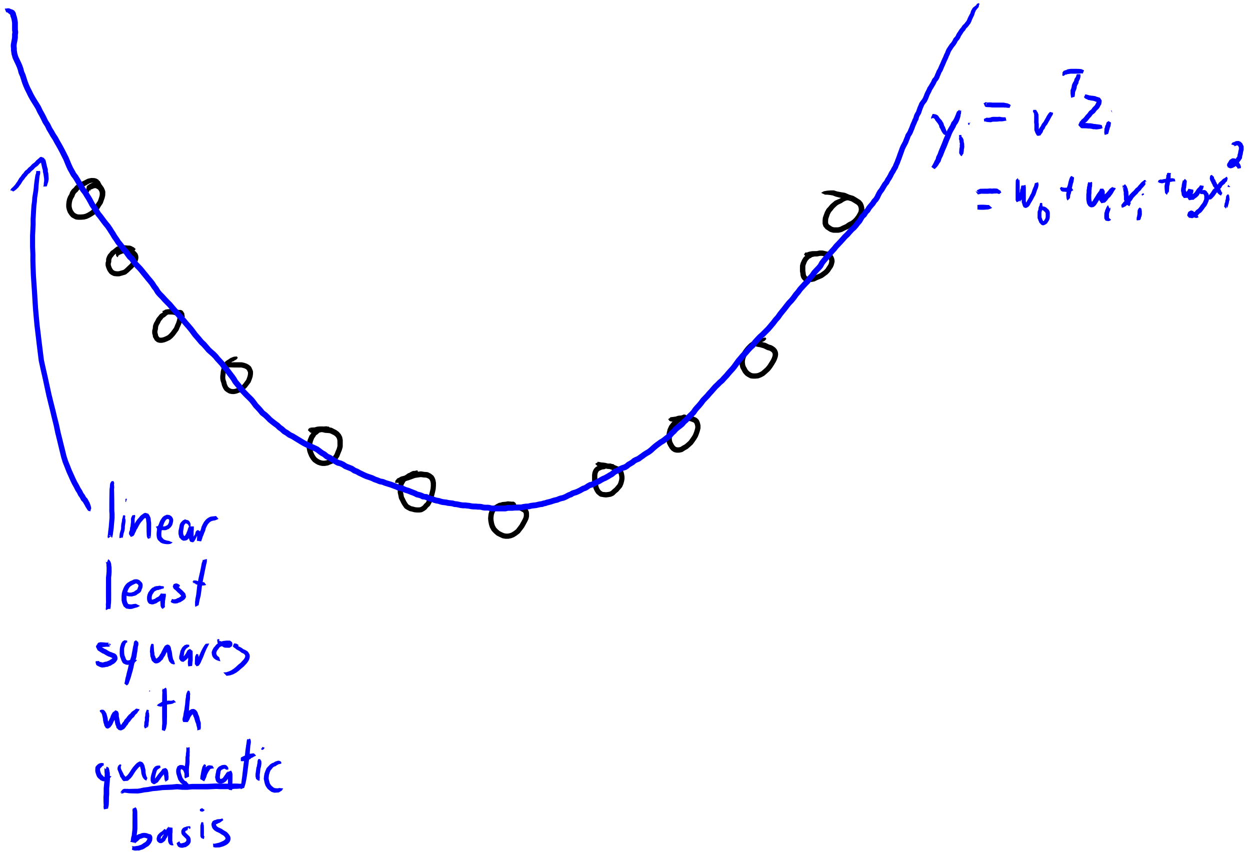
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad Z = \begin{array}{ccc} \left[\begin{matrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{matrix}\right] \\ \begin{matrix} \text{y-int} & \quad x \quad & x^2 \end{matrix} \end{array}$$

- 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 $y = \tilde{Z}v$

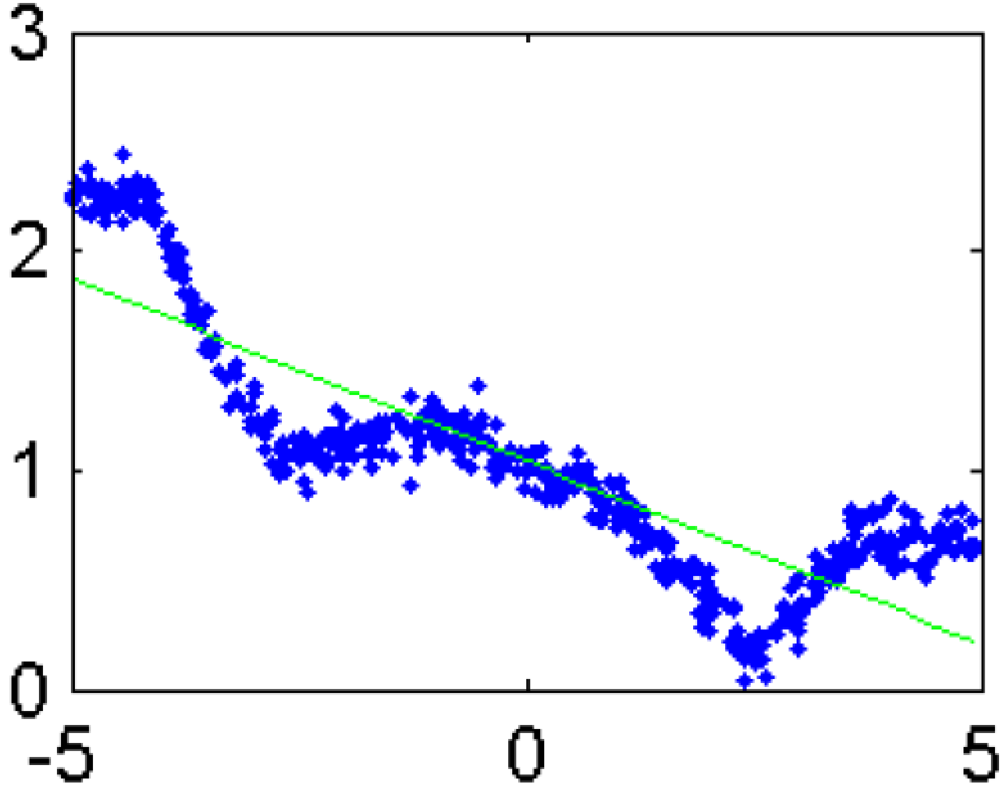
General Polynomial Features (d=1)

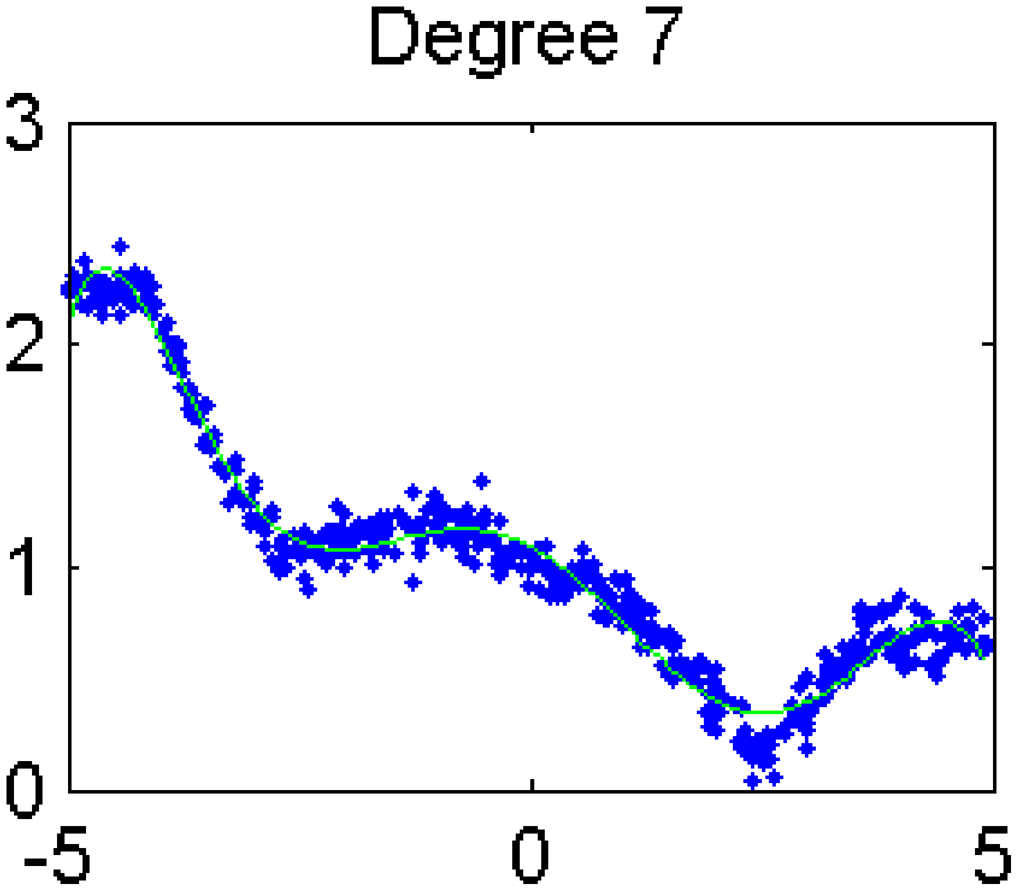
- We can have a polynomial of degree ‘p’ by using these features:

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

- There are polynomial basis functions that are numerically nicer:
  - E.g., 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 would 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”.

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:

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

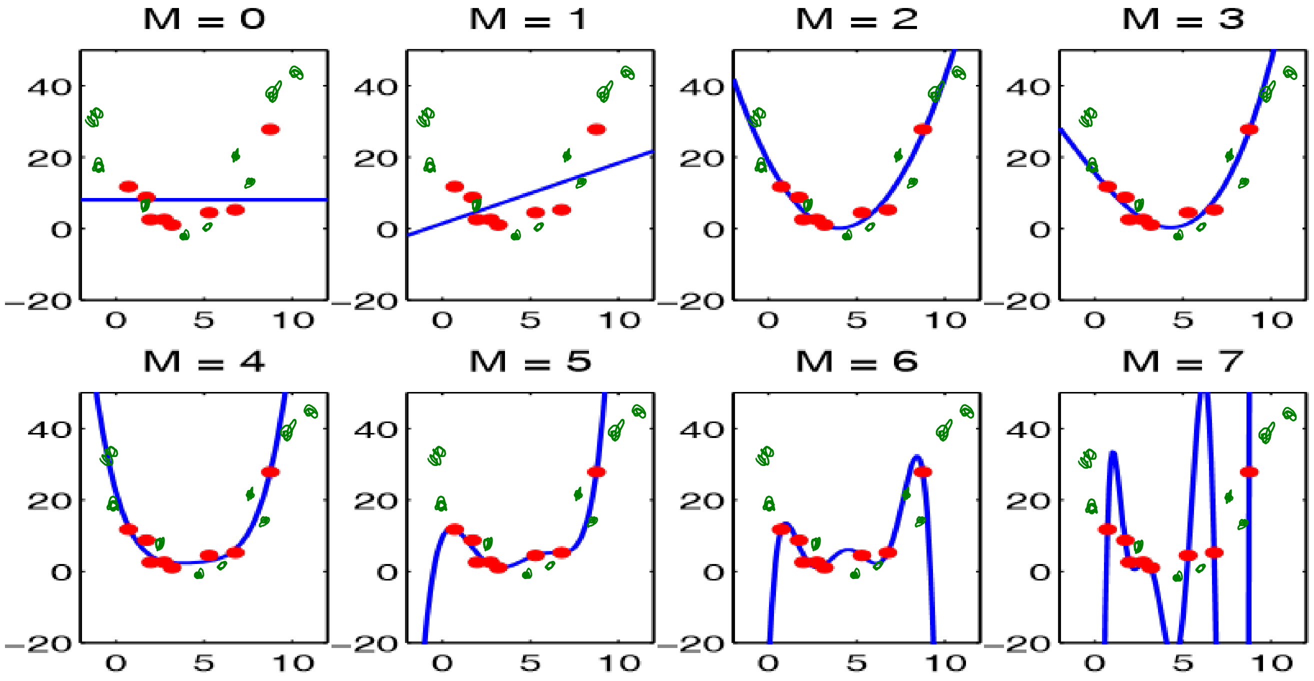
- Linear regression with nonlinear feature transforms:
  - We use 'Z' as our "n by k" data matrix, and 'v' as our parameters.
  - We can find k-dimensional 'v' by minimizing the squared error:

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

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

Degree of Polynomial and Fundamental Trade-Off

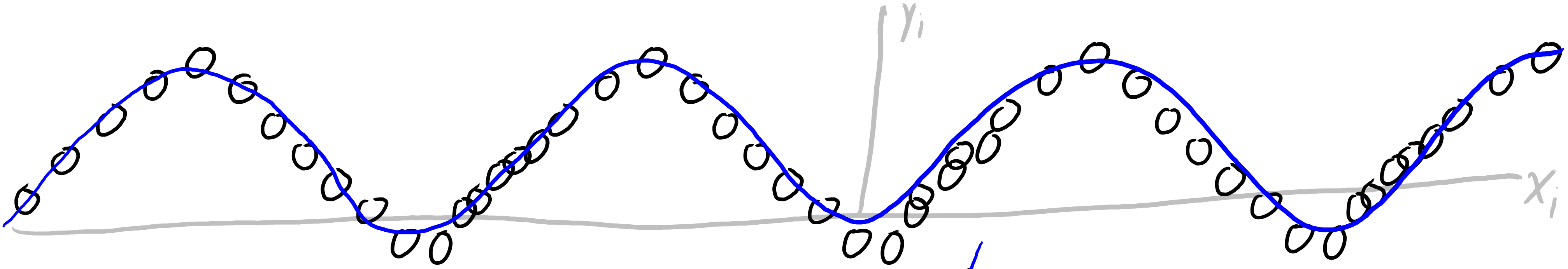
- As the polynomial degree increases, the training error goes down.



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

Beyond Polynomial Transformations

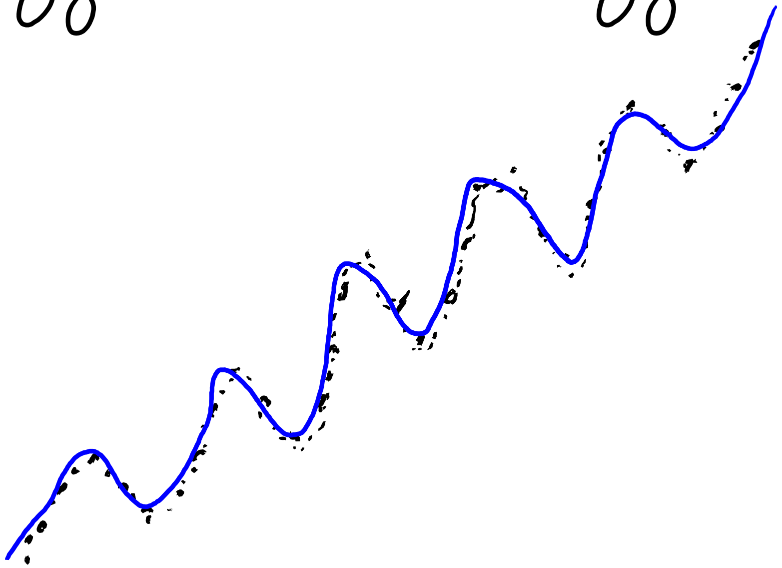
- Polynomials are not the only possible transformation:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right non-linear transform will vastly improve performance.



For periodic data we might use

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

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



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

- 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".
- Tree/probabilistic/non-parametric/ensemble regression methods.
- 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

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 = \frac{1}{2}\|Xw - y\|_2^2 = \frac{1}{2}(Xw-y)^T(Xw-y)$$

Let's expand then compute gradient.

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

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

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

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

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

Rule:

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

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

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

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

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

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

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 $d > n$.