

CPSC 340: Machine Learning and Data Mining

Nonlinear Regression

Fall 2021

Admin

- a2 late deadline passed
- a3 will be posted later today, or tomorrow at the latest
- Midterm
 - Thu Oct 21
 - 6:00-7:30pm
 - Fully remote
 - Open book
 - No communication with others allowed
 - Will be on Canvas

Last Time: Linear Regression

- We discussed **linear models**:

$$\begin{aligned}y_i &= w_1 x_{i1} + w_2 x_{i2} + \dots + w_d x_{id} \\&= \sum_{j=1}^d w_j x_{ij} = w^\top x_i\end{aligned}$$

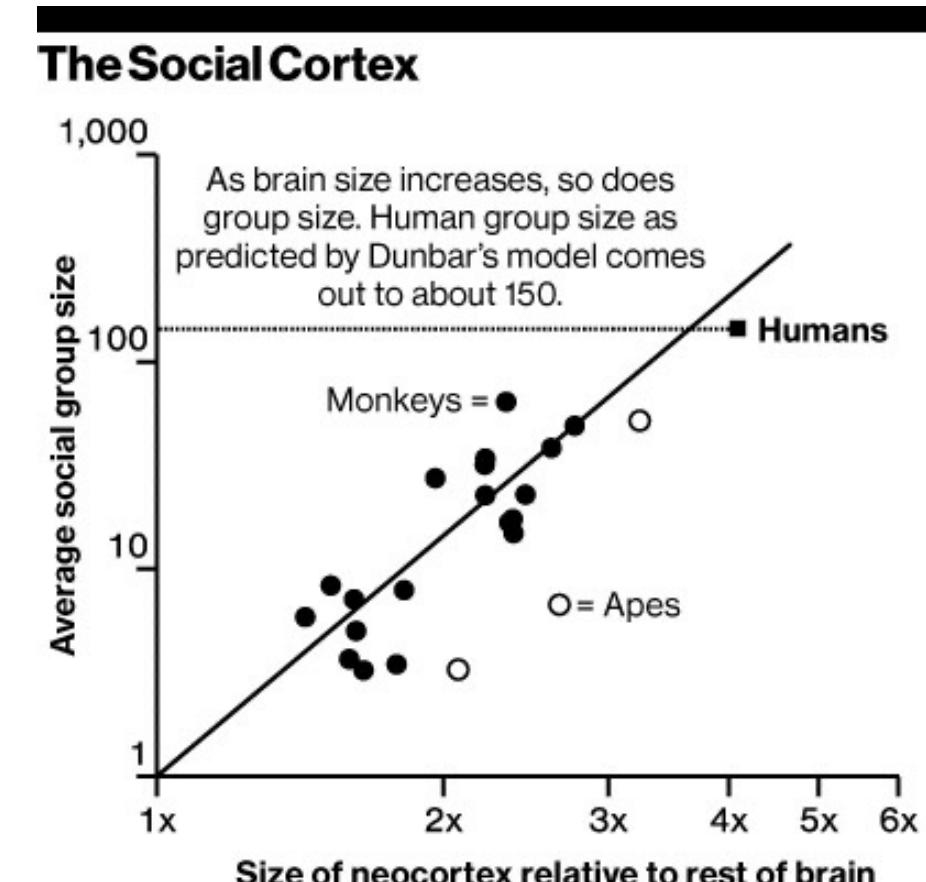
- “Multiply feature x_{ij} by weight w_j , add them to get y_i ”.
- We discussed **squared error** function:

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

Predicted value \swarrow \searrow True value

- Interactive demo:

- <http://setosa.io/ev/ordinary-least-squares-regression>



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

Matrix/Norm Notation (MEMORIZE/STUDY THIS)

- To solve the d-dimensional least squares, we use matrix notation:
 - 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’.

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

Matrix/Norm Notation (MEMORIZE/STUDY THIS)

- To solve the d-dimensional least squares, we use **matrix notation**:
 - Our prediction for example 'i' is given by the **scalar** $w^T x_i$.
 - Our predictions for all 'i' (n times 1 vector) is the **matrix-vector product** Xw .

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

Also, because $w^T x_i$ is a scalar,

We have $w^T x_i = x_i^T w$.

(e.g., $[5]^T = [5]$)

$$Xw = \begin{bmatrix} -x_1^T- \\ -x_2^T- \\ \vdots \\ -x_n^T- \end{bmatrix} \begin{bmatrix} | \\ w \\ | \end{bmatrix} = \begin{bmatrix} x_1^T w \\ x_2^T w \\ \vdots \\ x_n^T w \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \hat{y}$$

Prediction for
example 'i' in row 'i'

Matrix/Norm Notation (MEMORIZE/STUDY THIS)

- To solve the d-dimensional least squares, we use **matrix notation**:
 - Our prediction for example 'i' is given by the **scalar** $w^T x_i$.
 - Our predictions for all 'i' (n times 1 vector) is the **matrix-vector product** Xw .
 - Residual vector 'r' gives difference between y_i and predictions (n times 1).
 - Least squares can be written as the squared L2-norm of the residual.

$$r = \hat{y} - y = Xw - y = \begin{bmatrix} w^T y_1 \\ w^T y_2 \\ \vdots \\ w^T y_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} w^T y_1 - y_1 \\ w^T y_2 - y_2 \\ \vdots \\ w^T y_n - y_n \end{bmatrix}$$

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

$= \sum_{i=1}^n r_i r_i$

$= r^T r$

$= \|r\|^2 = \|Xw - y\|^2$

r_2 is difference for example 2.

Back to Deriving Least Squares for $d > 2$...

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

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

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

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

- We'll use this notation to derive d-dimensional least squares ' w '.
 - By setting the gradient $\nabla f(w)$ equal to the zero vector and solving for ' w '.

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:

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

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

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

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

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

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

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):

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

These are scalars so dimensions match.

- How do we compute gradient?

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

$$\begin{aligned} f(w) &= \frac{1}{2} w^T a w + w^T b + c \\ &= \frac{1}{2} a w^2 + w^T b + c \end{aligned}$$

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

Here are the generalizations to ' d ' dimensions:

$$\nabla[c] = 0 \quad (\text{zero vector})$$

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

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

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

Linear and Quadratic Gradients

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

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

$$= \frac{1}{2} w^T A w + w^T b + c$$

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

- Using definitions of 'A' and 'b': $= X^T X w - X^T y$
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 is 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:

$$w = (X^T X)^{-1} (X^T y) \quad (\text{only true if } X^T X \text{ is invertible})$$

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

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

Least Squares Cost

- Cost of solving “normal equations” $X^T X w = X^T y$?
- Forming $X^T y$ vector costs $O(nd)$.
 - It has ‘d’ elements, and each is an inner product between ‘n’ numbers.
- Forming matrix $X^T X$ costs $O(nd^2)$.
 - It has d^2 elements, and each is an inner product between ‘n’ numbers.
- Solving a $d \times d$ 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 be so big we **can't store $X^T X$** .
 - 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 .

$\rightarrow X$ is $n \times d$
so X^T is $d \times n$
and $X^T X$ 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.
 - I can increase weight on one feature, and decrease it on the other, **without changing predictions**.

$$\hat{y}_i = w_1 x_{i,1} + \underbrace{w_2 x_{i,1}}_{\text{copy}} = (w_1 + w_2) x_{i,1} + 0 x_{i,1}$$

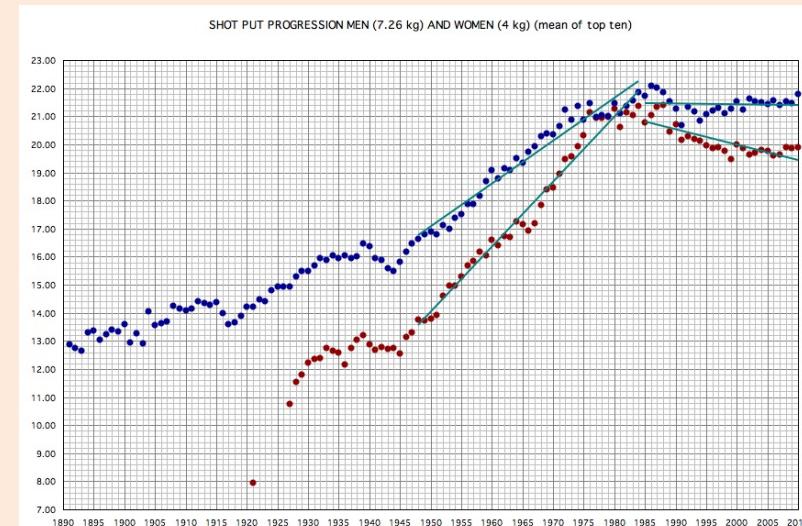
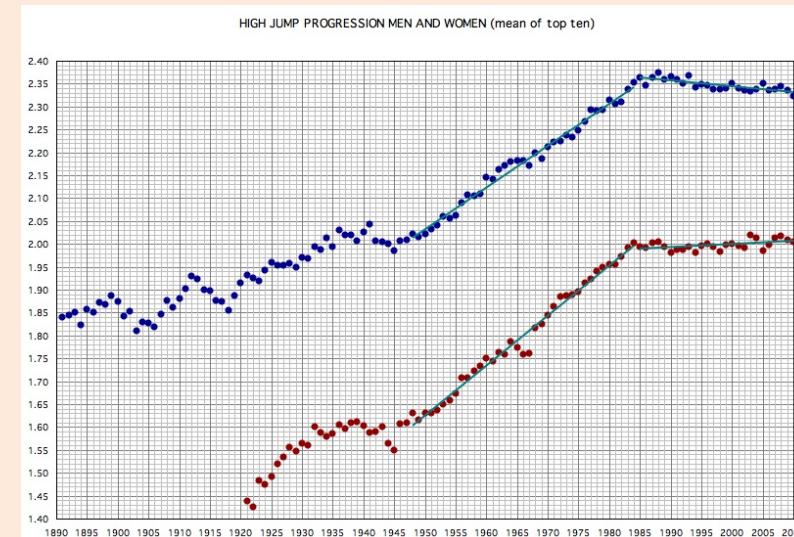
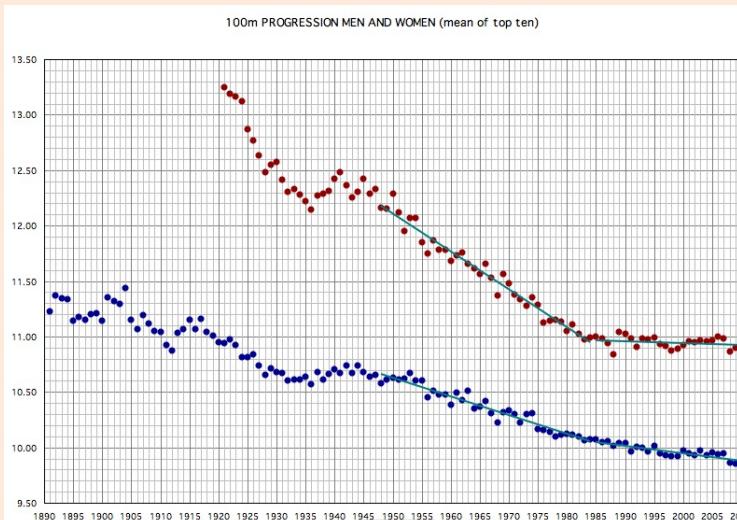
- Thus, if (w_1, w_2) is a solution then $(w_1 + w_2, 0)$ is another solution.
- This is special case of features being “**collinear**”:
 - One feature is a linear function of the others.
- But, any ‘w’ where $\nabla f(w) = 0$ is a global minimizer of ‘f’.
 - This is due to **convexity** of ‘f’, which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

(pause)

bonus!

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

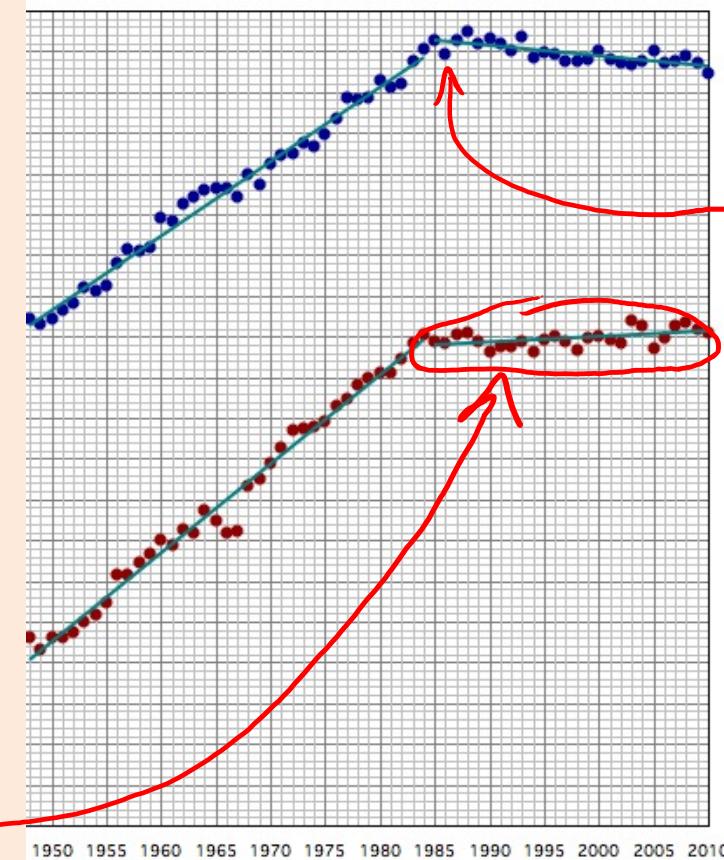
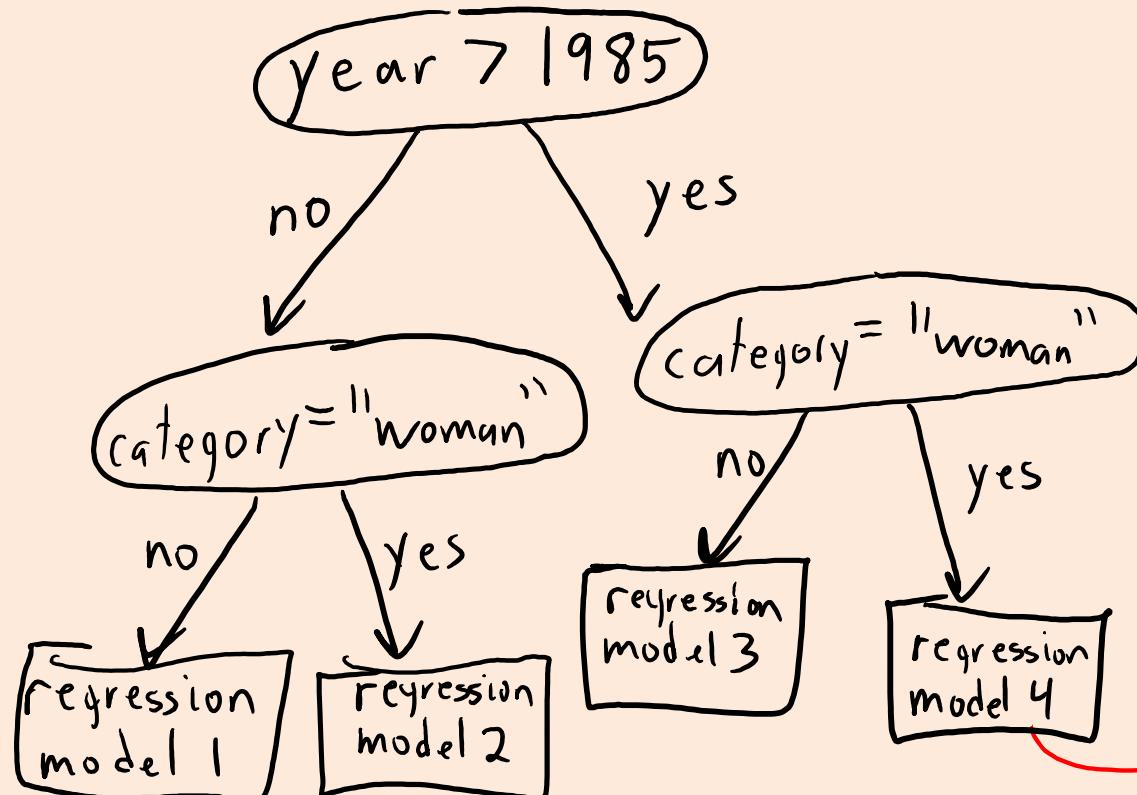
bonus!

- We can adapt our classification methods to perform regression:

bonus!

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.

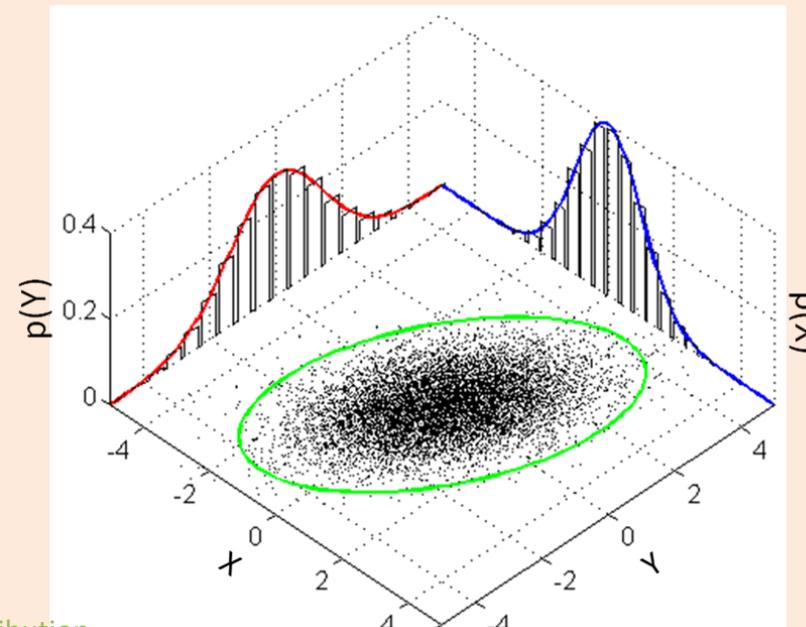


Not
necessarily
continuous.

Adapting Counting/Distance-Based Methods

bonus!

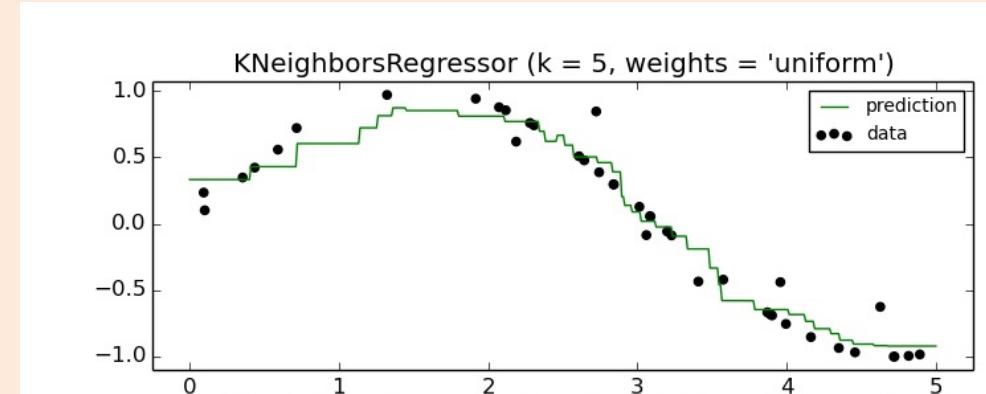
- We can adapt our classification methods to perform regression:
 - Regression tree: tree with mean value or linear regression at leaves.
 - Probabilistic models: fit $p(x_i | y_i)$ and $p(y_i)$ with Gaussian or other model.
 - Take CPSC 440.



bonus!

Adapting Counting/Distance-Based Methods

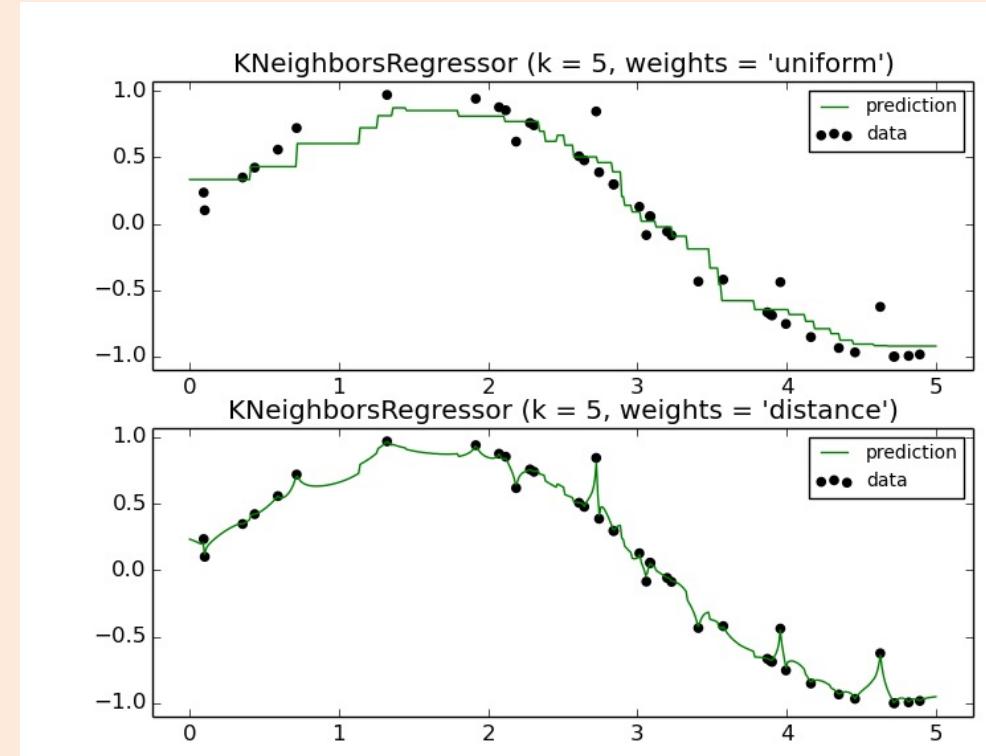
- We can adapt our classification methods to perform regression:
 - Regression tree: tree with mean value or linear regression at leaves.
 - Probabilistic models: fit $p(x_i | y_i)$ and $p(y_i)$ with Gaussian or other model.
 - Non-parametric models:
 - KNN regression:
 - Find 'k' nearest neighbours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



bonus!

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 $p(x_i | y_i)$ 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} .

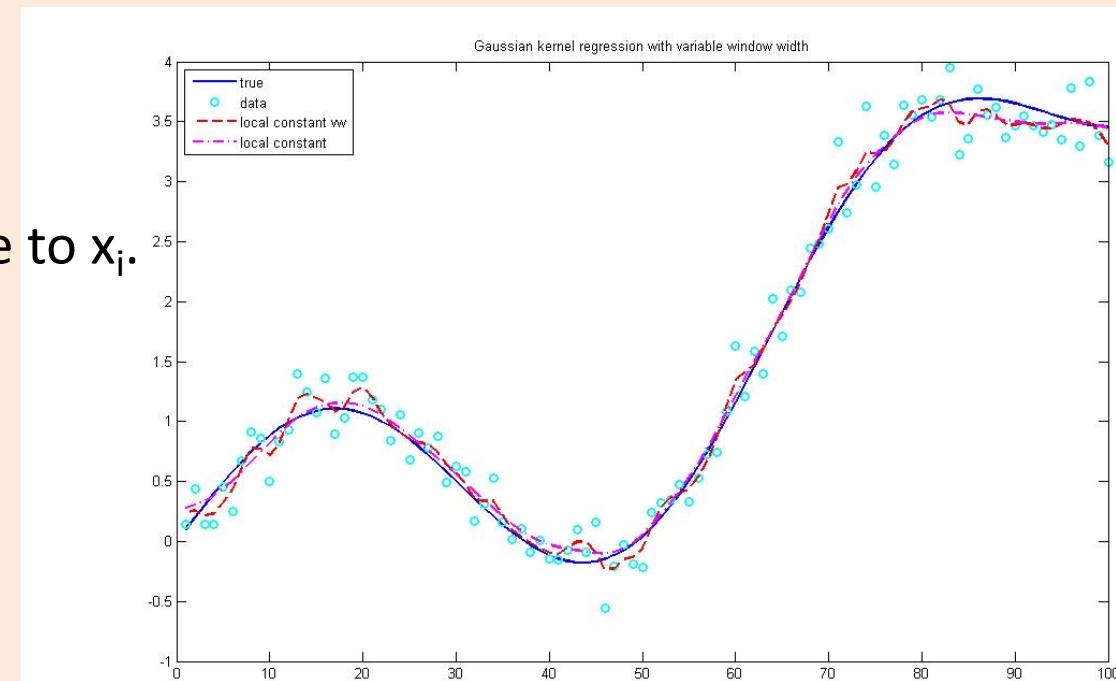


bonus!

Adapting Counting/Distance-Based Methods

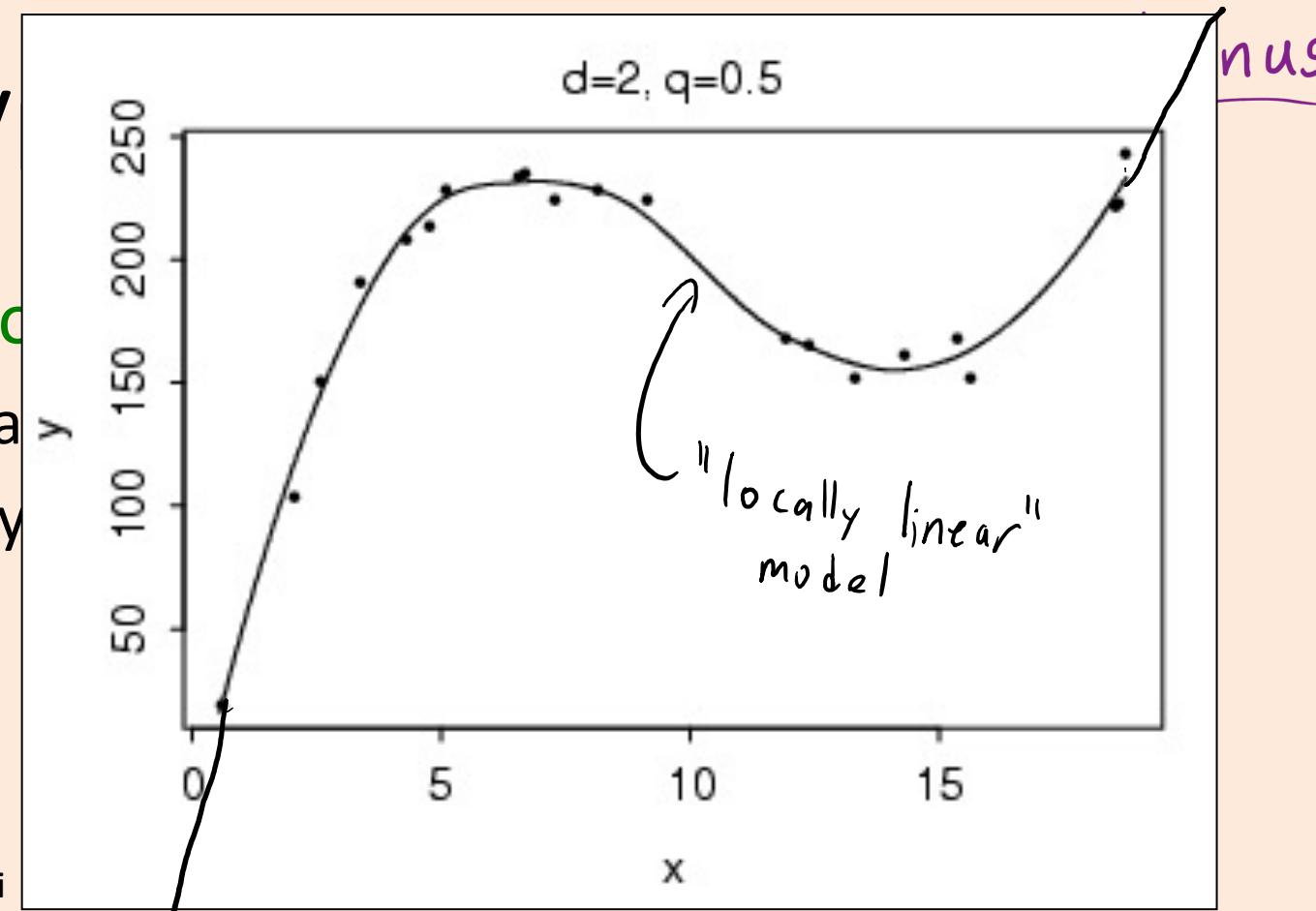
- We can adapt our classification methods to perform regression:
 - Regression tree: tree with mean value or linear regression at leaves.
 - Probabilistic models: fit $p(x_i | y_i)$ and $p(y_i)$ with Gaussian or other model.
 - Non-parametric models:
 - KNN regression.
 - Could be weighted by distance.
 - ‘Nadaraya-Waston’: weight *all* y_j by distance to x_i .

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



Adapting Counting/

- We can adapt our classification:
 - Regression tree: tree with mean
 - **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

bonus!

- 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 predictions across regression models.

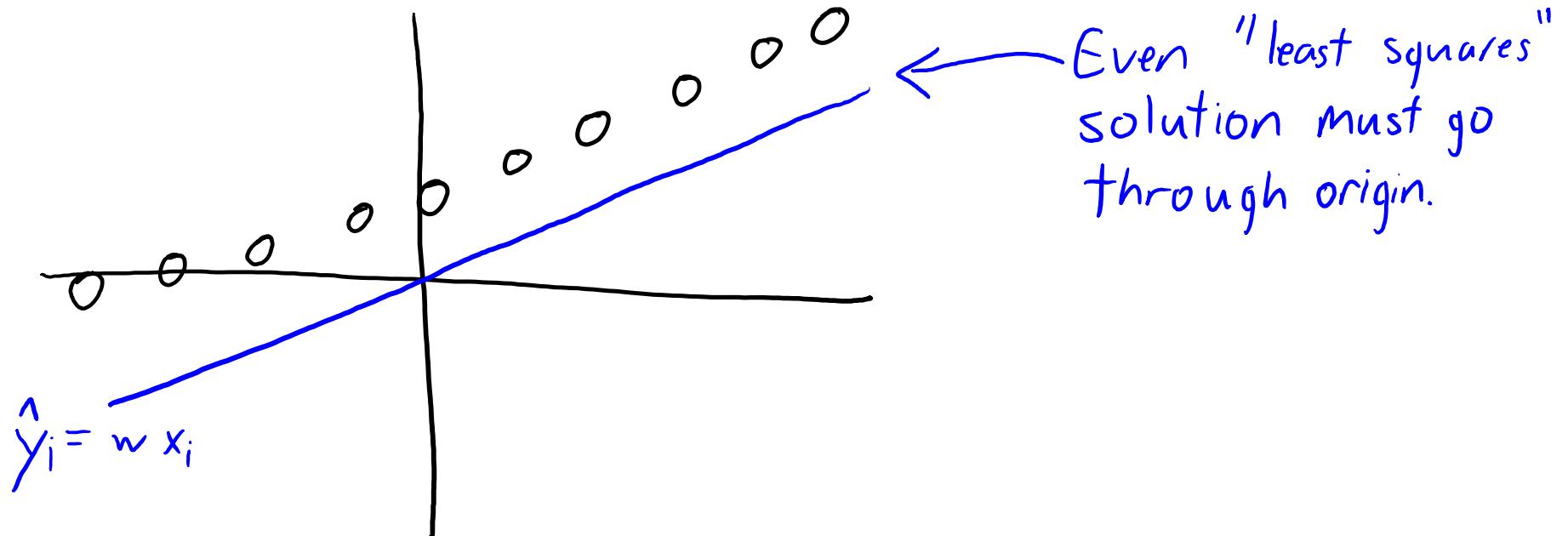
bonus!

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”.
 - See also “PatchMatch”: <https://vimeo.com/5024379>
 - KNN regression for “voice photoshop”:
 - <https://www.youtube.com/watch?v=l3l4XLZ59iw>
 - 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.

Why don't we have a y-intercept?

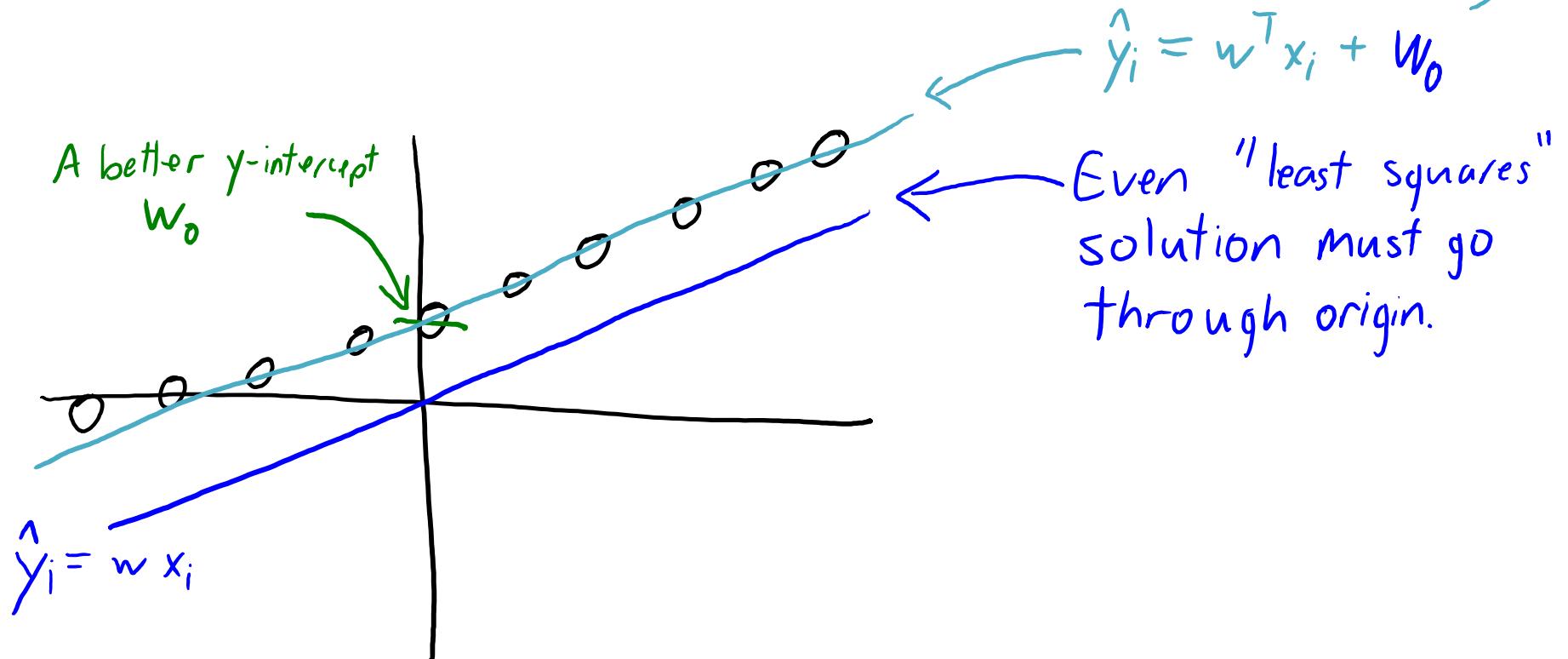
- Linear model is $\hat{y}_i = w x_i$ instead of $\hat{y}_i = w x_i + w_0$ with y-intercept w_0 .
- Without an intercept, if $x_i = 0$ then we must predict $\hat{y}_i = 0$.



Why don't we have a y-intercept?

- Linear model is $\hat{y}_i = w x_i$ instead of $\hat{y}_i = w x_i + w_0$ 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.



Adding a Bias Variable

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

$$X = \begin{bmatrix} -0.1 \\ 0.3 \\ 0.2 \end{bmatrix}$$

$$Z = \begin{bmatrix} 1 & -0.1 \\ 1 & 0.3 \\ 1 & 0.2 \end{bmatrix}$$

"always 1" X

- Now use “Z” as your features in linear regression.
 - We’ll use ‘v’ instead of ‘w’ as regression weights when we use features ‘Z’.

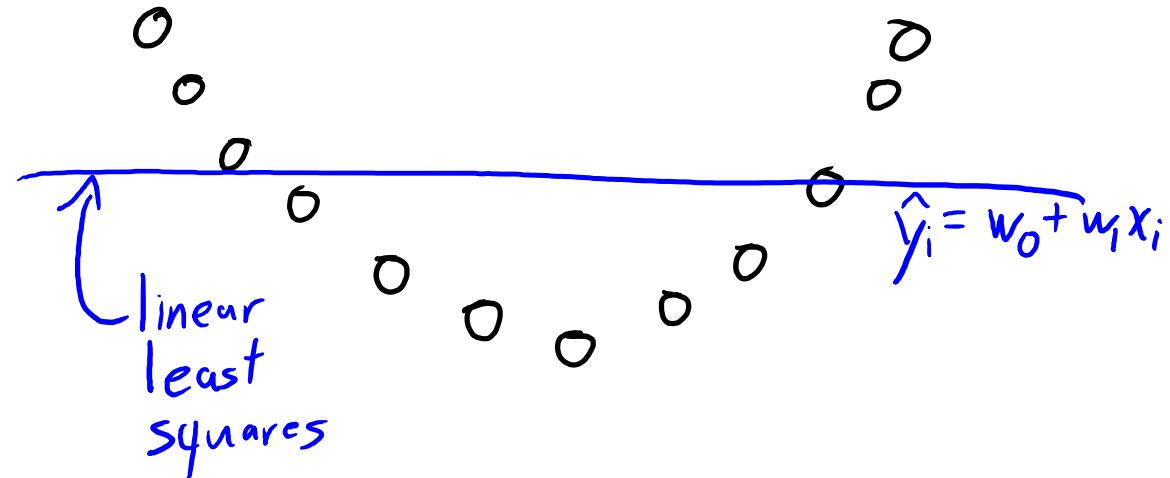
$$\hat{y}_i = v_1 Z_{i1} + v_2 Z_{i2} = w_0 + w_1 x_{i1}$$

w_0 1 w_1 x_{i1}

- So we can have a non-zero y-intercept by changing features.
 - This means we can ignore the y-intercept in our derivations, which is cleaner.

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):

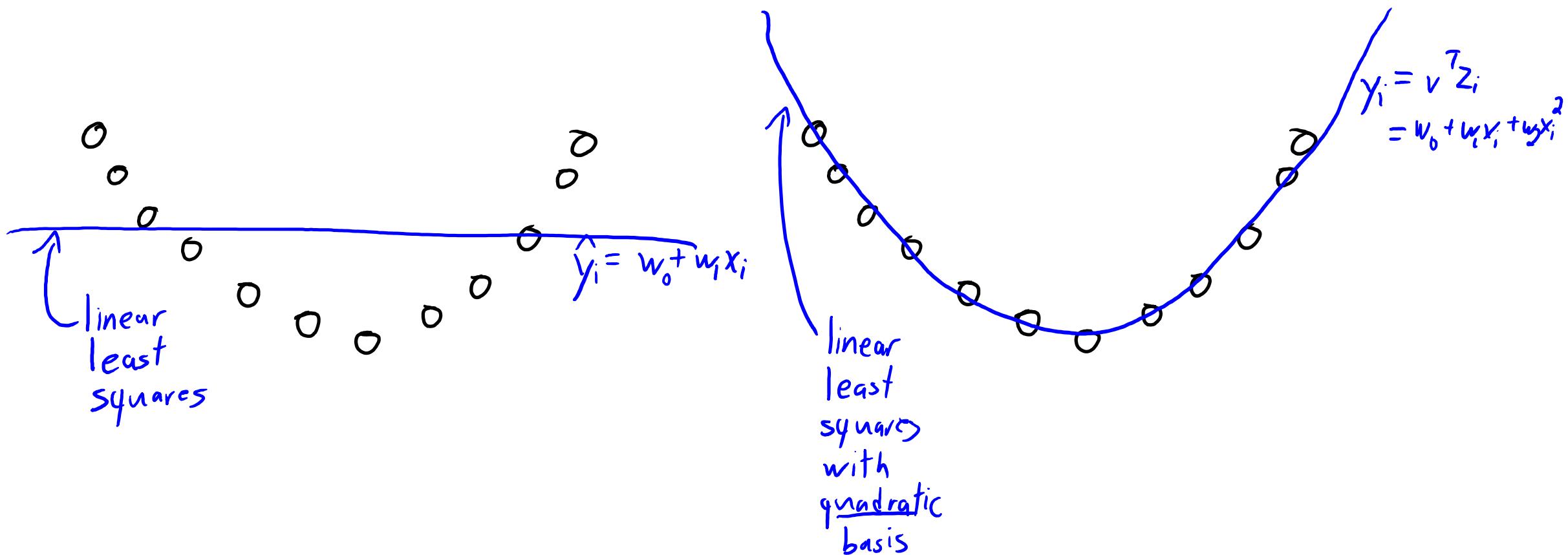
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix}$$

y-int x x²

- Fit new parameters 'v' under "change of basis": solve $Z^T Z v = Z^T y$.
- It's a linear function of w , but a quadratic function of x_i .

$$\hat{y}_i = v^T Z_i = \underbrace{v_1}_{w_0} \underbrace{z_{i,1}}_1 + \underbrace{v_2}_{w_1} \underbrace{z_{i,2}}_{x_i} + \underbrace{v_3}_{w_2} \underbrace{z_{i,3}}_{x_i^2}$$

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:

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

- There are polynomial basis functions that are numerically nicer:
 - E.g., Lagrange polynomials (see CPSC 303).

Summary

- Matrix notation for expressing least squares problem.
- Normal equations: solution of least squares as a linear system.
 - Solve $(X^T X)w = (X^T y)$.
- 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

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

Let's expand

then compute
gradient.

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

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

$$= \frac{1}{2} (w^T X^T (Xw - y) - y^T (Xw - y)) \quad (A+B)C = AC + BC$$

$$= \frac{1}{2} (w^T X^T Xw - w^T X^T y - y^T Xw + y^T y) \quad A(B+C) = AB + AC$$

$$= \frac{1}{2} w^T X^T Xw - w^T X^T y + \frac{1}{2} y^T y$$

Rule:

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

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

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

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

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

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

all of these are scalars.

bonus!

Vector View of Least Squares

- We showed that least squares minimizes:

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

- The $\frac{1}{2}$ 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 $X w$.

bonus!

Bonus Slide: Householder(-ish) Notation

- **Householder 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: $w = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$, $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $a = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

→ 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!

Bonus Slide: Householder(-ish) Notation

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

Our ultimate least squares notation:

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

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?

bonus!

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”, $X^T X$ is invertible, $X^T X$ has non-zero eigenvalues, $\det(X^T X) > 0$.
 - Note that we cannot have independent columns if $d > n$.