CPSC 340:
Machine Learning and Data Mining

Gradient Descent

Summer 2021

In This Lecture

1. Analyzing Least Squares (10 minutes)
2. Change of Basis (15 minutes)
3. Gradient Descent (15 minutes)

Coming Up Next

ANALYZING LEAST SQUARES

Least Squares Cost

- Cost of solving "normal equations" $X^TXw = X^Ty$?
- Forming X^Ty vector costs O(__).
  - It has 'd' elements, and each is an inner product between 'n' numbers.
- Forming matrix X^TX costs O(__).
  - 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(_________).
  - Which term dominates depends on 'n' and 'd'.

Least Squares Issues

- Issues with least squares model:
  - Solution might not be _______.
  - It is sensitive to _______.
  - It always uses all features.
  - What is we had a million features?
    - Difficult to store X^TX (WHY?)
    - $O(nd^2 + d^3)$ time cost will be huge
  - It might predict outside range of y_i values.
  - It assumes a linear relationship between x_i and y_i.

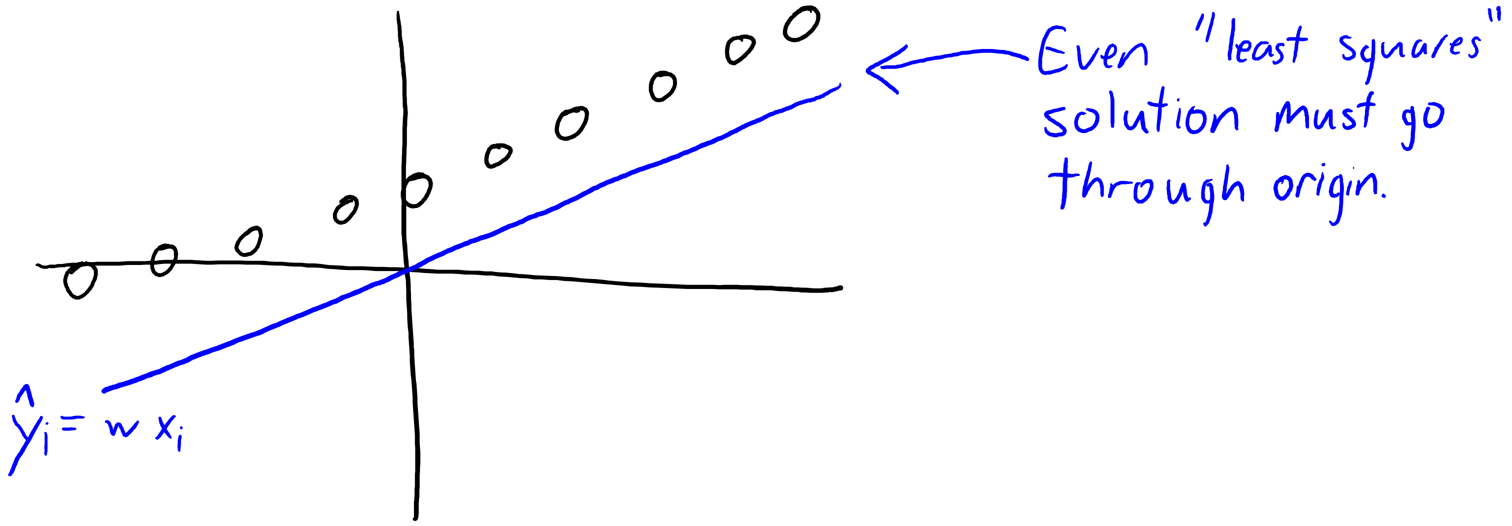
Non-Uniqueness of Least Squares Solution

- 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. $\hat{y}_i = w_1 x_{i1} + w_2 \underbrace{x_{i1}}_{\text{copy}} = (w_1 + w_2) x_{i1} + 0 x_{i1}$
  - Thus, if (w_1, w_2) is a solution then $(w_1+w_2, 0)$ is another solution.
  - This is special case of features being "collinear":
    - One feature is a linear function of the others.

Q: Will this break my model?

Why don’t we have a y-intercept?

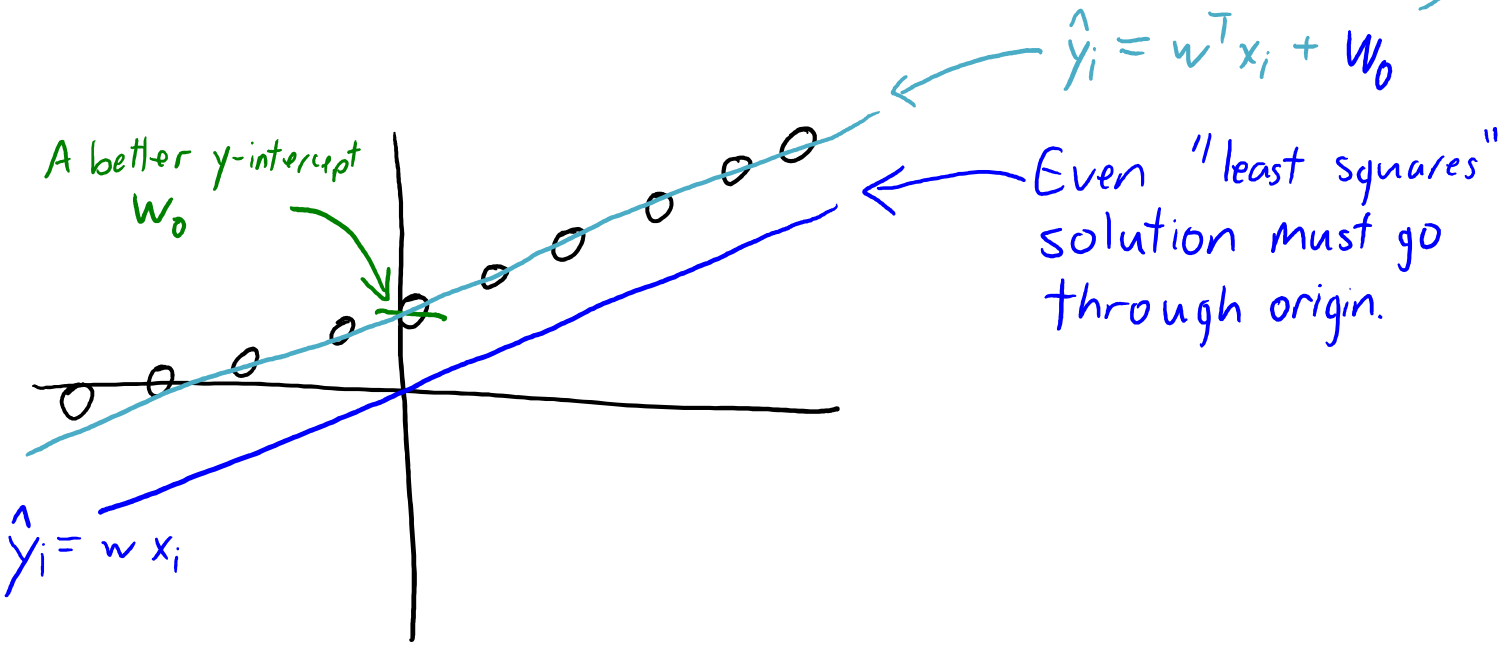
- Linear model is $\hat{y}_i = wx_i$ instead of $\hat{y}_i = wx_i + w_0$ with y-intercept w_0.
- Without an intercept, if $x_i = 0$ then we must predict $\hat{y}_i = 0$.



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.
A better y-intercept w_0
$\hat{y}_i = w x_i$

Adding a Bias Variable

- Simple trick to add a y-intercept ("bias") variable:
  - Make a new matrix "Z" with a ____________________.

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

$$\underbrace{}_{\text{"always 1"}} \quad \underbrace{}_{X}$$

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

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

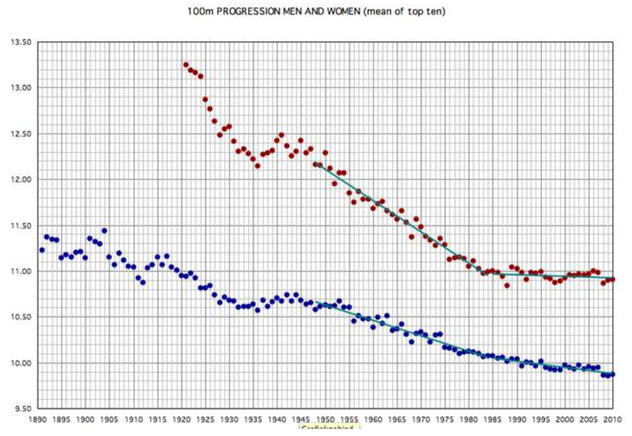
- So we can have a non-zero y-intercept by changing features.
  - This means we can ignore the y-intercept in our derivations, which is cleaner.

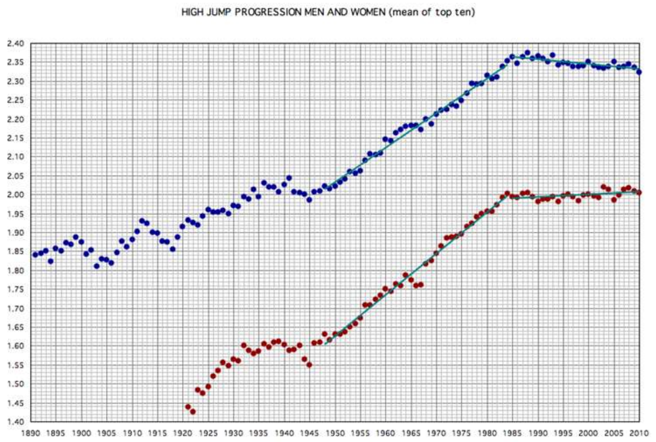
Coming Up Next

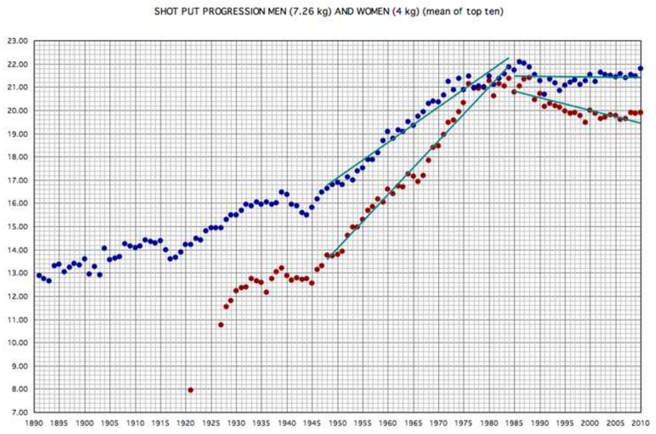
CHANGE OF BASIS

Motivation: Non-Linear Progressions in Athletics

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



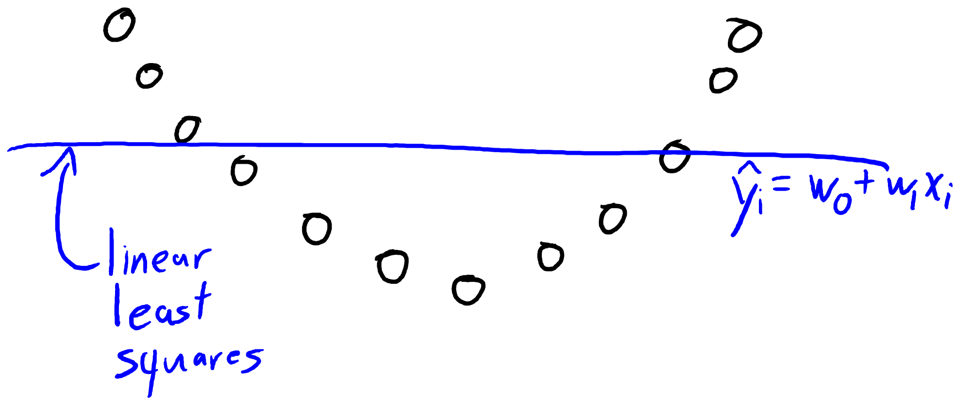




http://www.at-a-lanta.nl/weia/Progressie.html
https://en.wikipedia.org/wiki/Usain_Bolt
http://www.britannica.com/biography/Florence-Griffith-Joyner

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?

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

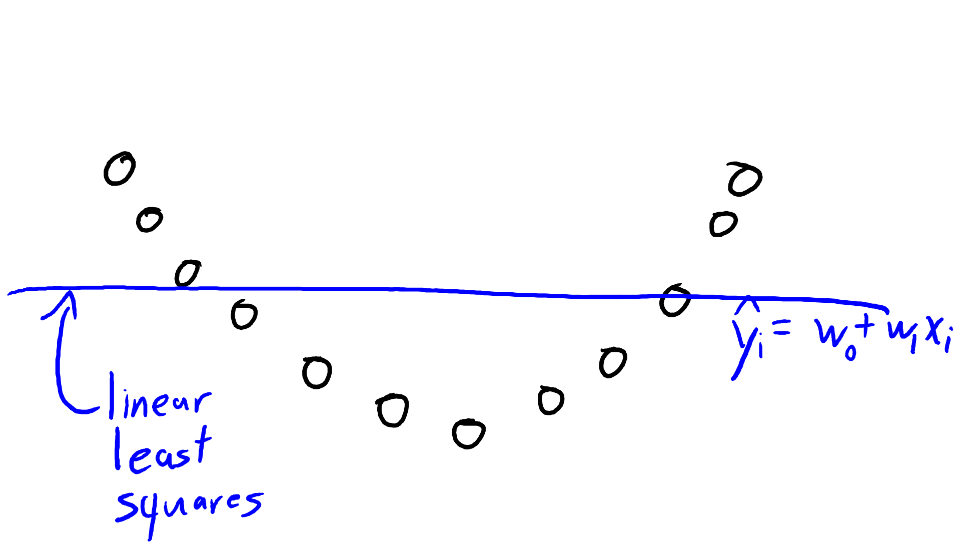
- You can do this by changing the features (change of ________):

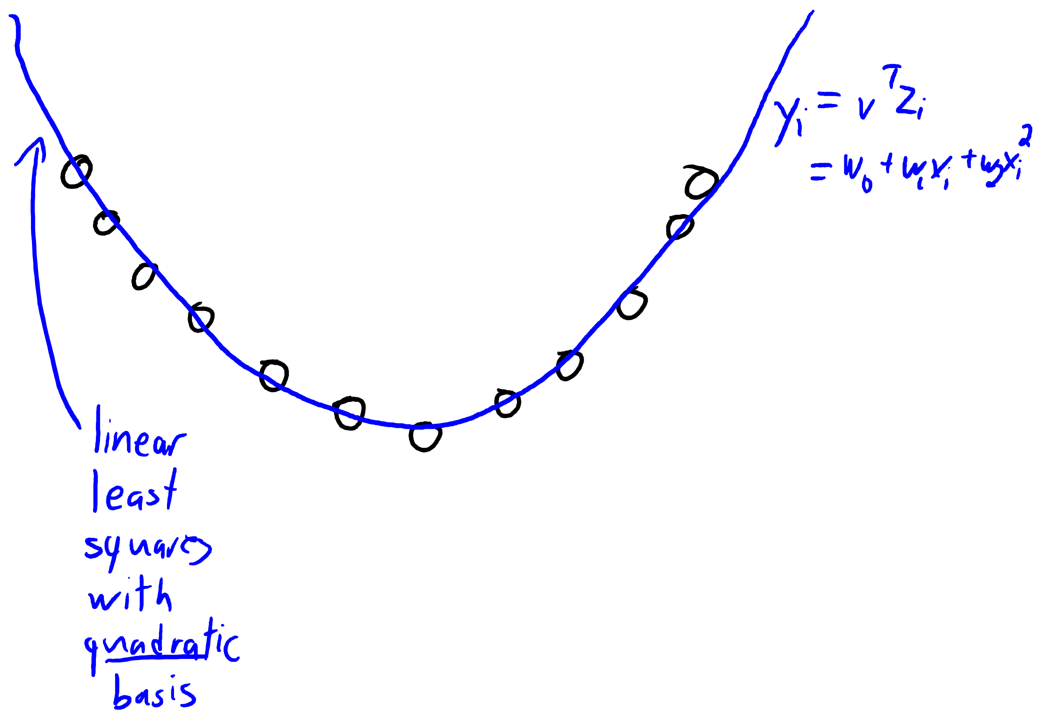
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad Z = \begin{array}{c} \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix} \\ \begin{array}{ccc} y\text{-int} & x & x^2 \end{array} \end{array}$$

- 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 $y = \tilde{Z}v$

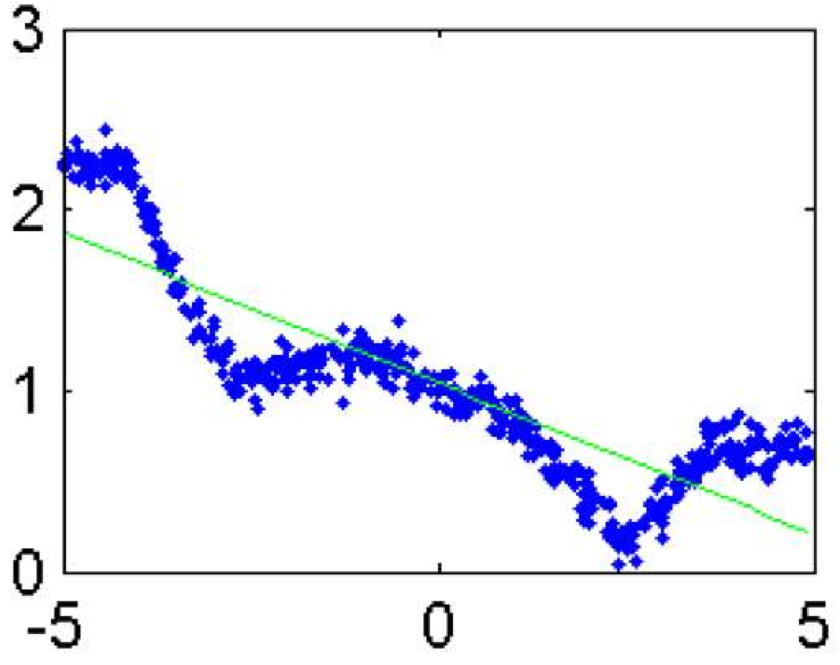
General Polynomial Features (d=1)

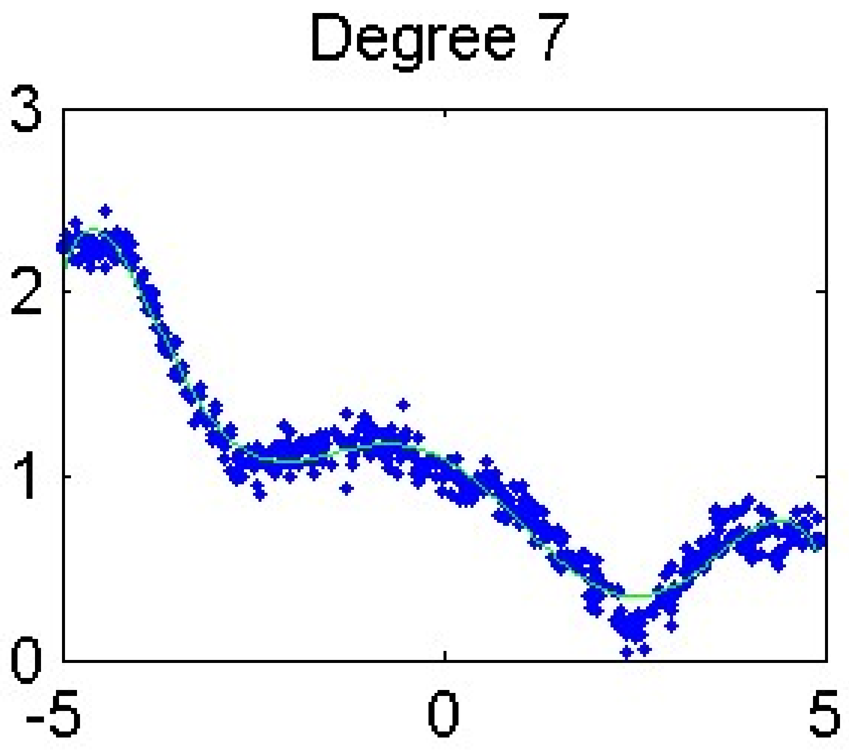
- We can have a polynomial of degree 'p' by using these features:

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

- 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 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

(You'll use this in A3)

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 _-dimensional 'w' by minimizing the squared error:

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

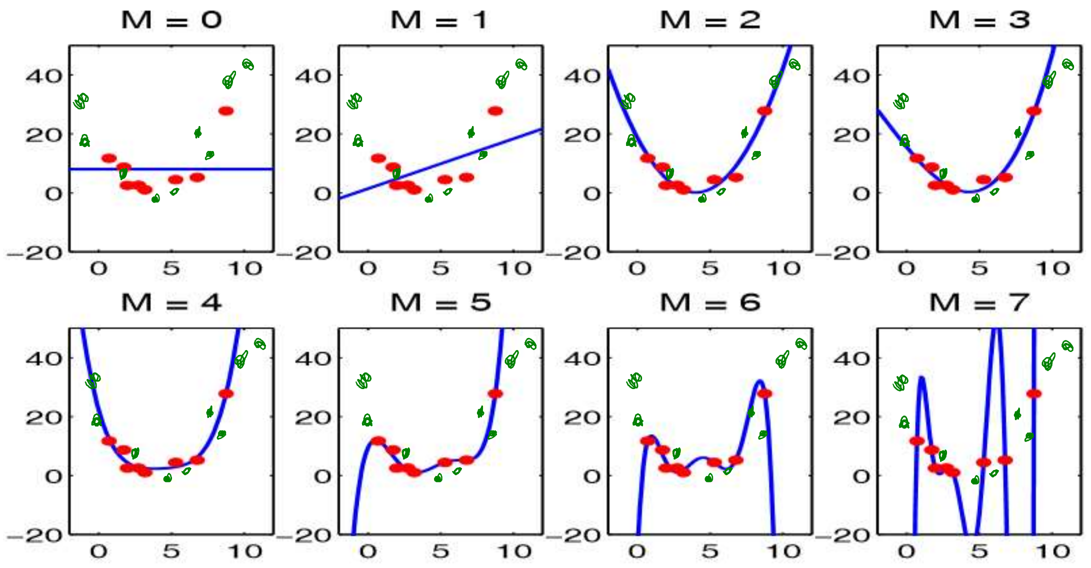
- Linear regression with nonlinear feature transforms:
  - We use 'Z' as our "n by k" data matrix, and 'v' as our parameters.
  - We can find _-dimensional 'v' by minimizing the squared error:

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

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

Degree of Polynomial and Fundamental Trade-Off

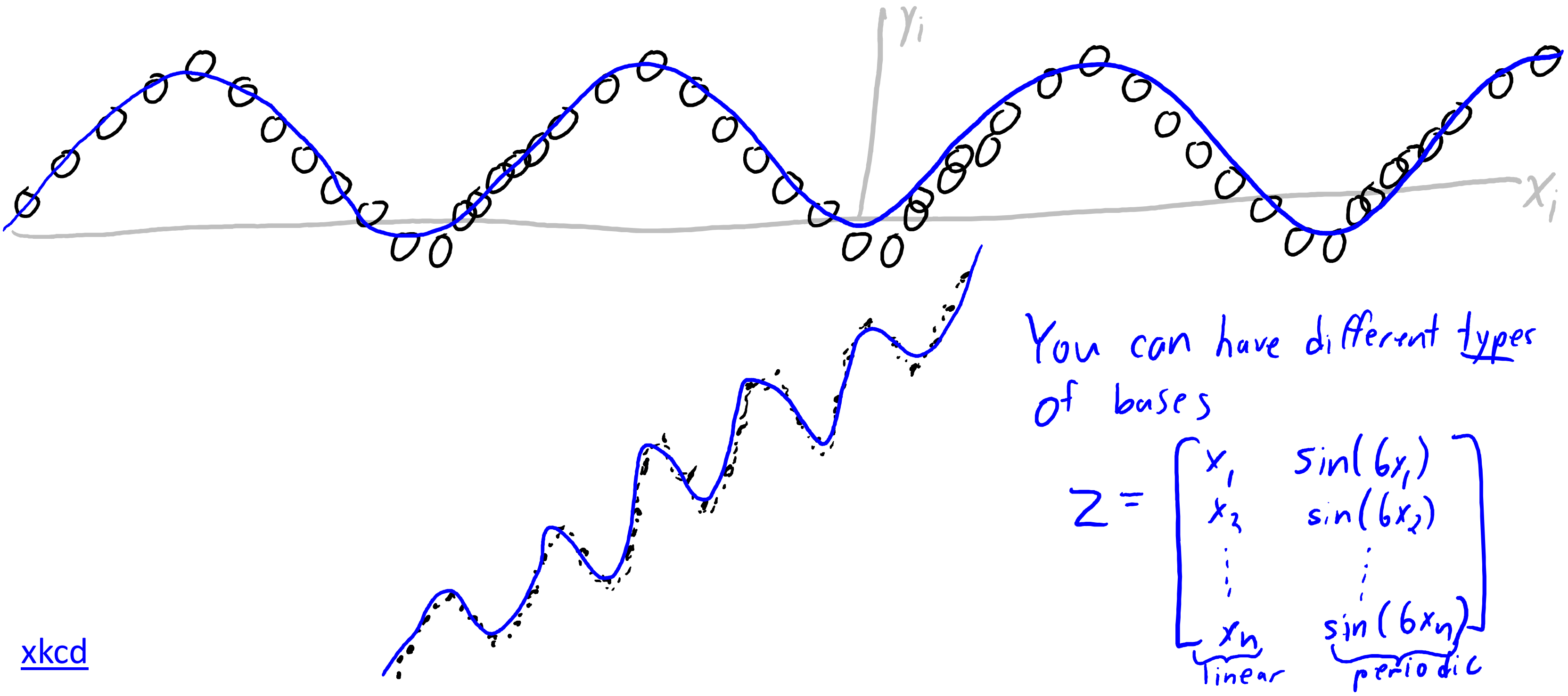
- As the polynomial degree increases, the training error goes down.



- But approximation error goes up: we start overfitting with large ‘p’.
- Usual approach to selecting degree: validation or cross-validation (A3)

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

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

xkcd

Mount Vesuvius “Zorbing”

Coming Up Next

GRADIENT DESCENT INTRO

Optimization Terminology

- When we minimize or maximize a function we call it "optimization".
  - In least squares, we want to solve the "optimization problem":

$$\underset{\underset{\text{Parameters}}{w}\,\in\,\underset{\text{domain (search space of parameters)}}{\mathbb{R}^d}}{\text{Minimize}} \quad \underbrace{\frac{1}{2}\sum_{i=1}^{n}(w^Tx_i-y_i)^2}_{\text{Objective}}$$

$$\underset{\text{minimizer}}{w^*} = \underset{w\in\mathbb{R}^d}{\operatorname{argmin}}\ f(w)$$

Discrete vs. Continuous Optimization

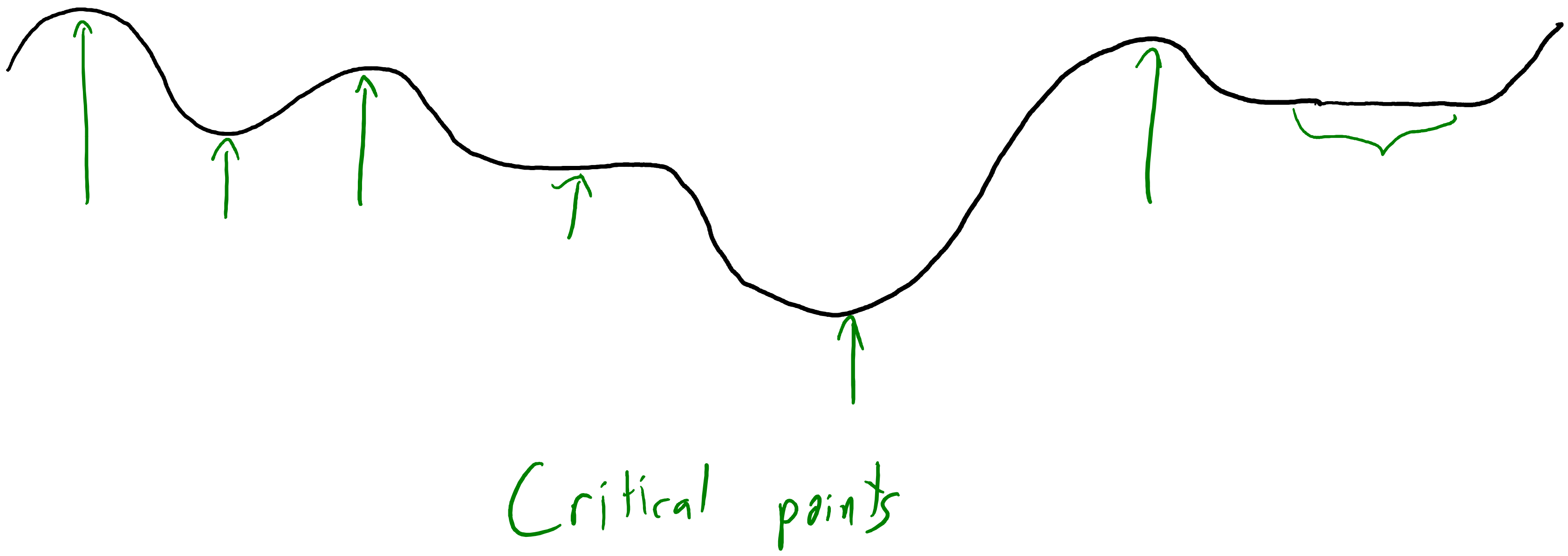
- We have seen examples of continuous optimization:
  - _______________:
    - Domain is the real-valued set of parameters ‘w’.
    - Objective is the sum of the squared training errors.

- We have seen examples of discrete optimization:
  - _______________:
    - Domain is the grid (finite set) of unique rules {j, t}.
    - Objective is the number of classification errors (or infogain).

- We have also seen a mixture of discrete and continuous:
  - ____________: clusters are discrete and means are continuous.

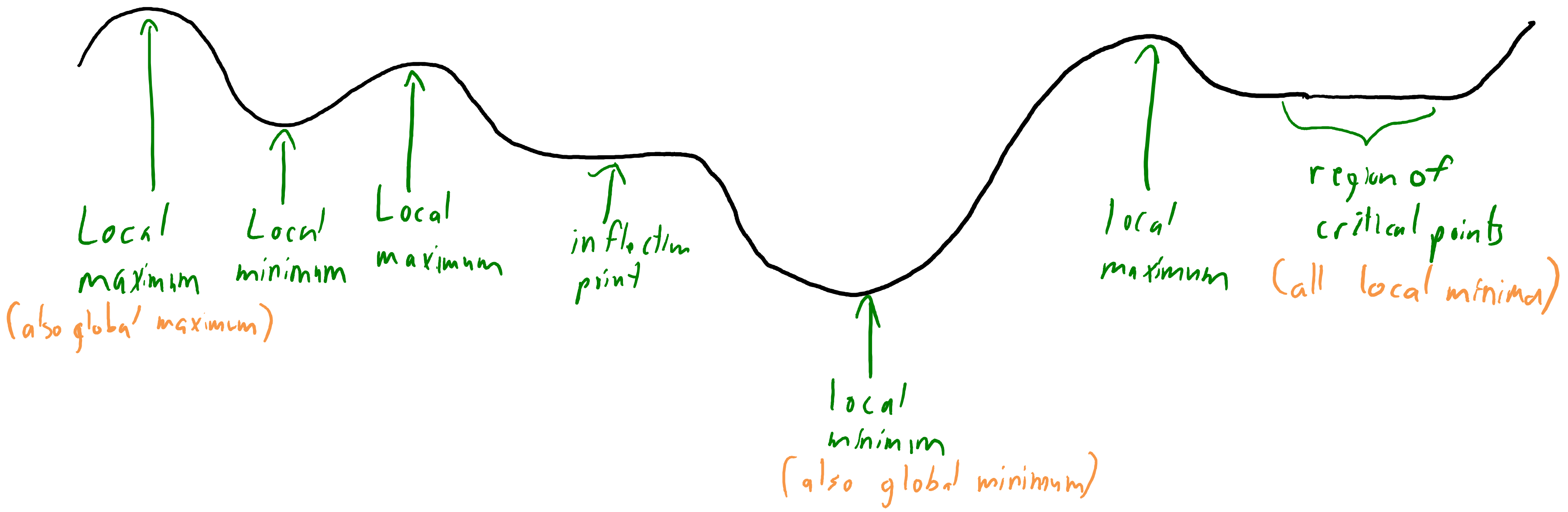
Stationary/Critical Points

- A ‘w’ with ∇ f(w) = 0 is called a stationary point or critical point.
  - The ______ is zero so the tangent plane is “flat”.



Stationary/Critical Points

- A ‘w’ with $\nabla f(w) = 0$ is called a stationary point or critical point.
  - The slope is zero so the tangent plane is “flat”.



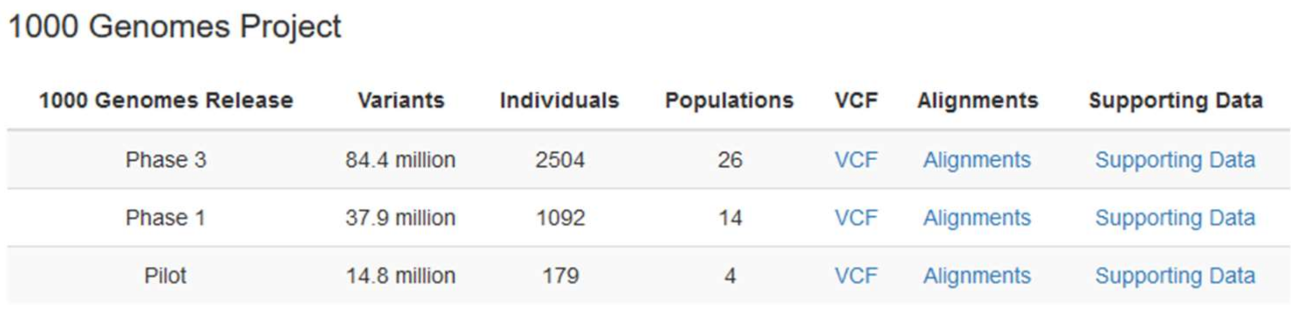
  - If we’re minimizing, we would ideally like to find a global minimum.
    - But for some problems the best we can do is find a stationary point where $\nabla f(w)=0$.

Motivation: Large-Scale Least Squares

- Recall: normal equations find ‘w’ with ∇ f(w) = 0 in $O(nd^2 + d^3)$ time.

$$(X^TX)w = X^Ty$$

  - Very slow if ‘d’ is large.



1000 Genomes Project

1000 Genomes Release	Variants	Individuals	Populations	VCF	Alignments	Supporting Data
Phase 3	84.4 million	2504	26	VCF	Alignments	Supporting Data
Phase 1	37.9 million	1092	14	VCF	Alignments	Supporting Data
Pilot	14.8 million	179	4	VCF	Alignments	Supporting Data

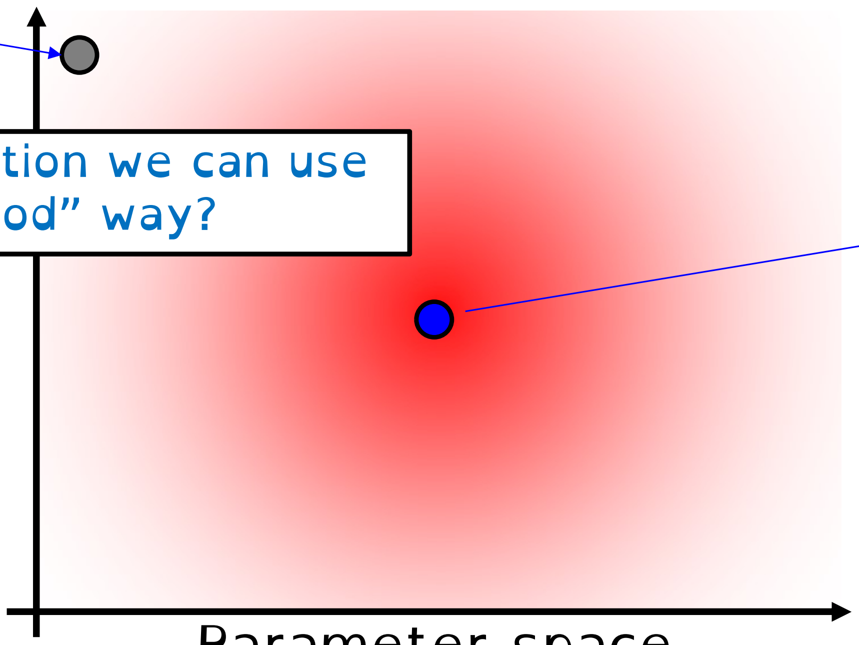
n=2504, d=84.4 million!!!

- Alternative when ‘d’ is large is gradient descent methods.
  - Probably the most important class of algorithms in machine learning.

What is Gradient Descent?

- Goal: navigate the parameter space and find a locally optimal parameter value

Let's say we start here



Q: Is there any information we can use to move in a "good" way?

$f: \mathbb{R}^d \longrightarrow \mathbb{R}$

Thicker colour:
Lower objective value

The blue point is a

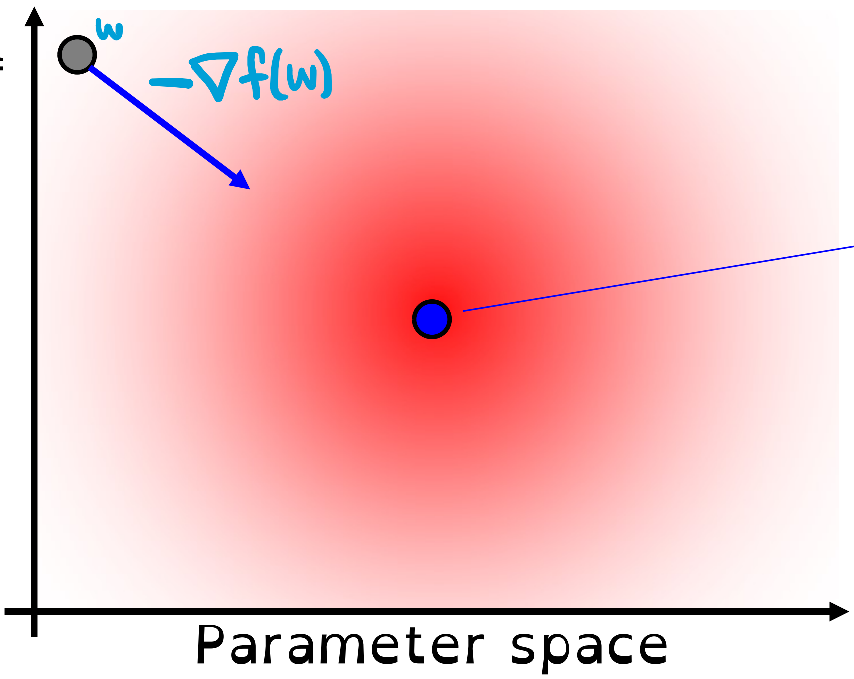
____

What is Gradient Descent?

- Goal: navigate the parameter space and find a locally optimal parameter value

1. Negative gradient is a _________
2. Negative gradient is the direction and magnitude of

____



$f: \mathbb{R}^d \longrightarrow \mathbb{R}$

Thicker colour: Lower objective value

The blue point is a minimizer

Gradient Descent for Finding a Local Minimum

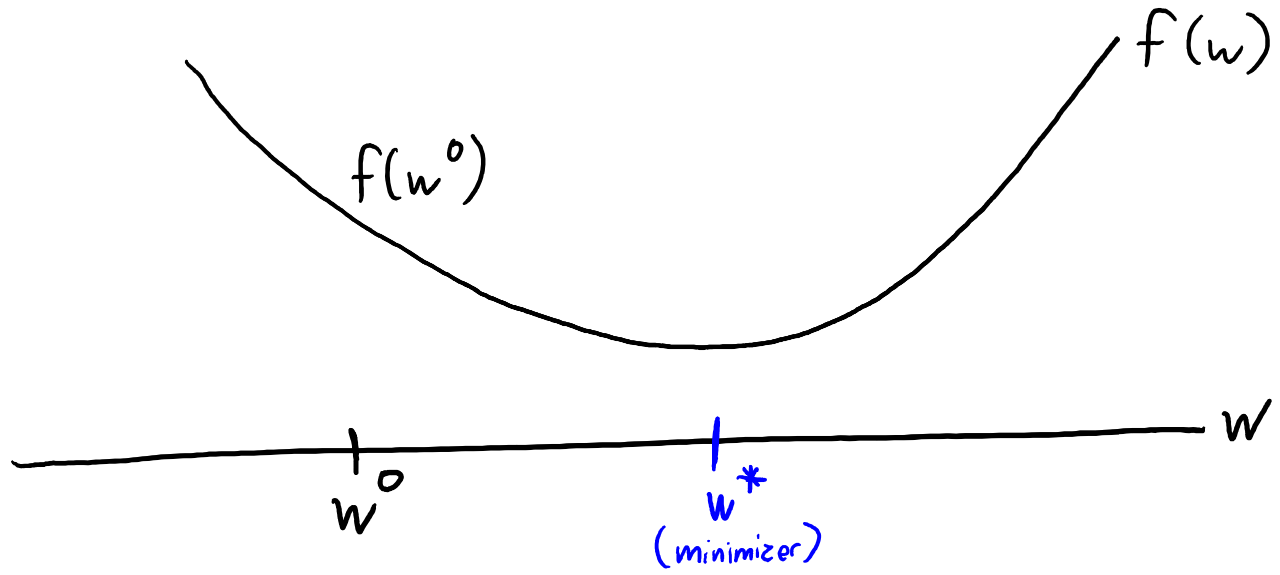
- Gradient descent is an iterative optimization algorithm:
  - It starts with a "guess" w^0.
  - It uses the gradient $\nabla f(w^0)$ to generate a better guess w^1.
  - It uses the gradient $\nabla f(w^1)$ to generate a better guess w^2.
  - It uses the gradient $\nabla f(w^2)$ to generate a better guess w^3.

    ...
  - The limit of w^t as 't' goes to ∞ has $\nabla f(w^t) = 0$.

- It converges to a global optimum if 'f' is "convex".

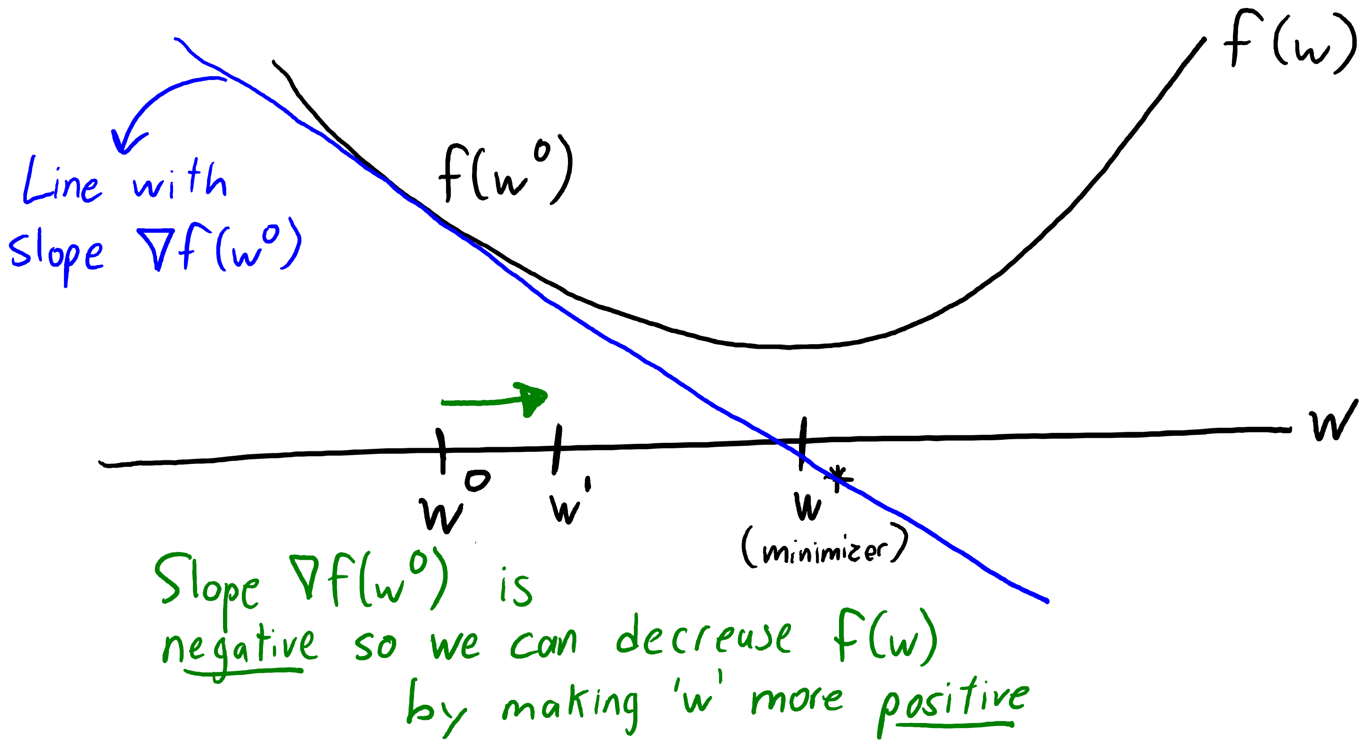
Gradient Descent for Finding a Local Minimum

- Gradient descent is based on a simple observation:
  - Give parameters ‘w’, the direction of largest decrease is $-\nabla f(w)$.



Gradient Descent for Finding a Local Minimum

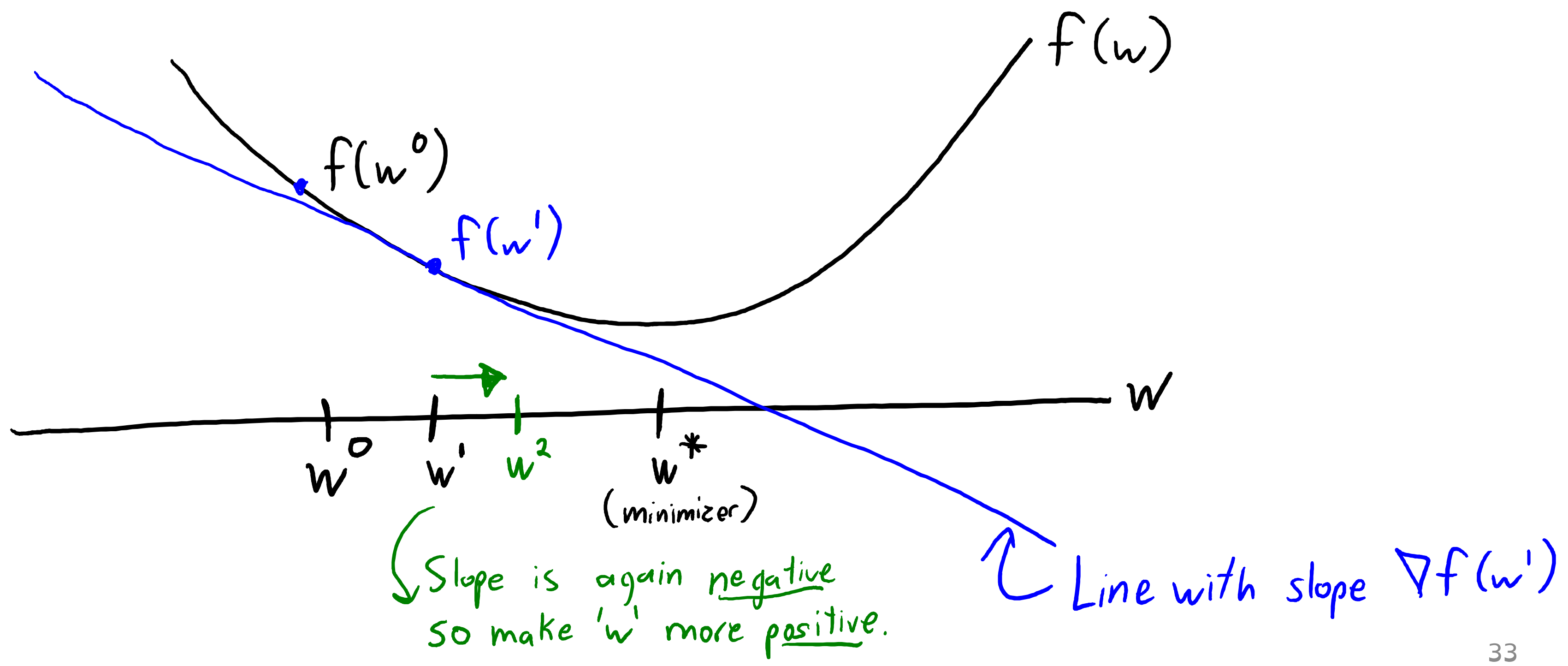
- Gradient descent is based on a simple observation:
  - Give parameters ‘w’, the direction of largest decrease is $-\nabla f(w)$.



f(w)
$f(w^0)$
Line with slope $\nabla f(w^0)$
w
w^0
w^1
w^*
(minimizer)
Slope $\nabla f(w^0)$ is negative so we can decrease f(w) by making ‘w’ more positive

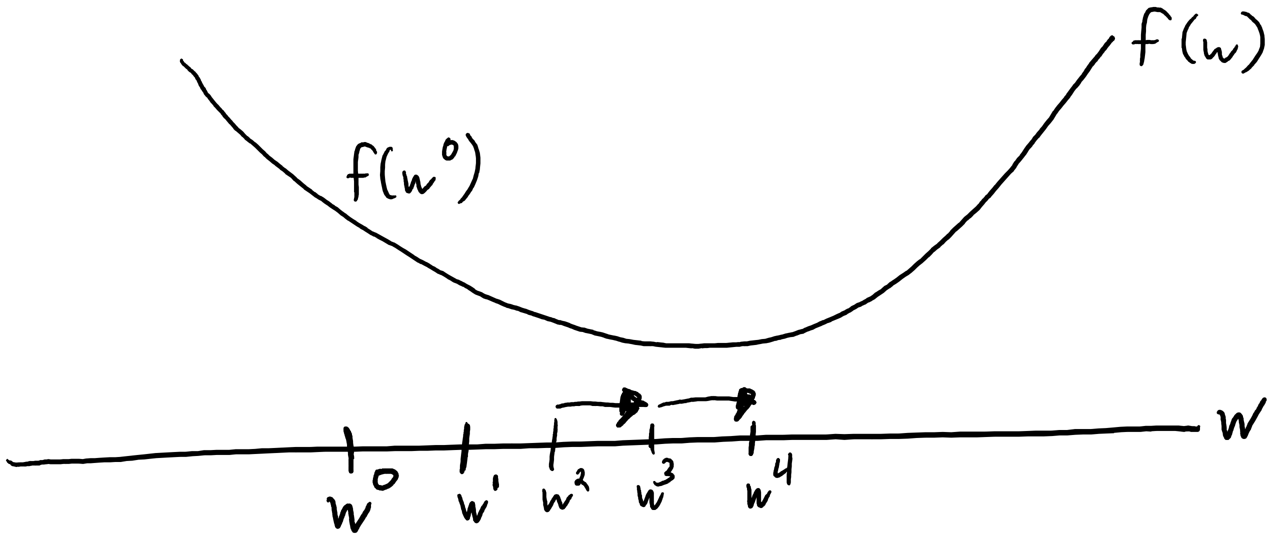
Gradient Descent for Finding a Local Minimum

- Gradient descent is based on a simple observation:
  - Give parameters ‘w’, the direction of largest decrease is $-\nabla f(w)$.



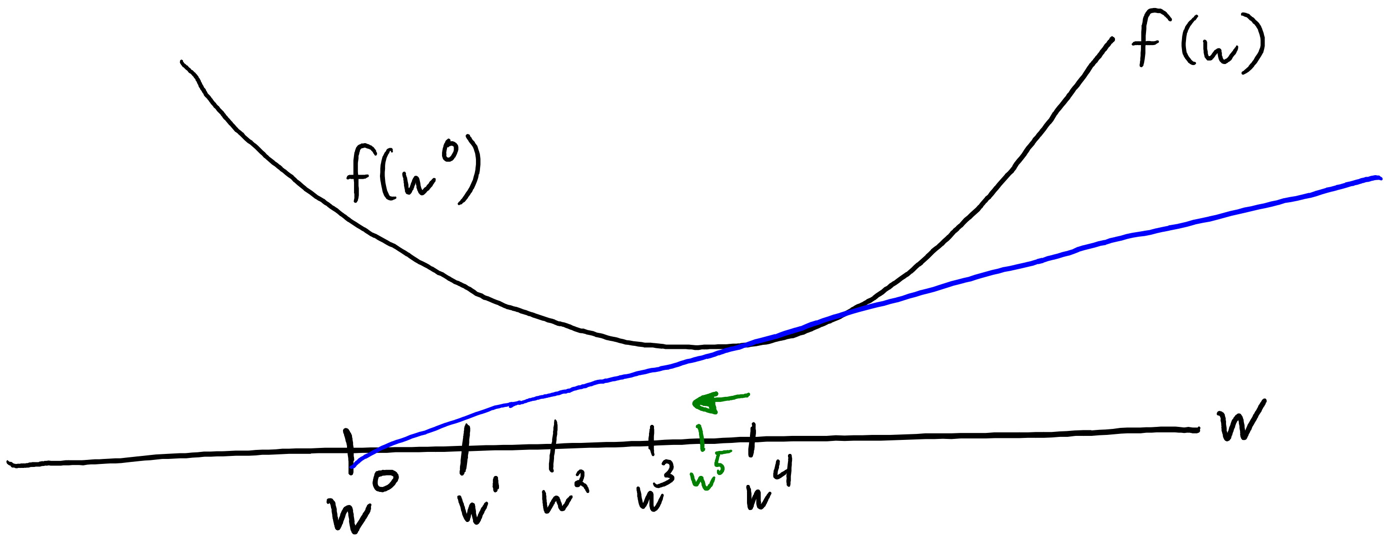
Gradient Descent for Finding a Local Minimum

- Gradient descent is based on a simple observation:
  - Give parameters ‘w’, the direction of largest decrease is $-\nabla f(w)$.



Gradient Descent for Finding a Local Minimum

- Gradient descent is based on a simple observation:
  - Give parameters 'w', the direction of largest decrease is $-\nabla f(w)$.



Now the slope $\nabla f(w^4)$ is positive
so we move in the negative direction.

Coming Up Next

MORE FORMAL DISCUSSION OF GRADIENT DESCENT

Gradient Descent for Finding a Local Minimum

- We start with some initial guess, w^0.
- Generate new guess by moving in the negative gradient direction:

$$\underbrace{w^1}_{\text{new guess}} = \underbrace{w^0}_{\text{old guess}} - \underbrace{\alpha^0}_{\text{step size or ``learning rate''}} \underbrace{\nabla f(w^0)}_{\text{gradient evaluated at } w^0}$$

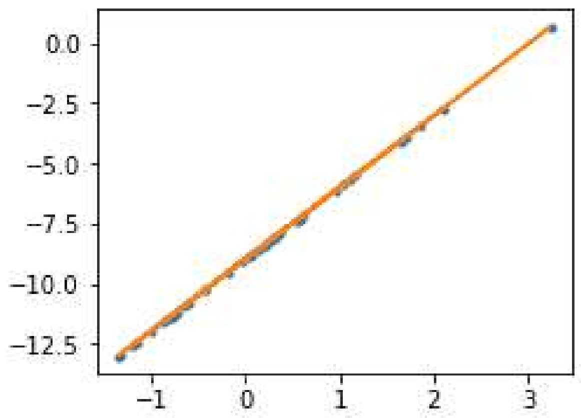
  - This decreases 'f' if the "step size" α^0 is small enough.
  - Usually, we decrease α^0 if it increases 'f' (see A3 "optimizers.py").
- Repeat to successively refine the guess:

$$w^{t+1} = w^t - \alpha^t \nabla f(w^t) \quad \text{for } t = 1, 2, 3, \ldots$$

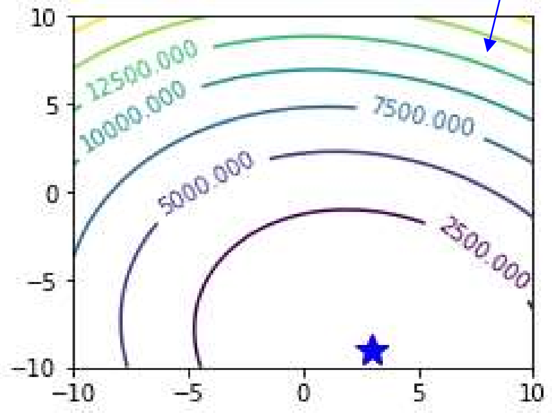
- Stop if not making progress or $\underbrace{\|\nabla f(w^t)\| \leq \epsilon}_{\text{Approximate local minimum}}$ $\hookrightarrow$ Some small scalar.

Gradient Descent in 2D Regression

Q: What do these boundaries mean?

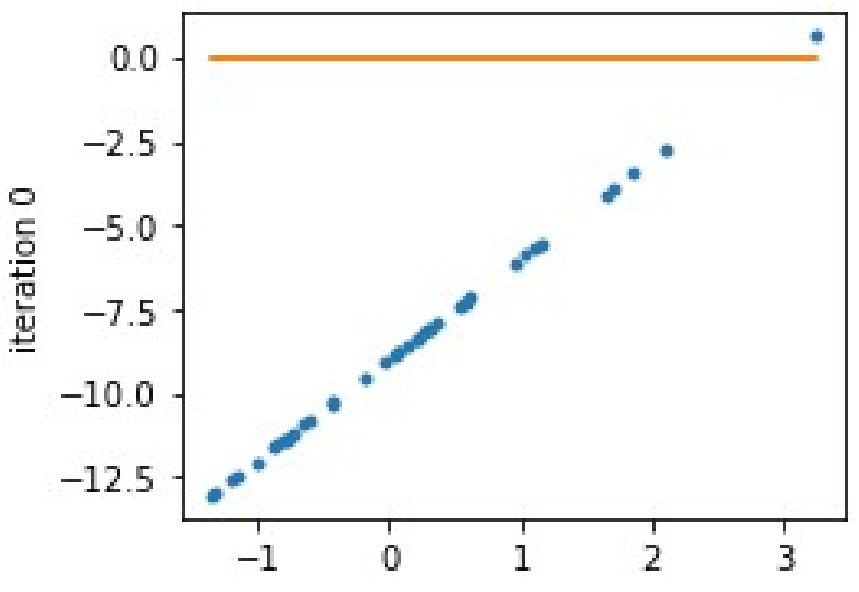


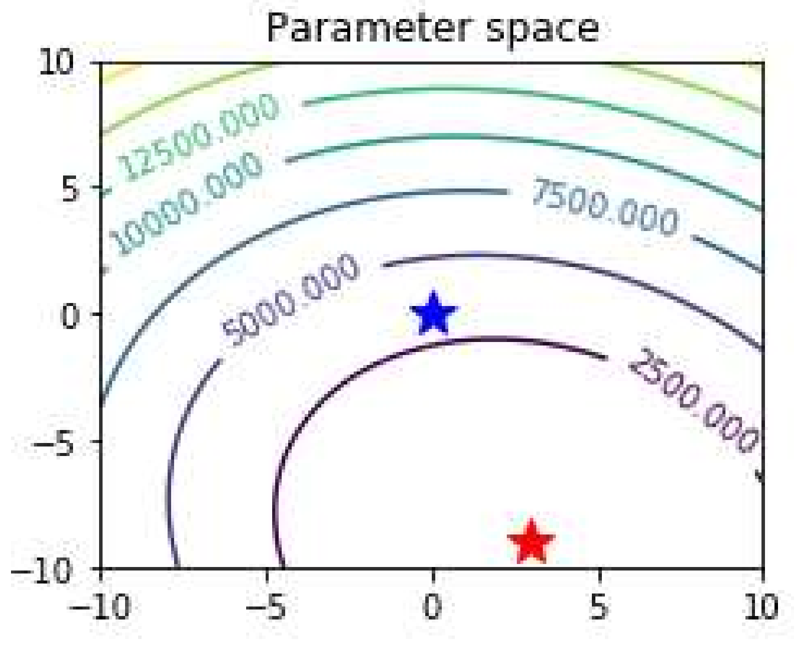
Feature space



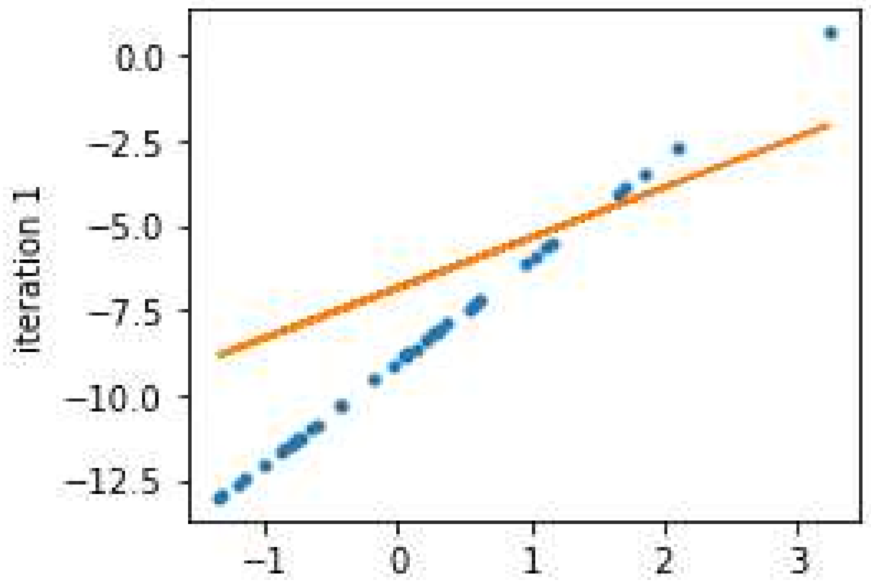
Parameter space

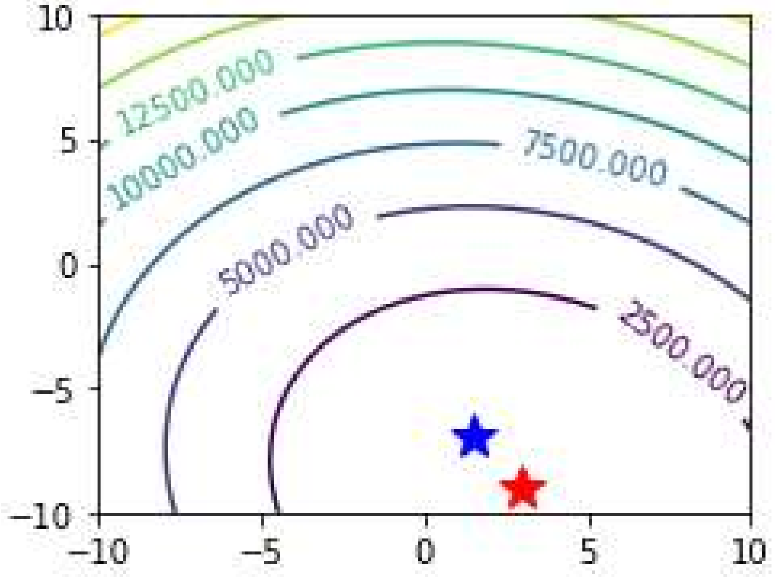
Gradient Descent in 2D Regression



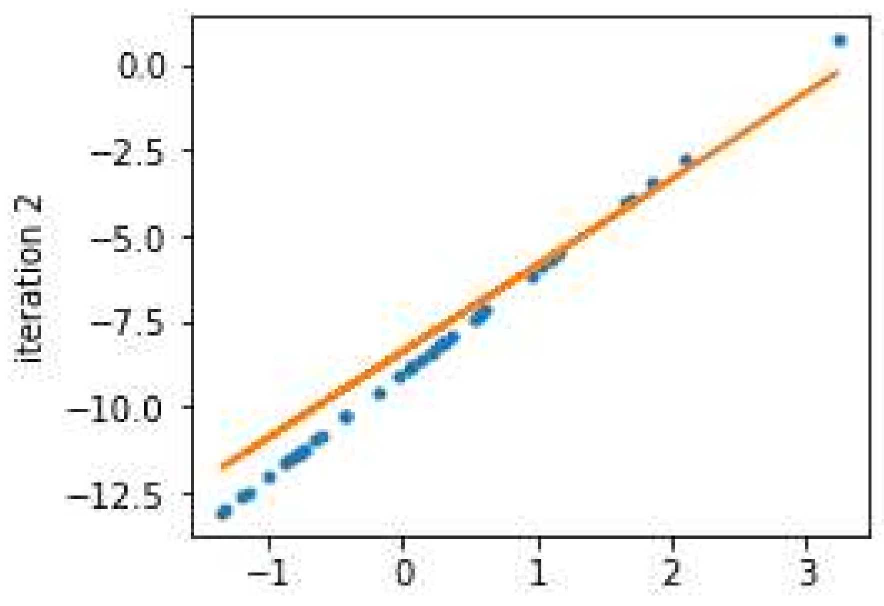


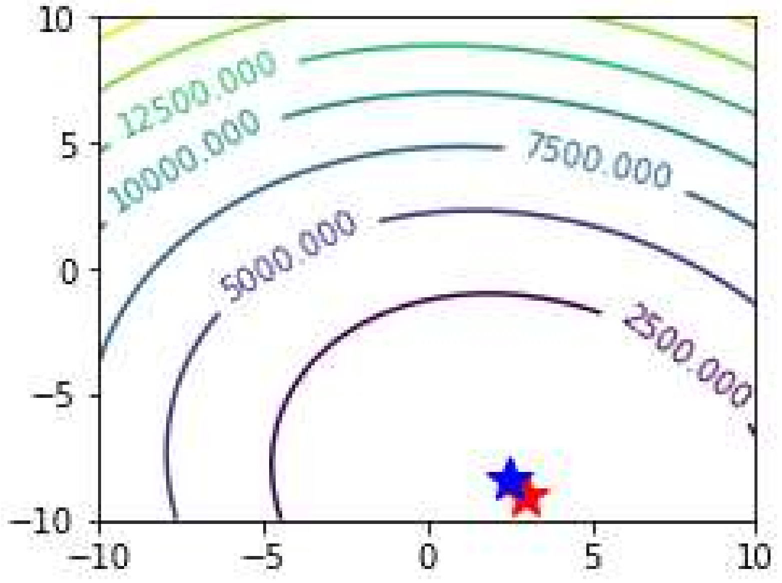
Gradient Descent in 2D Regression



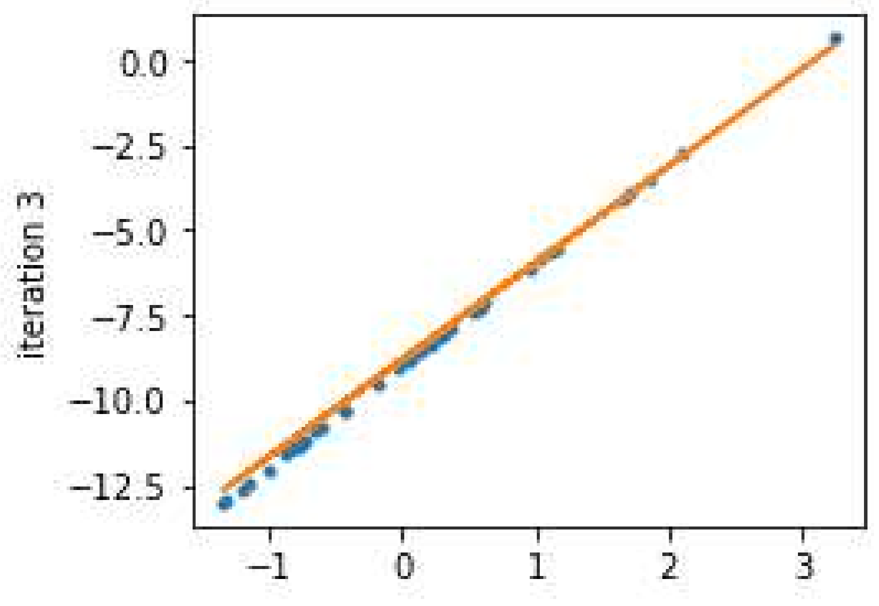


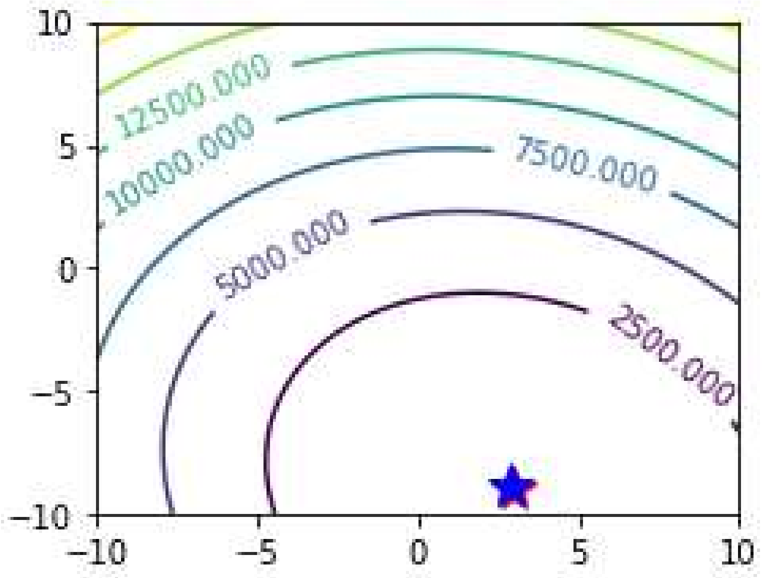
Gradient Descent in 2D Regression



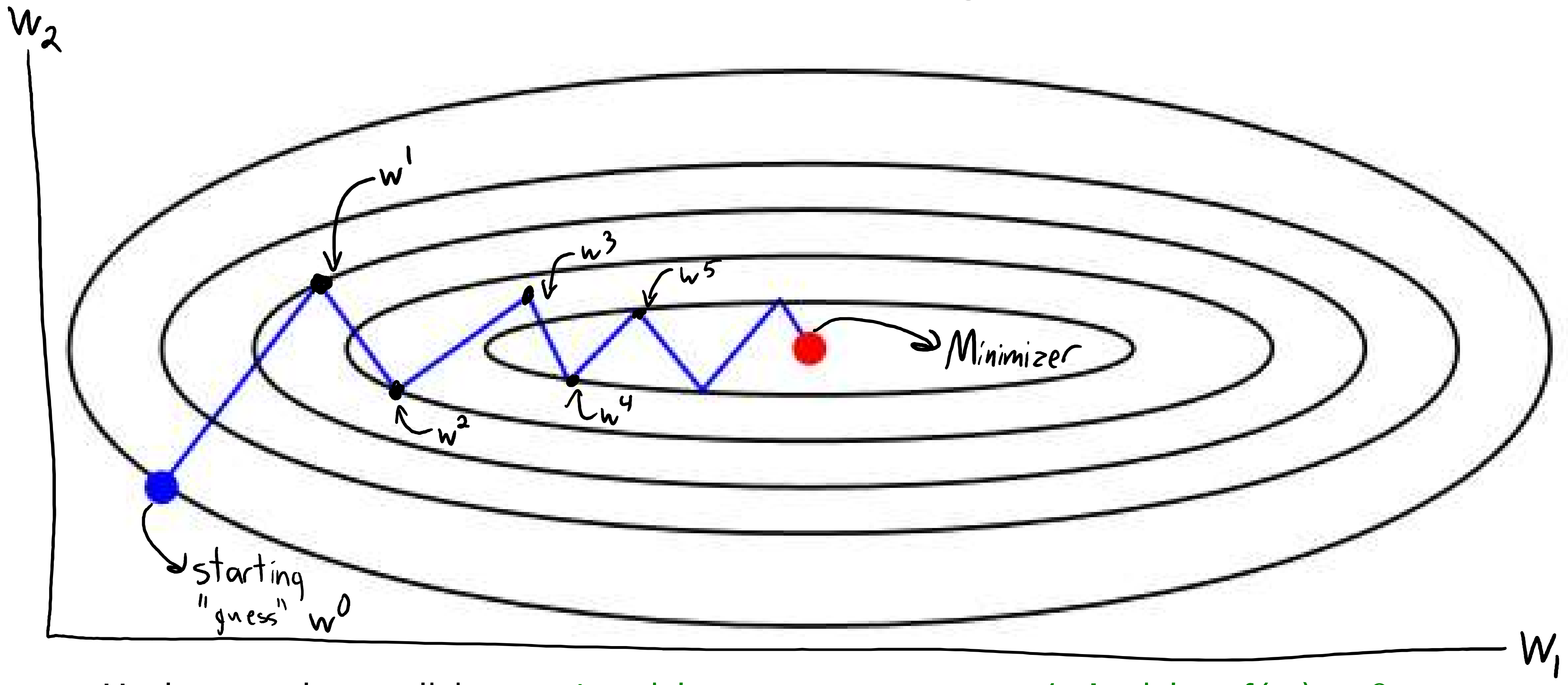


Gradient Descent in 2D Regression





“Parameter Trajectory” According to Gradient Descent



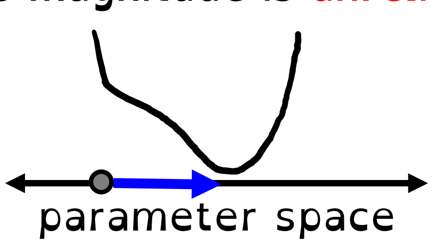
- Under weak conditions, algorithm converges to a ‘w’ with $\nabla f(w) = 0$.
  - ‘f’ is bounded below, ∇f can’t change arbitrarily fast, small-enough constant α^t.

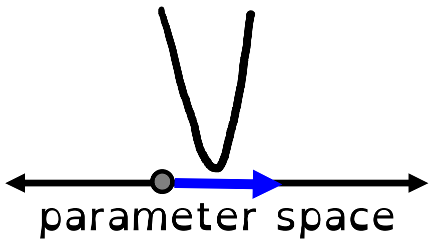
Step Size Considerations

$$w^{t+1} = w^t - \alpha^t \nabla f(w^t)$$

Q: Why is t in α^t ?

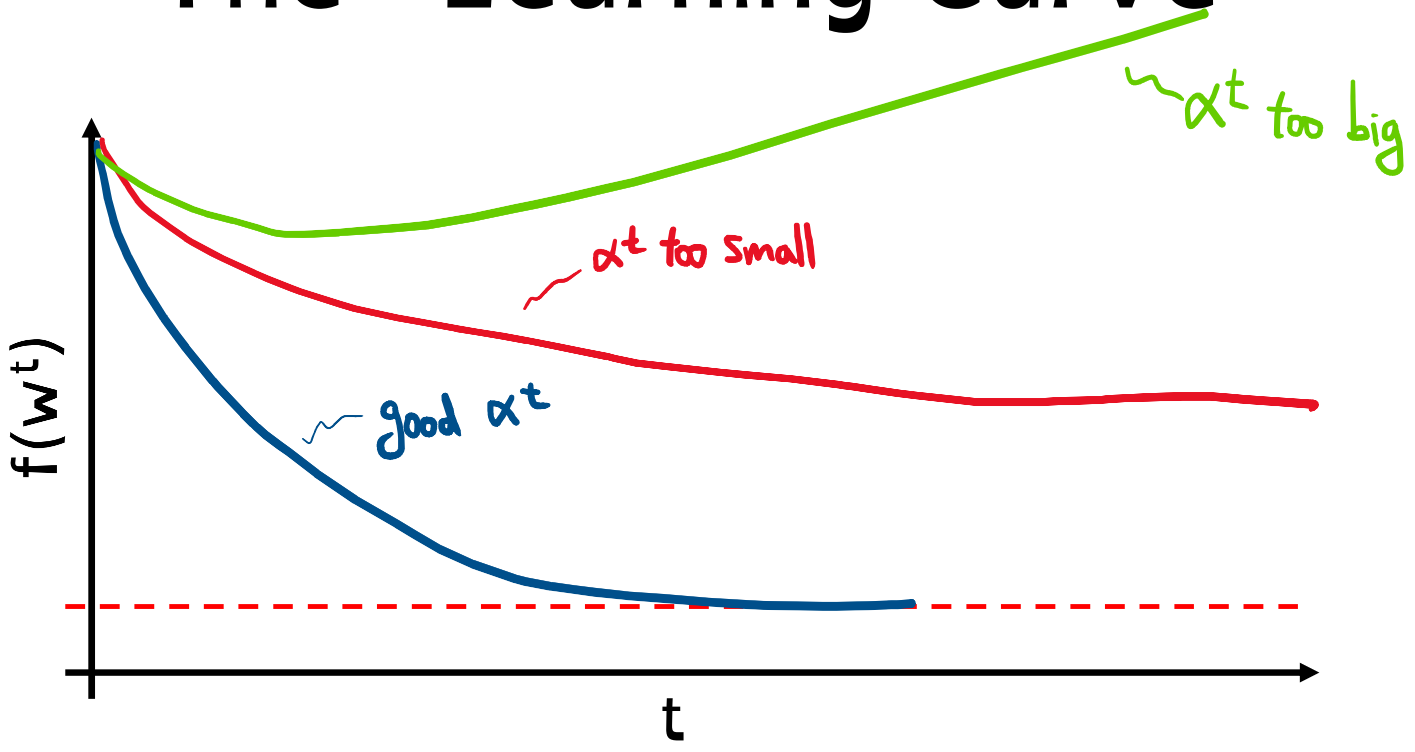
- $-\nabla f(w^t)$ has __________ and ____________ of steepest decrease
  - But this magnitude is unreliable!





- α_t must be "tuned" carefully for gradient descent to work
  - Too large, we might ____________________________________
  - Too small, we might ____________________________________
  - Industry standard: optimize learning rate or use adaptive learning rate

The "Learning Curve"



- Number of iterations on the x-axis
- Objective value on the y-axis
- Helps visualize and compare performance of algorithms

Gradient Descent for Least Squares

- The least squares objective and gradient:

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

- Gradient descent iterations for least squares:

$$w^{t+1} = w^t - \alpha^t \underbrace{X^T(Xw^t - y)}_{\nabla f(w^t)}$$

- Cost of gradient descent iteration is O(__) (no need to form X^TX).

Bottleneck is computing $\nabla f(w^t) = \underbrace{X^T\underbrace{(\underbrace{Xw^t}_{O(nd)} - y)}_{O(n)}}_{O(nd)}$

Normal Equations vs. Gradient Descent

- Least squares via normal equations vs. gradient descent:
  - Normal equations cost $O(nd^2 + d^3)$.
  - Gradient descent costs O(___) to run for 't' iterations.
    - Each of the 't' iterations costs O(nd).
  - Normal equations only solve linear least squares problems.
    - Gradient descent solves many other problems.

Beyond Gradient Descent

- Gradient descent can be faster when 'd' is very large:
  - If solution is "good enough" for a 't' less than minimum(d,d^2/n).
  - Proportional to "condition number" of X^TX (no direct 'd' dependence).

- There are many variations on gradient descent.
  - Methods employing a "line search" to choose the step-size.
  - "Conjugate" gradient and "accelerated" gradient methods.
  - Newton's method (which uses second derivatives).
  - Quasi-Newton and Hessian-free Newton methods.
  - Stochastic gradient (later in course).

- This course focuses on gradient descent and stochastic gradient:
  - They're simple and give reasonable solutions to most ML problems.
  - But the above can be faster for some applications.

Summary

- Least Squares: 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.
- Gradient descent:
  - Find a local minimum using gradients to navigate parameter space

- Next time: the bane of existence for gradient-based methods

Review Questions

- Q1: What is the dimensionality of the parameter space when we add a y-intercept to linear regression?

- Q2: Why can gradient descent only find local minima?

- Q3: In what situation is gradient descent the best choice for optimization, even when 'd' is small?

- Q4: Given training data, how can we tune the learning rate?

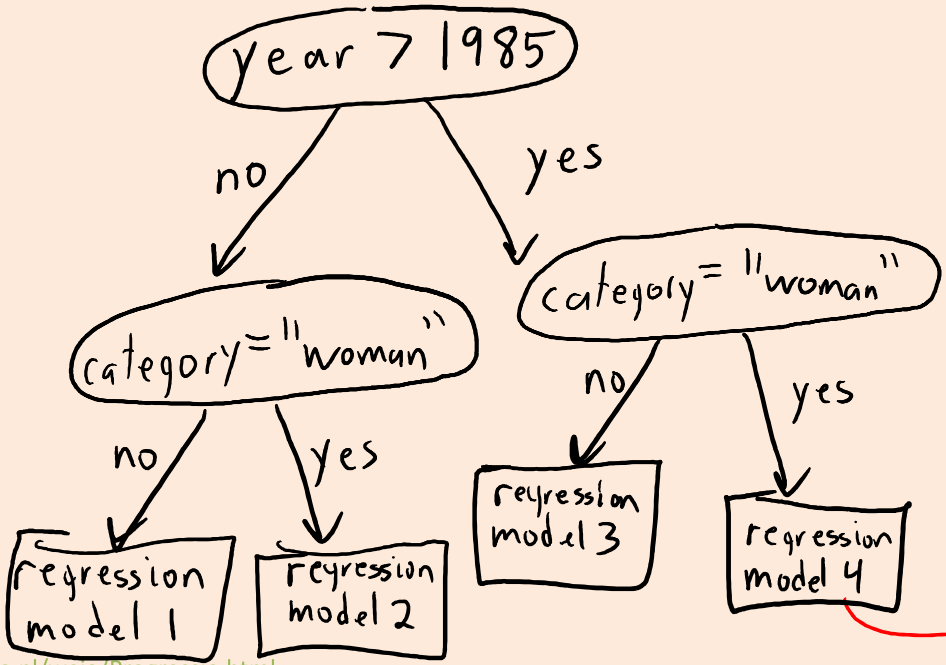
Adapting Counting/Distance-Based Methods

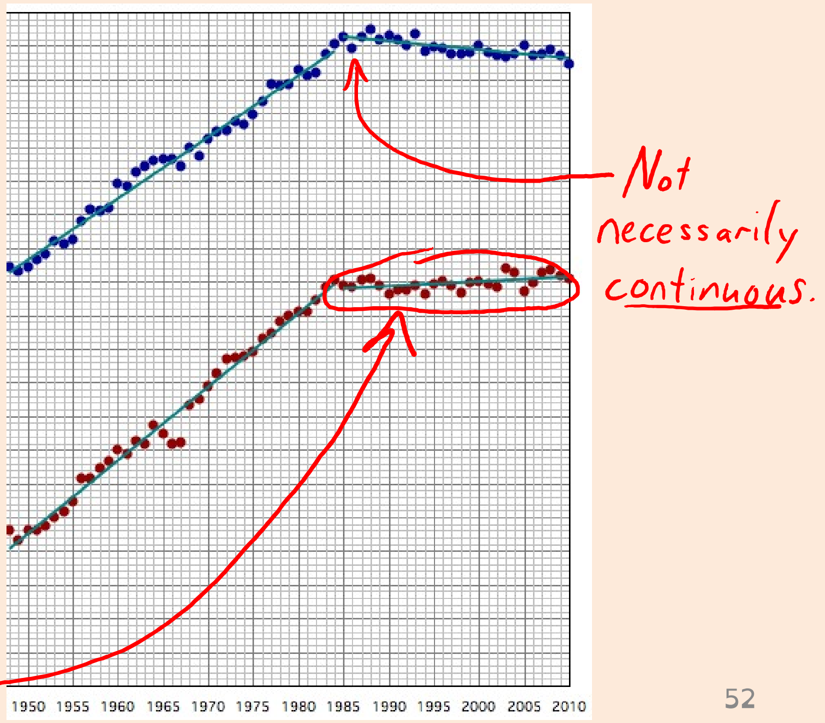
- We can adapt our classification methods to perform regression:

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.

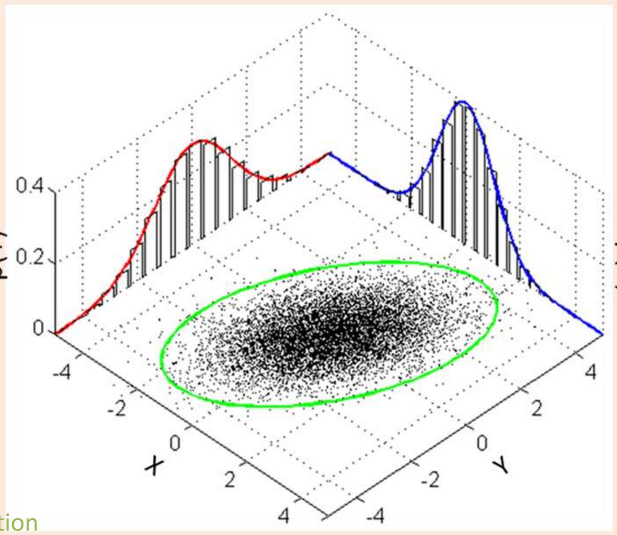




http://www.at-a-lanta.nl/weia/Progressie.html

Adapting Counting/Distance-Based Methods

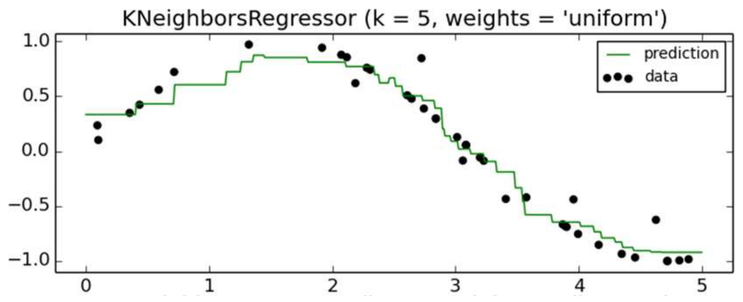
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
    - Take CPSC 440/540.



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

Adapting Counting/Distance-Based Methods

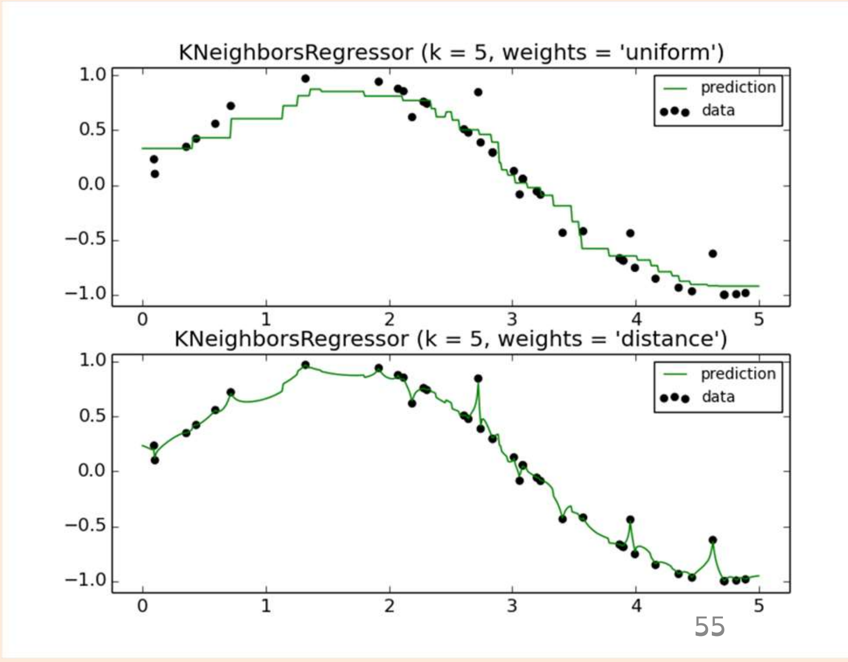
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression:
      - Find 'k' nearest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

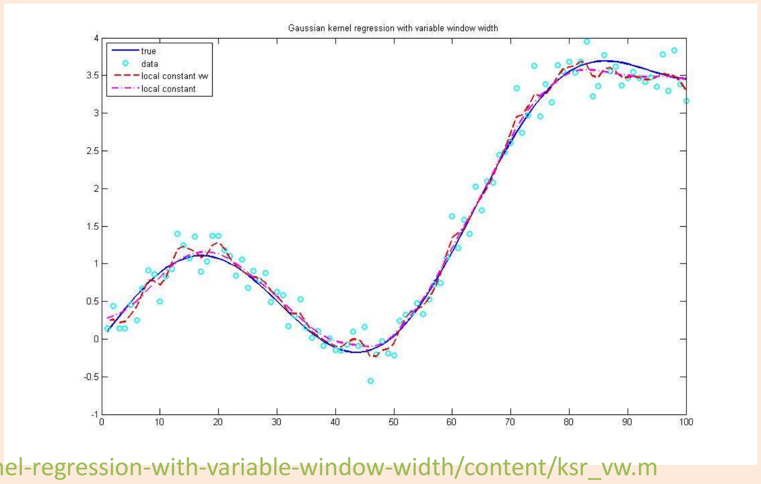


http://scikit-learn.org/stable/modules/neighbors.html

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.

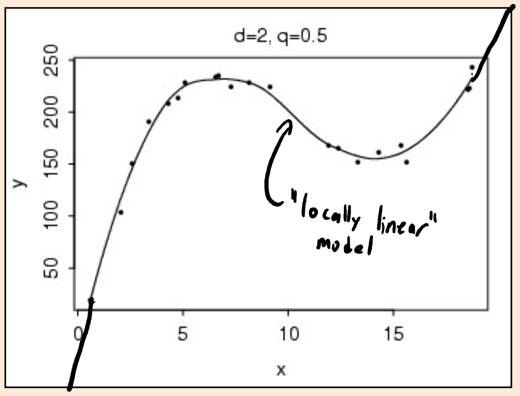
$$\hat{y}_i = \frac{\sum_{j=1}^{n} v_{ij} y_j}{\sum_{j=1}^{n} v_{ij}}$$



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

Adapting Counting/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.)



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 predictions 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".
    - See also "PatchMatch": https://vimeo.com/5024379
  - 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

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

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