CSC480/580: Principles of Machine Learning

Linear Models

Jason Pacheco

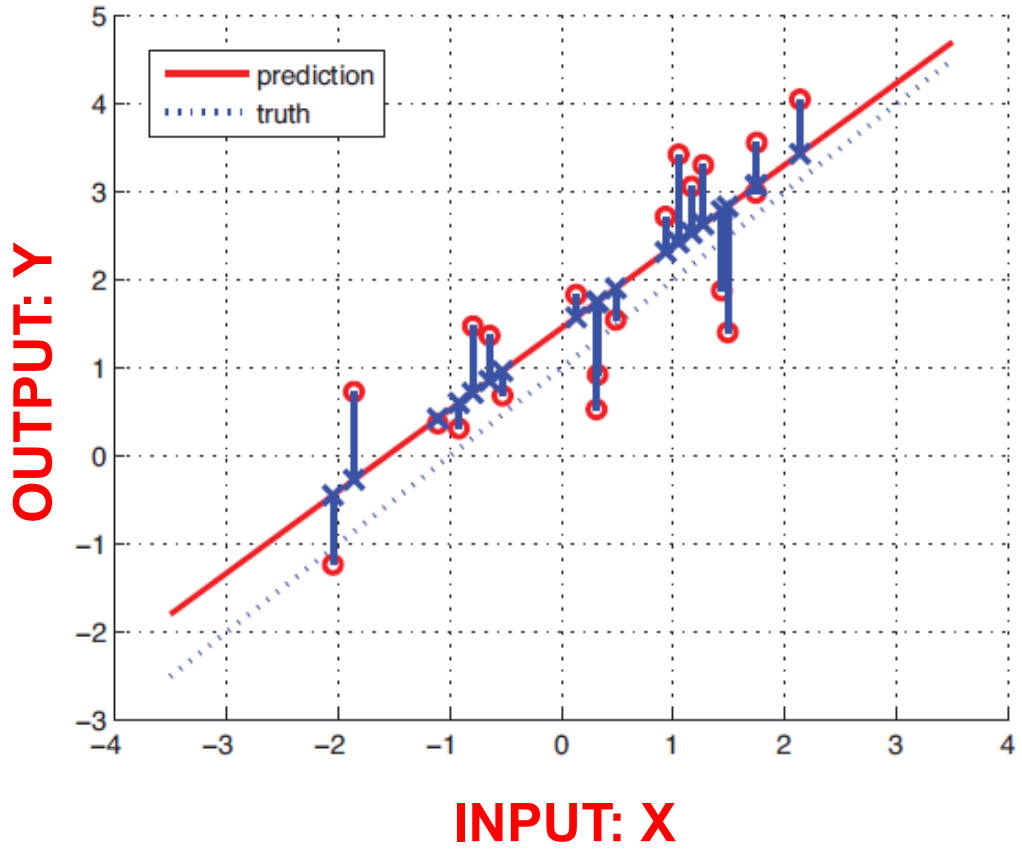
Outline

- Linear Models for Regression
  - Least Squares Estimation
  - Regularized Least Squares
- Linear Models for Classification
  - Logistic Regression

Outline

- Linear Models for Regression
  - Least Squares Estimation
  - Regularized Least Squares
- Linear Models for Classification
  - Logistic Regression

Linear Regression



Regression Learn a function that predicts outputs from inputs,

$$y = f(x)$$

Outputs y are real-valued

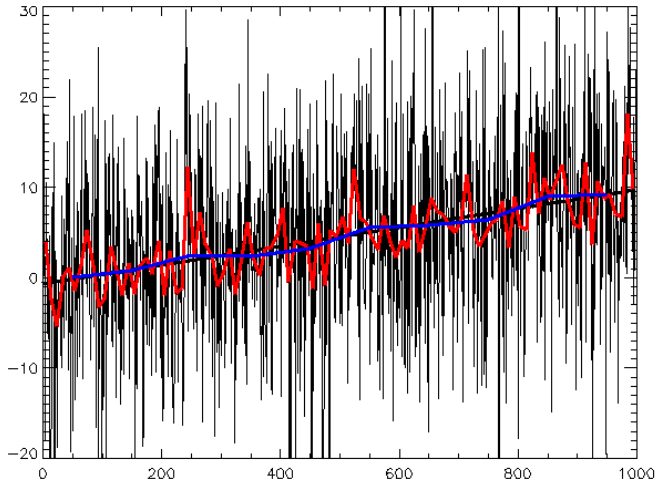
Linear Regression As the name suggests, uses a *linear function*:

$$y = w^T x + b$$

We allow noise in data… So the predictor may not be perfect

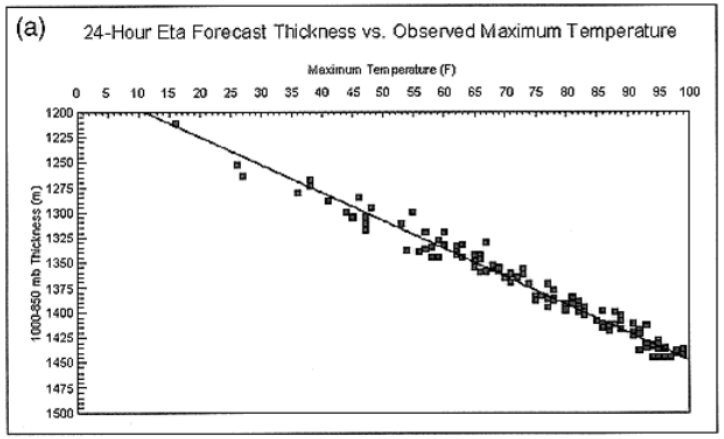
Linear Regression

Where is linear regression useful?



Trendlines

Stock Prediction

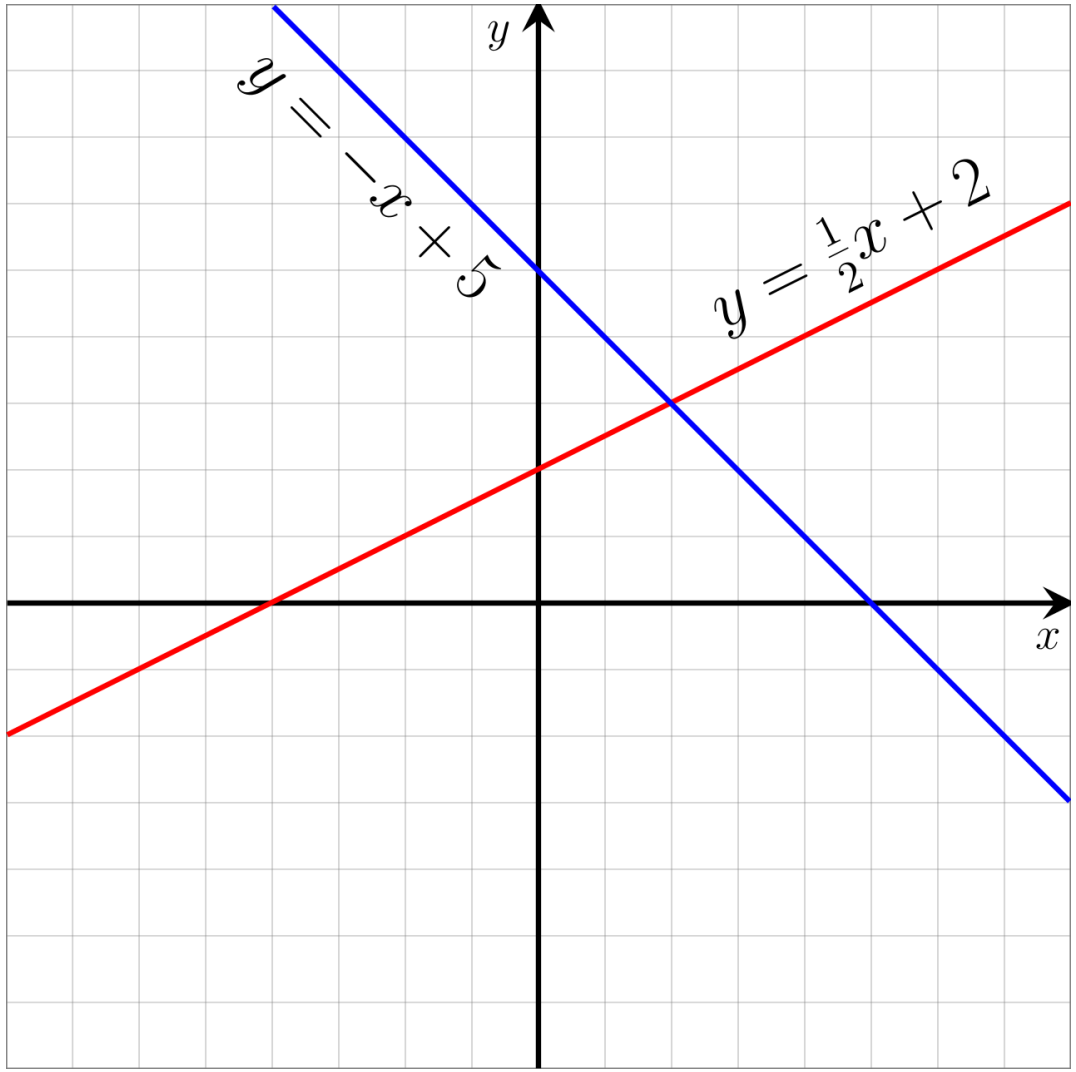


Climate Models

Massie and Rose (1997)

Used anywhere a linear relationship is assumed between continuous inputs / outputs

Line Equation



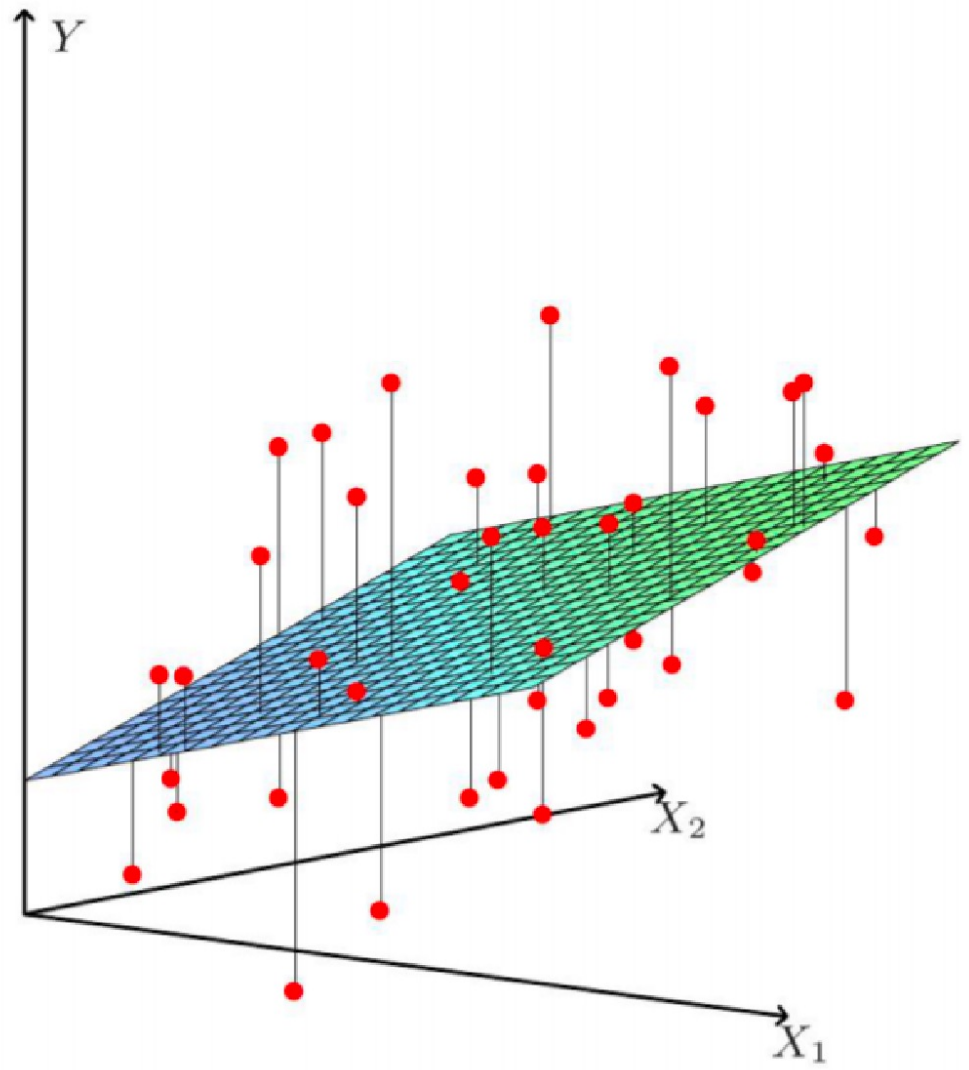
Recall the equation for a line has a *slope* and an *intercept*,

$$y = w \cdot x + b$$

Slope **Intercept**

- Intercept (b) indicates where line crosses y-axis
- Slope controls angle of line
- Positive slope (w) → Line goes up left-to-right
- Negative slope → Line goes down left-to-right

Linear regression in dimension >= 2



$$h(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + b = \langle w, x \rangle + b$$

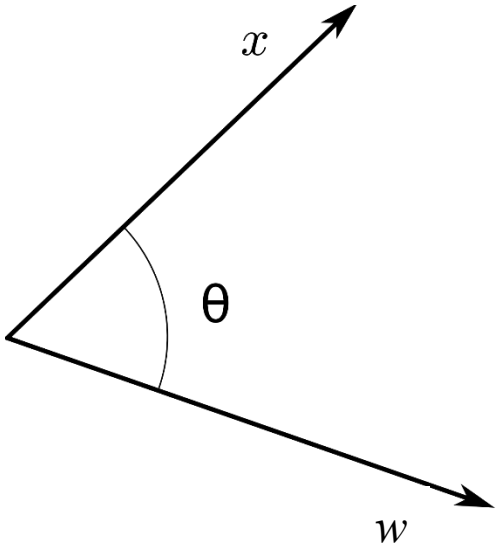
$y = \langle w, x \rangle + b$ can be viewed as a hyperplane

Inner Products

Recall the definition of an *inner product*:

$$w^T x = w_1 x_1 + w_2 x_2 + \ldots + w_D x_D$$

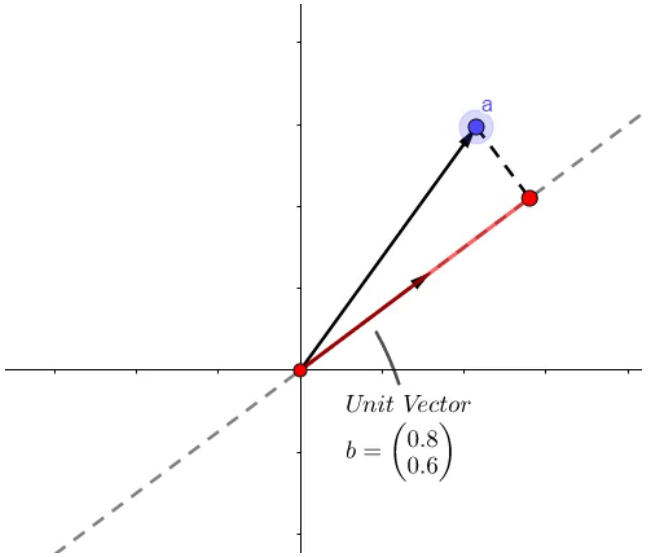
$$= \sum_{d=1}^{D} w_d x_d$$



Projection of one vector onto another,

$$w^T \hat{x} = |w| \cos\theta \qquad \text{where} \qquad \hat{x} = \frac{x}{|x|} = \frac{x}{\sqrt{\sum_d x_d^2}}$$

Unit Vector



Linear Regression

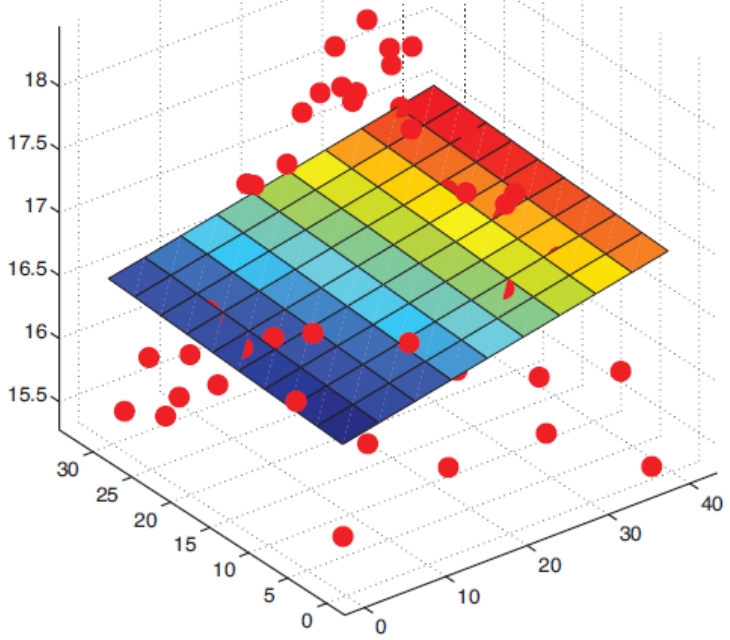
For D-dimensional input vector $x \in \mathbb{R}^D$ the plane equation,

$$y = w^T x + b$$

Sometimes we simplify this by including the intercept into the weight vector,

$$\widetilde{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_D \\ b \end{pmatrix} \qquad \widetilde{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{pmatrix} \qquad y = \widetilde{w}^T \widetilde{x}$$

[Image: Murphy, K. (2012)]



Since:

$$\widetilde{w}^T \widetilde{x} = \sum_{d=1}^{D} w_d x_d + b \cdot 1$$

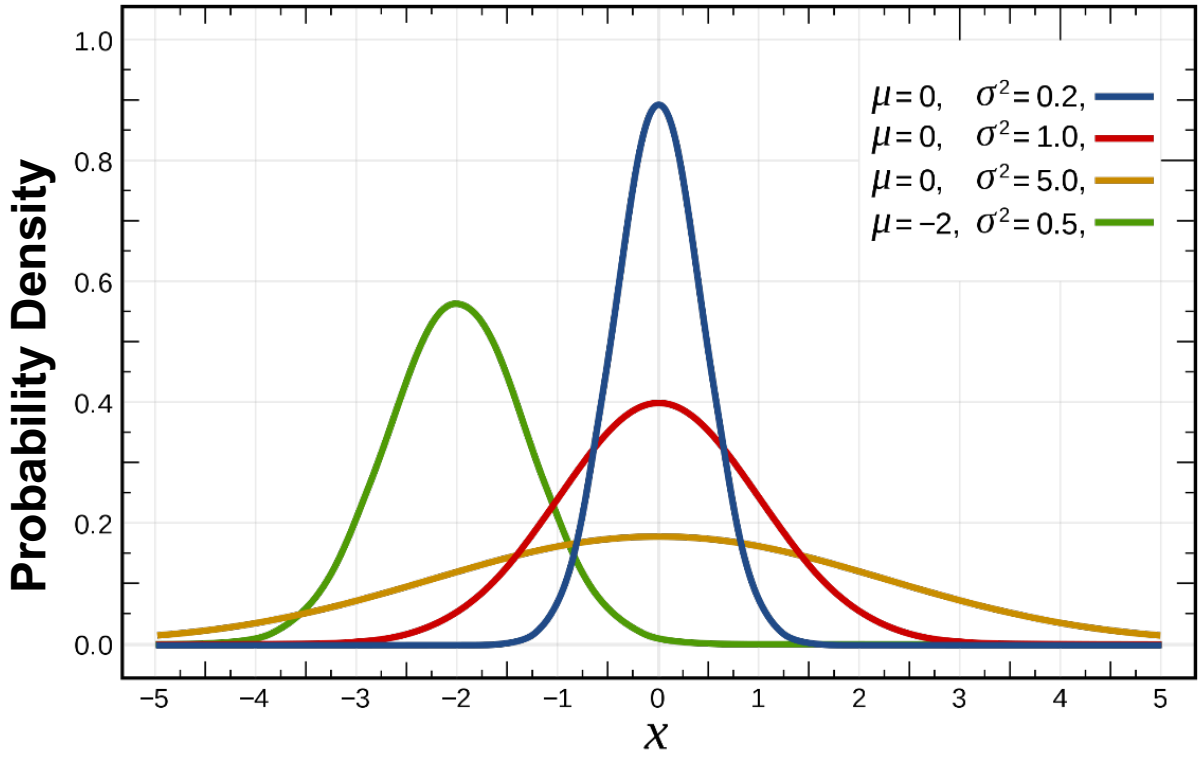
$$= w^T x + b$$

Modeling Noise in Data

Gaussian (a.k.a. Normal) distribution with mean (location) μ and variance (scale) σ^2 parameters,

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2}(x-\mu)^2/\sigma^2$$

We say $X \sim \mathcal{N}(X \mid \mu, \sigma^2)$

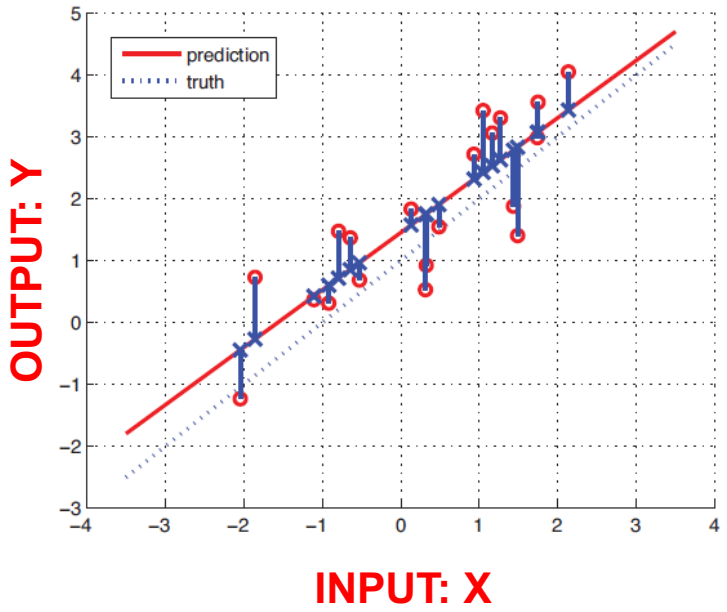


Linear Regression

Input-output mapping is not exact, so we will assume data has zero-mean Gaussian noise,

Independent noise for all examples

$$y = w^T x + \epsilon \quad \text{where}$$



This is equivalent to:

$$p(y \mid w, x) = \mathcal{N}(y \mid w^T x, \sigma^2)$$

Because Adding a constant to a Normal RV is still a Normal RV,

$$z \sim \mathcal{N}(m, P) \qquad z + c \sim \mathcal{N}(m + c, P)$$

In the case of linear regression $z \to \epsilon$ and $c \to w^T x$

Learning linear regression models

We need to learn the model from data
by learning the regression weights

Data – We have this

$$y = w^T x + \epsilon$$

Random; Can't do anything about it

Don't know these; need to learn them

How to do this?
What makes *good* weights?

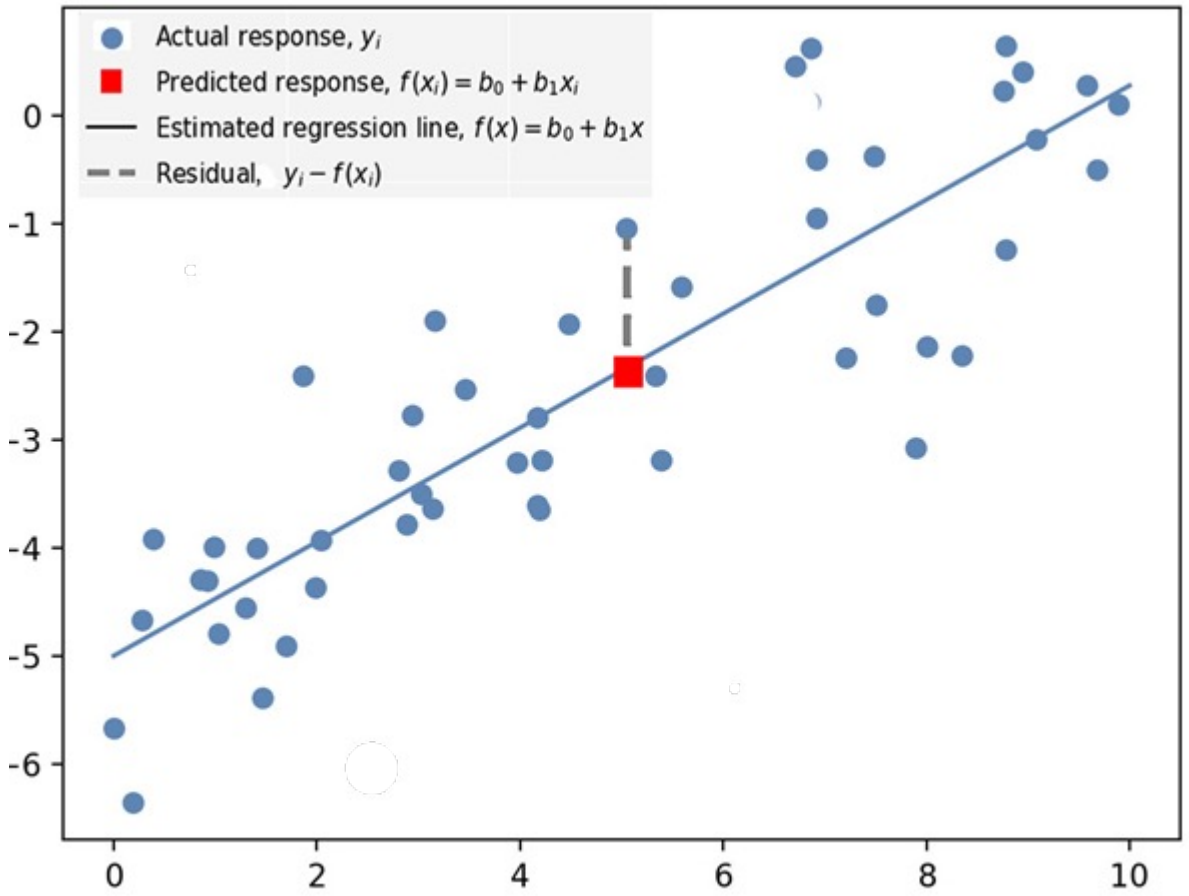
Learning Linear Regression Models

There are several ways to think about fitting regression:

- **Intuitive** Find a plane/line that is close to data
- **Functional** Find a line that minimizes the total *square error*
- **Estimation** Find maximum likelihood estimate of parameters

They are all equivalent…

Fitting Linear Regression



Intuition Find a line that is as *close as possible* to every training data point

The distance from each point to the line is the **residual**

$$y - w^T x$$

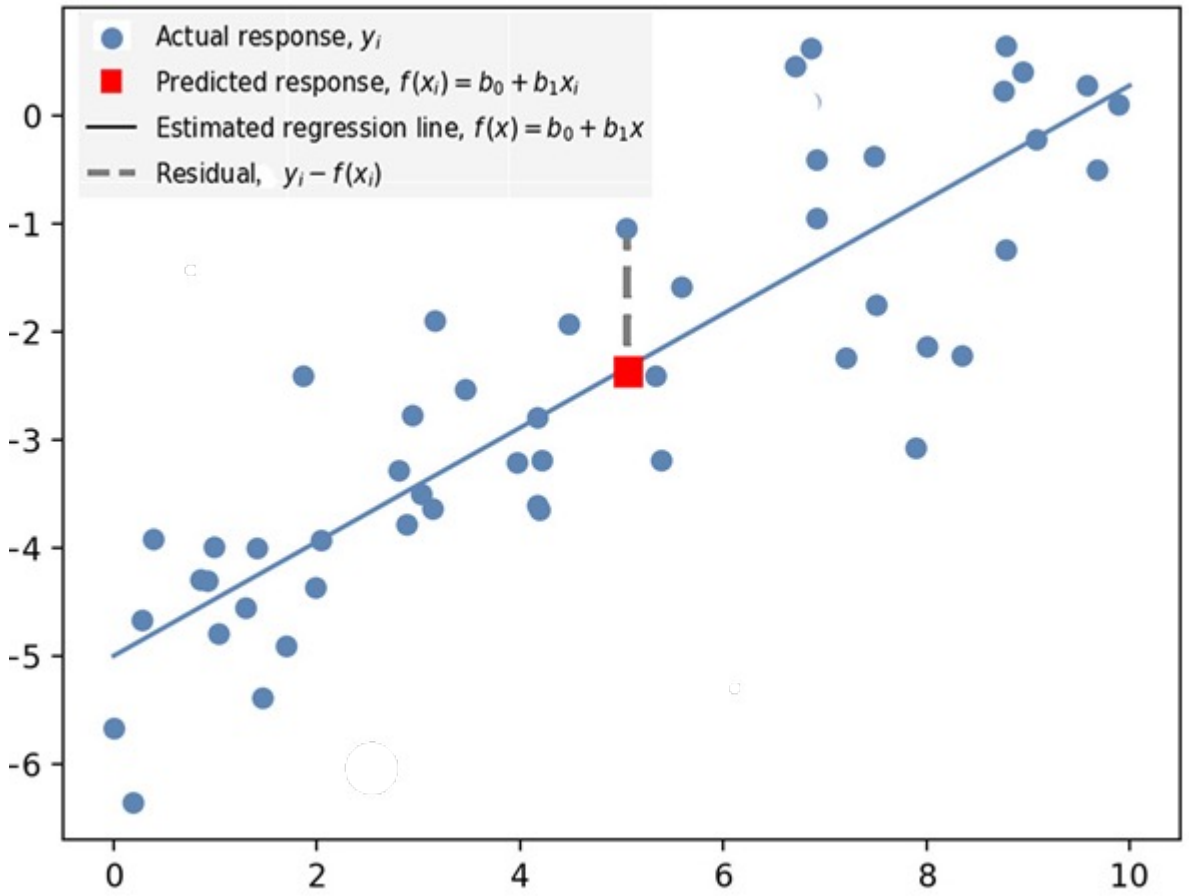
Training Output **Prediction**

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

Outline

- Linear Models for Regression
  - Least Squares Estimation
  - Regularized Least Squares
- Linear Models for Classification
  - Logistic Regression

Least Squares Solution



Functional Find a line that minimizes *square error*: the sum of squared residuals

$$w^* = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

Over all the training data,

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Least squares regression

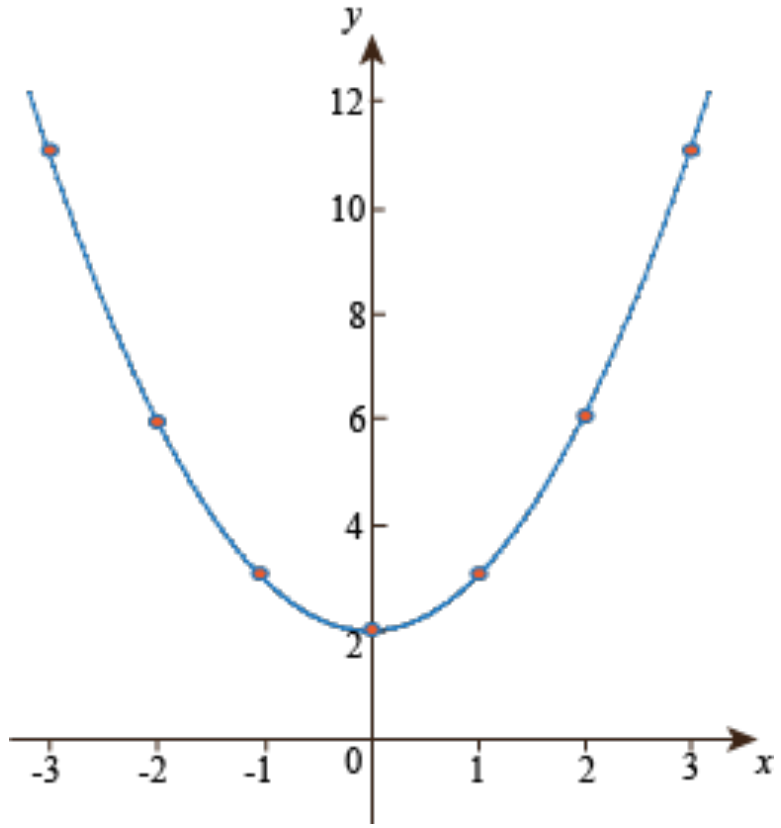
https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

Least Squares

$$\min_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

This is just a quadratic function…

- *Convex* => all local minima are global
- Minimum given by zero-derivative
- Can find a closed-form solution



Least Squares in Higher Dimensions

Things are a bit more complicated in higher dimensions and involve more linear algebra,

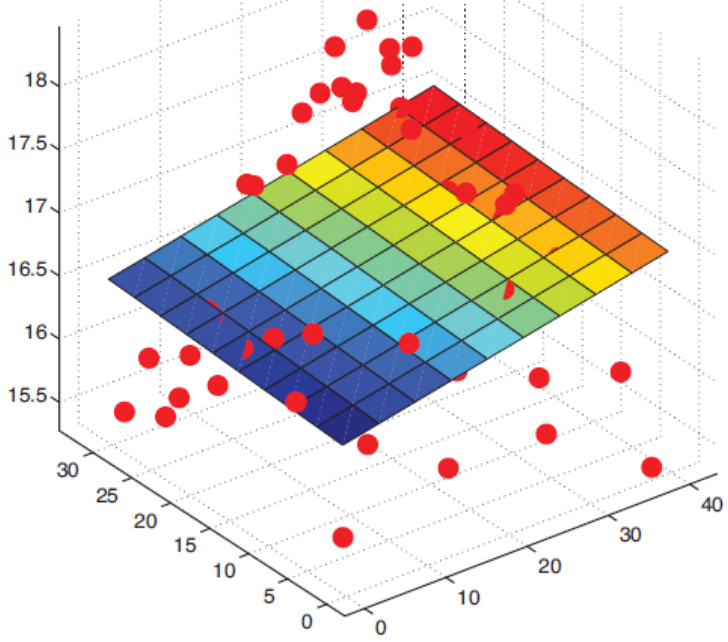
$$X = \begin{pmatrix} x_{11} & \cdots & x_{1D} \\ x_{21} & \cdots & x_{2D} \\ & \cdots & \\ x_{N1} & \cdots & x_{ND} \end{pmatrix}$$

Design Matrix
(each training input on a row)

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{pmatrix}$$

Vector of Training labels

[Image: Murphy, K. (2012)]



Can write regression over *all training data* more compactly…

$$y \approx Xw$$

← **Nx1 Vector**

Least Squares in Higher Dimensions

Least squares can also be written more compactly,

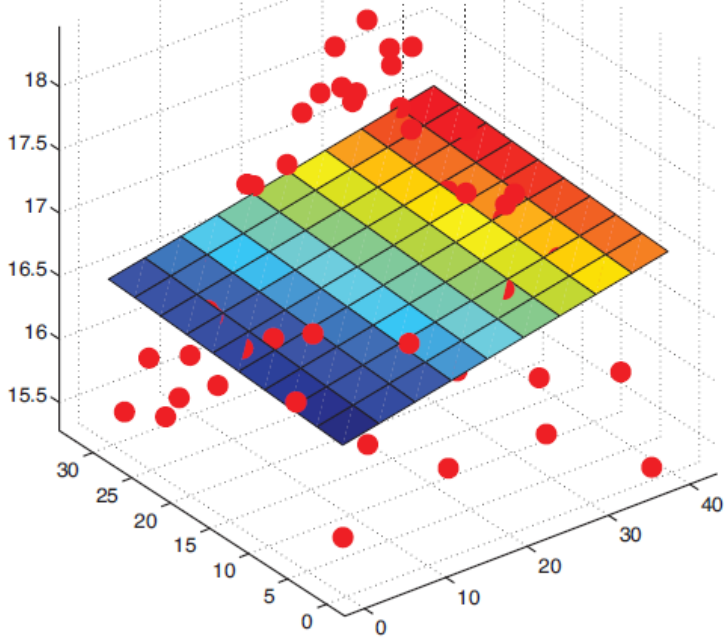
$$\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2 = \|\mathbf{y} - \mathbf{X}w\|^2$$

Some slightly more advanced linear algebra gives us a solution,

$$w = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Ordinary Least Squares (OLS) solution

[Image: Murphy, K. (2012)]



Derivation a bit involved for lecture but…

- We know it has a closed-form
- We can evaluate it

Jupyter notebook demo here

Learning Linear Regression Models

There are several ways to think about fitting regression:

- **Intuitive** Find a plane/line that is close to data
- **Functional** Find a line that minimizes the *least squares* loss
- **Estimation** Find maximum likelihood estimate of parameters

They are all the same thing…

Linear Regression Summary

Ordinary least squares solution

$$w^{\mathrm{OLS}} = \arg\min_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

can be calculated in closed-form:

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1D} \\ x_{21} & \cdots & x_{2D} \\ & \cdots & \\ x_{N1} & \cdots & x_{ND} \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{pmatrix} \qquad w^{\mathrm{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Design Matrix
(each training input on a column)

Vector of
Training labels

Jupyter notebook demo here

A word on matrix inverses…

$$w^{\mathrm{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Least squares solution requires inversion of the term,

$$(\mathbf{X}^T\mathbf{X})^{-1}$$

What are some issues with this?

1. Requires $\mathcal{O}(D^3)$ time for D input features

2. May be numerically unstable (or even non-invertible)

$$(x+\epsilon)^{-1} = \frac{1}{x+\epsilon}$$

Small numerical errors in input can lead to large errors in solution

Pseudoinverse

The *Moore-Penrose pseudoinverse* is denoted as X^+

$$w^{\text{OLS}} = X^+ y$$

- Generalization of the standard matrix inverse
  - If $X^\top X$ is invertible, $X^+ = (X^\top X)^{-1} X^\top$
- Exists even for non-invertible $X^\top X$
- Directly computable in most libraries
- In Numpy it is: `linalg.pinv`

notebook

Linear regression: extensions

- What if we have multivariate labels $y \in \mathbb{R}^k$?
  - E.g. Predict from income, family size, region:
    - food expenditure
    - Housing expenditure
    - Transportation expenditure
    - Healthcare spending
  - Conceptually, can think about the prediction from x to y as k separate linear regression problems

Learning Linear Regression Models

There are several ways to think about fitting regression:

- **Intuitive** Find a plane/line that is close to data
- **Functional** Find a line that minimizes the *least squares* loss
- **Estimation** Find maximum likelihood estimate of parameters

They are all the same thing…

Linear Regression: Probabilistic view

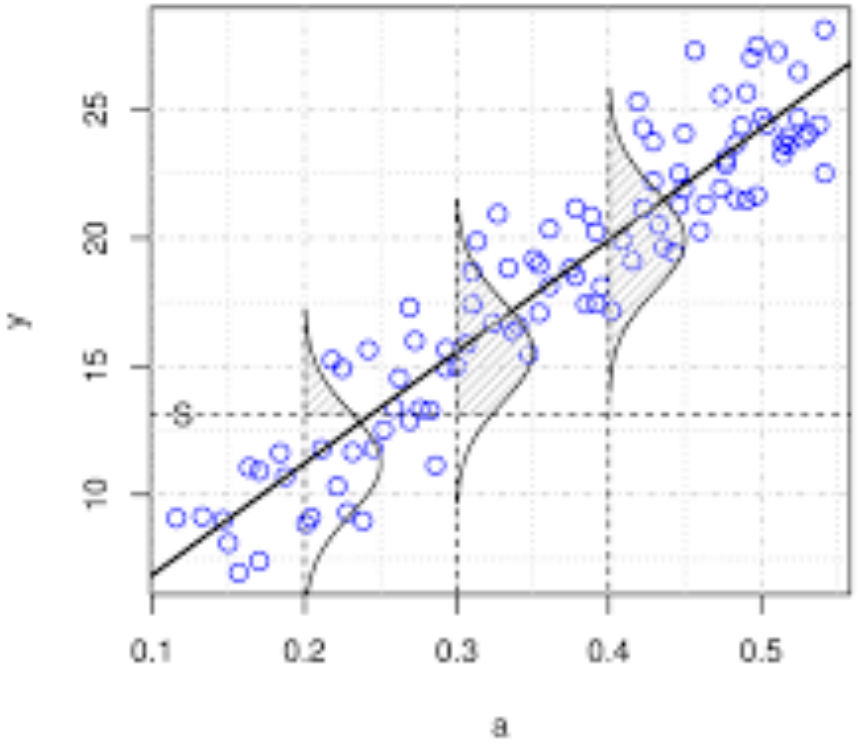
Definition of linear regression model,

$$y = w^T x + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Equivalent to:

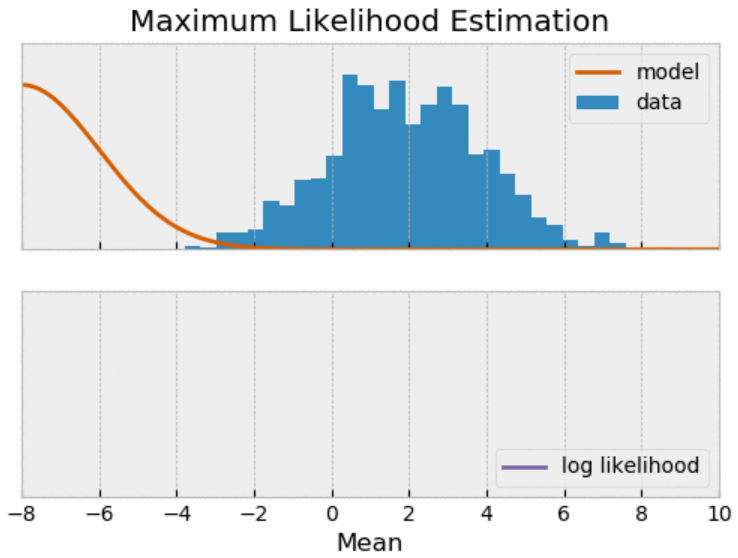
$$p(y \mid w, x) = \mathcal{N}(y \mid w^T x, \sigma^2)$$

Estimate w using maximum likelihood



Recap: Maximum Likelihood Estimation

Suppose we observe N data points from a Gaussian model $\mathcal{N}(\mu, \sigma^2)$ and wish to estimate its mean parameter μ



Likelihood Principle: *Given a statistical model, the likelihood function describes how well a parameter "supports" the observed data (evidence)*

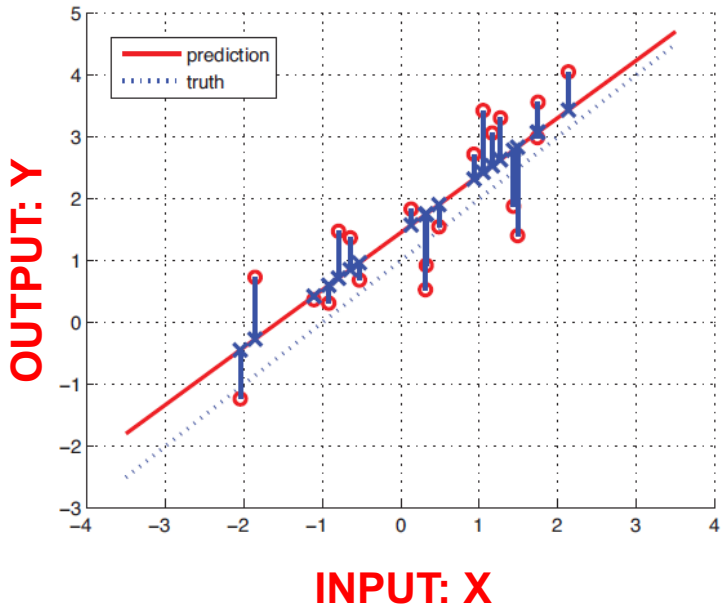
Fact: Assuming $p(y \mid w, x) = \mathcal{N}(y \mid w^T x, \sigma^2)$ for every example. The logarithm of the likelihood function = negative square error of w

https://towardsdatascience.com/maximum-likelihood-estimation-984af2dcfcac

MLE for Linear Regression

Given training data $\{(x_i, y_i)\}_{i=1}^N$ likelihood function is given by,

$$\log \prod_{i=1}^{N} p(y_i \mid x_i, w) = \sum_{i=1}^{N} \log p(y_i \mid x_i, w)$$



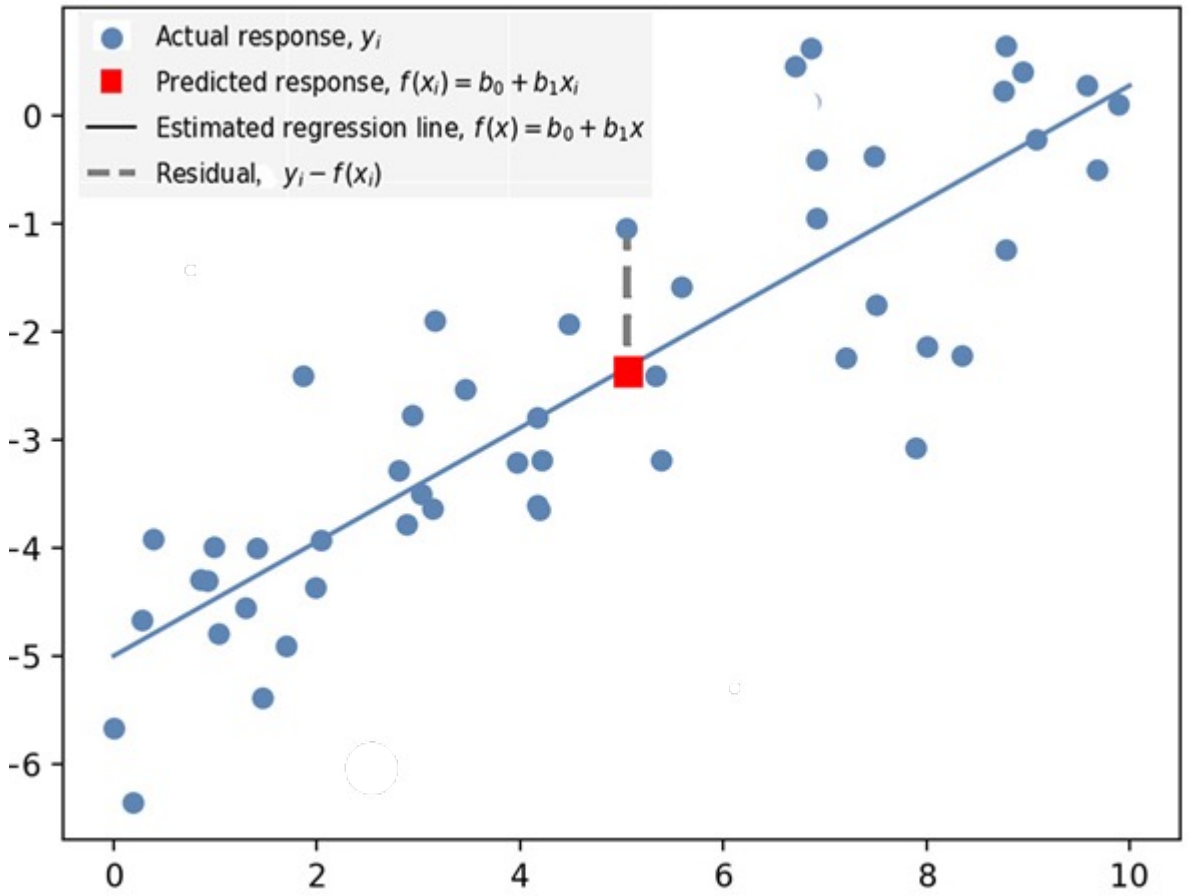
Recall that the likelihood is Gaussian:

$$p(y \mid w, x) = \mathcal{N}(y \mid w^T x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2}(y - w^T x)^2/\sigma^2$$

So MLE maximizes the log-likelihood over the whole data as,

$$w^{\mathrm{MLE}} = \arg\max_w \sum_{i=1}^{N} \log \mathcal{N}(y_i \mid w^T x_i, \sigma^2) = \arg\max_w \sum_{i=1}^{N} \mathrm{const} - \frac{1}{2\sigma^2}(y_i - w^T x_i)^2$$

MLE for Linear Regression



After simplification, we have,

$$w^{\mathrm{MLE}} = \min_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

So for Linear Regression, MLE = Least Squares Estimation

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

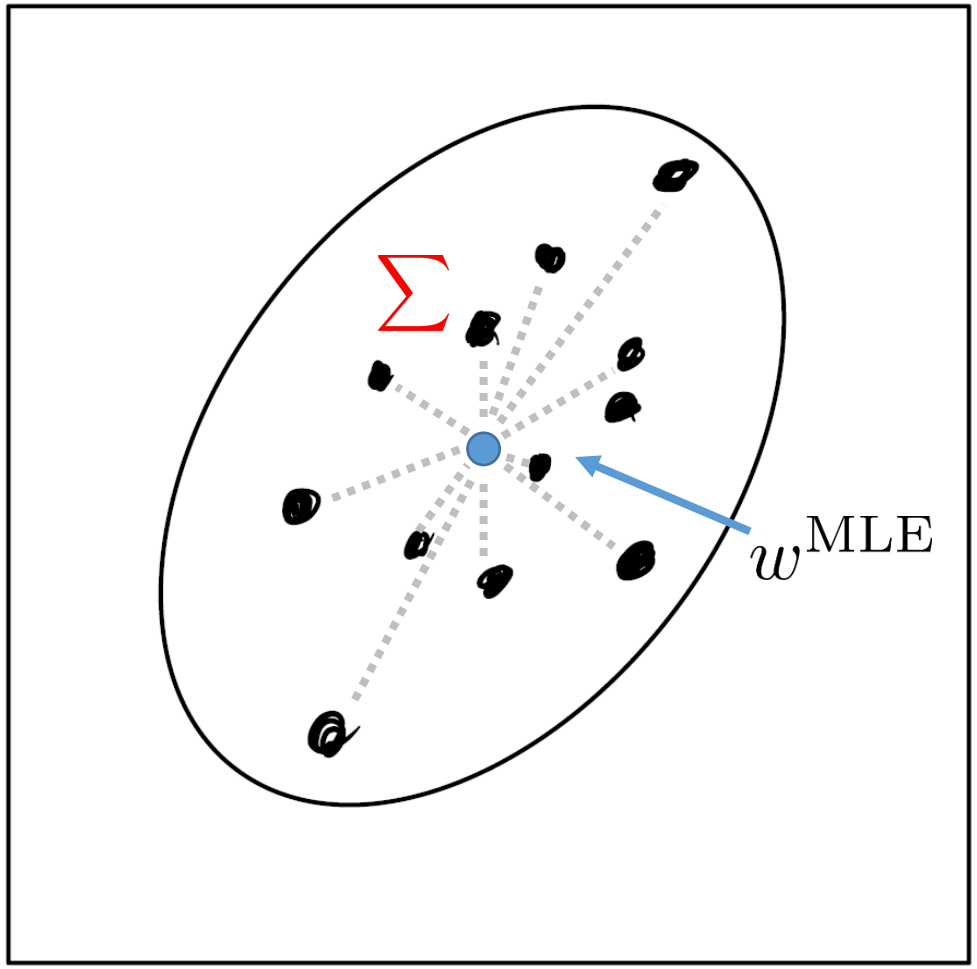
Outline

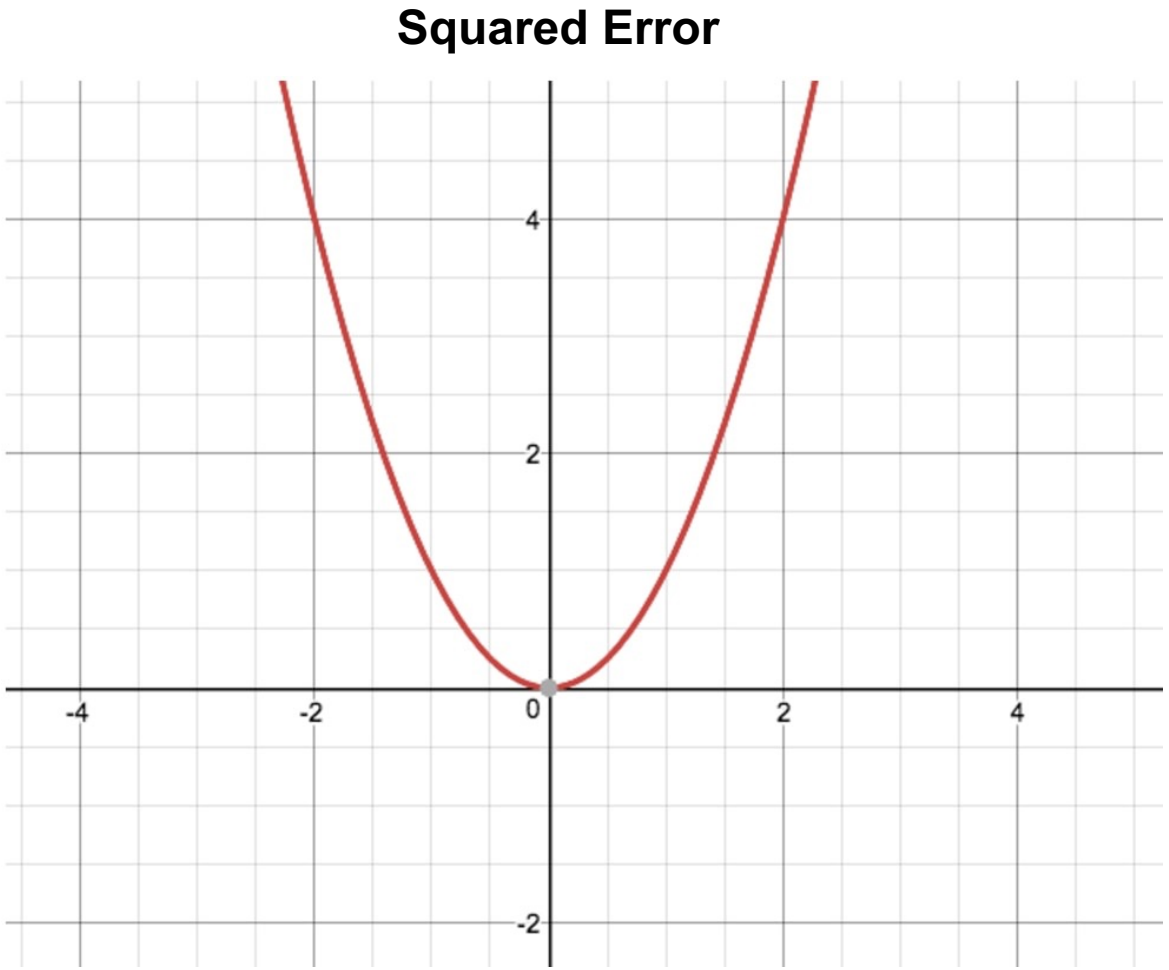
- Linear Models for Regression
  - Least Squares Estimation
  - Regularized Least Squares
- Linear Models for Classification
  - Logistic Regression

Outliers

How does an outlier affect the estimator?

Example: estimate the mean of a population

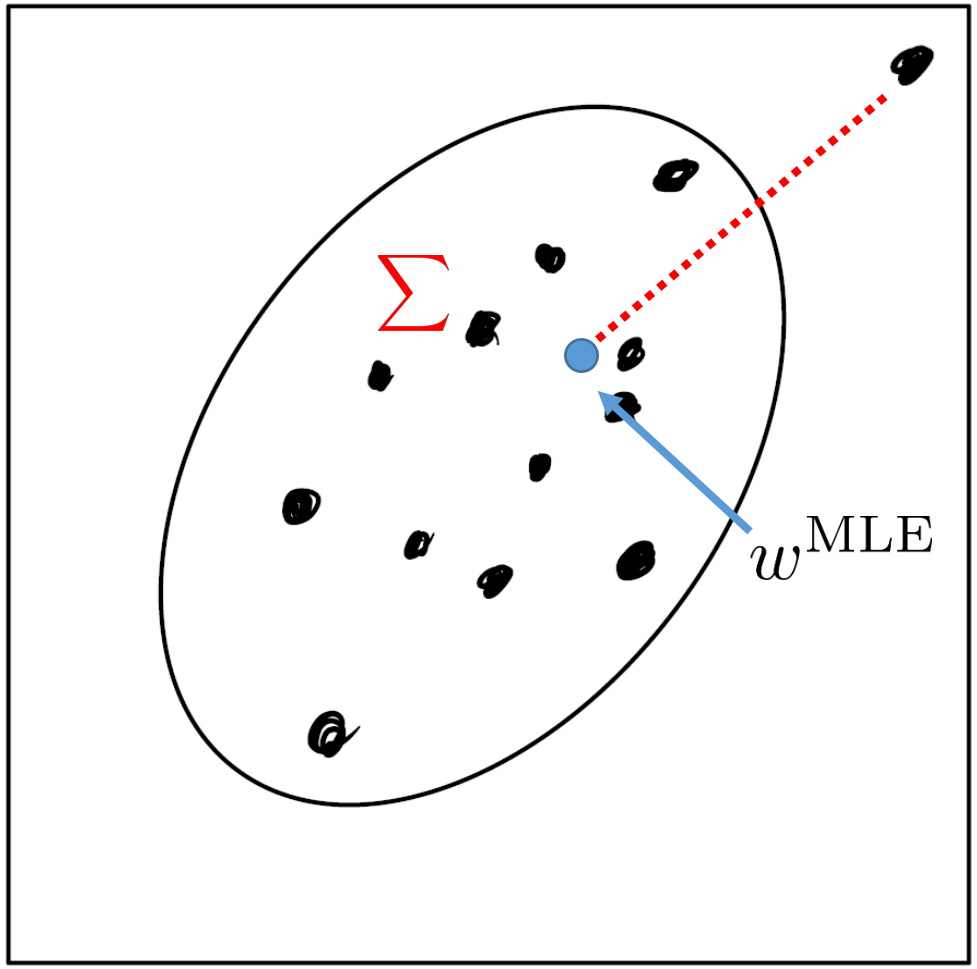


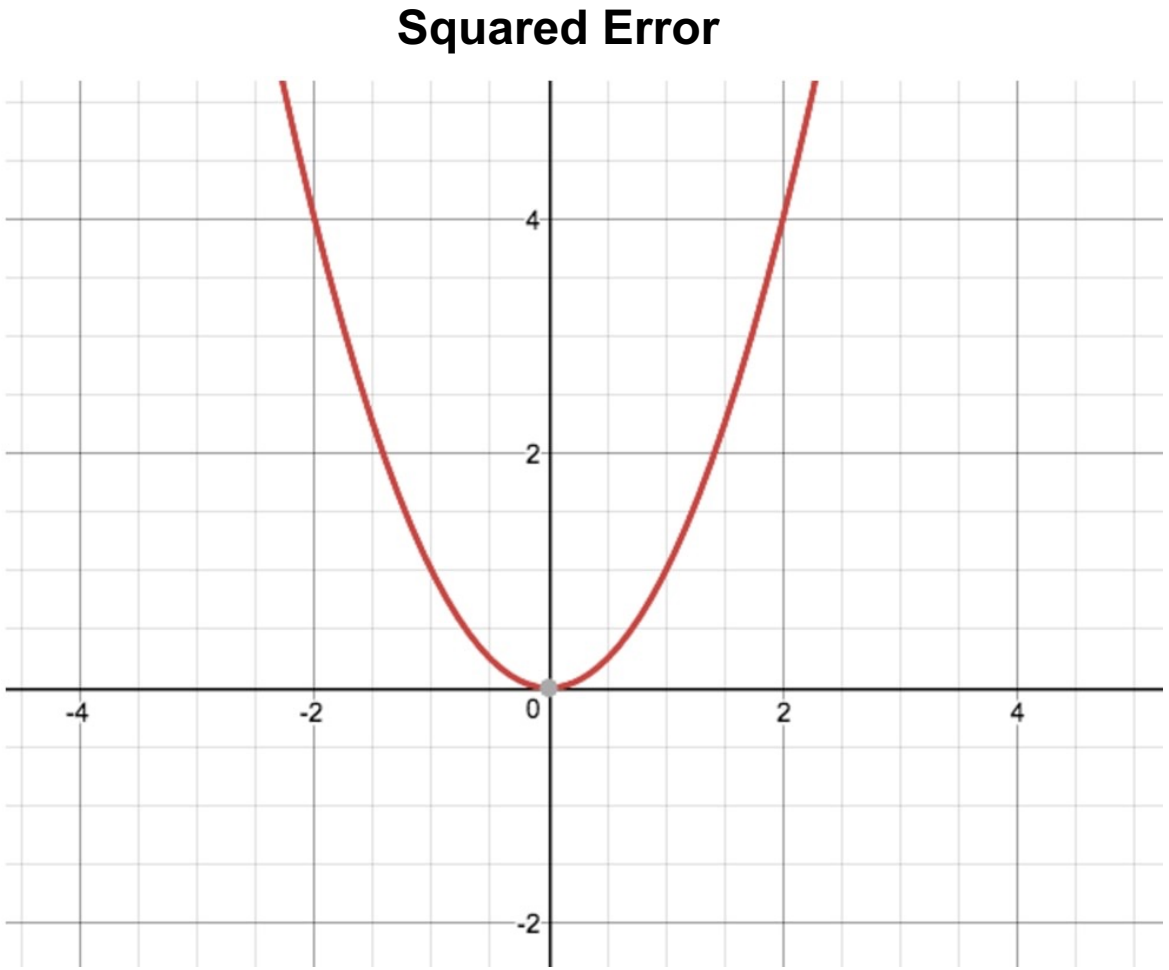


Outliers

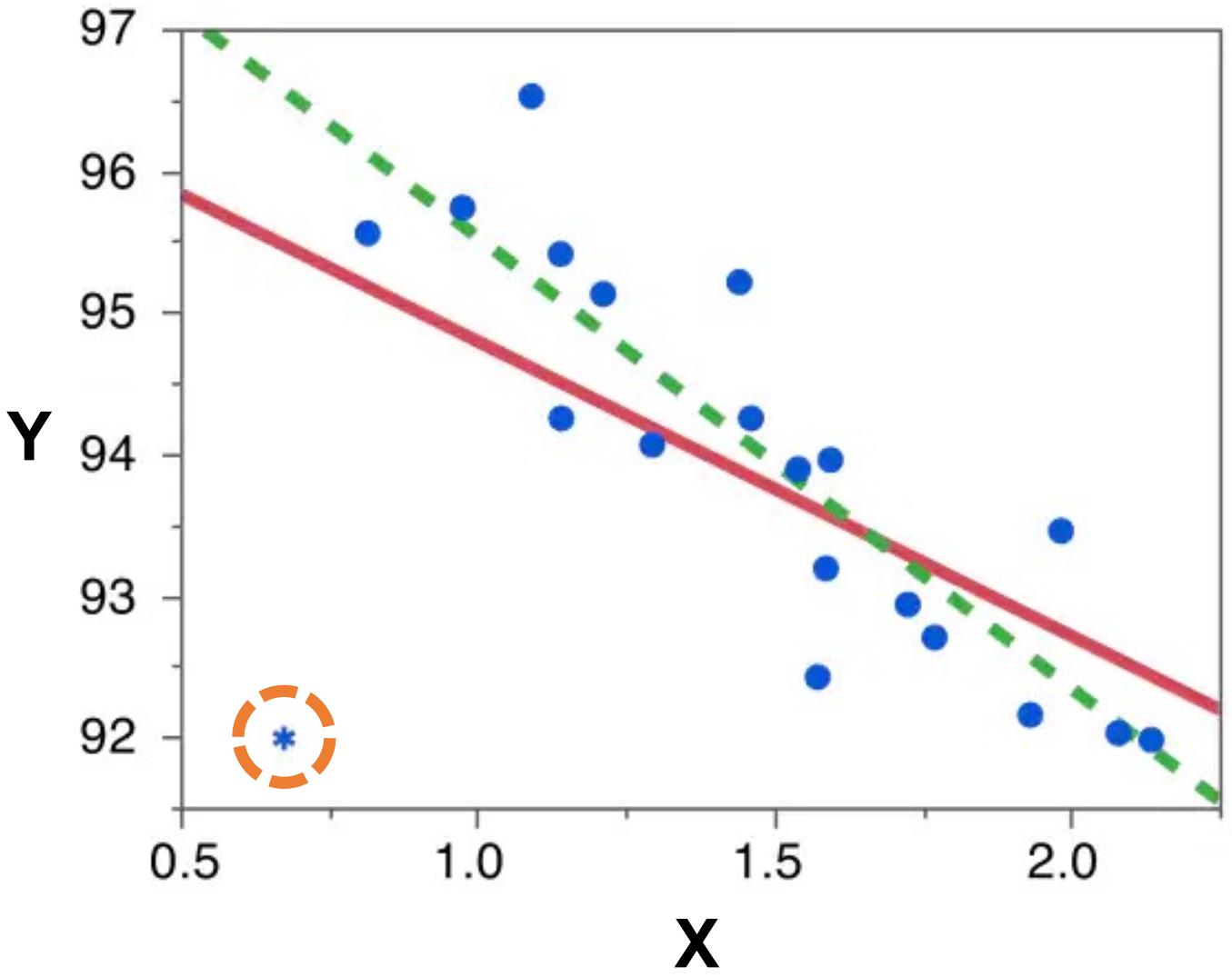
How does an outlier affect the estimator?

Example: estimate the mean of a population





Outliers in Linear Regression



Outlier “pulls” regression line away from inlier data

Need a way to *ignore* or to *down-weight* impact of outlier

https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-multiple-regression/mlr-residual-analysis-and-outliers.html

Dealing with Outliers

Too many outliers can indicate many things: non-Gaussian (heavy-tailed) data, corrupted data, bad data collection, …

A few ways to handle outliers…

1. Use a heavy-tailed distribution to model noise (e.g. Student's t)

Fitting regression becomes difficult

2. Identify outliers and discard them

Perhaps needs to solve integer programming (NP-Hard)

3. Penalize extreme weights to avoid overfitting (Regularization)

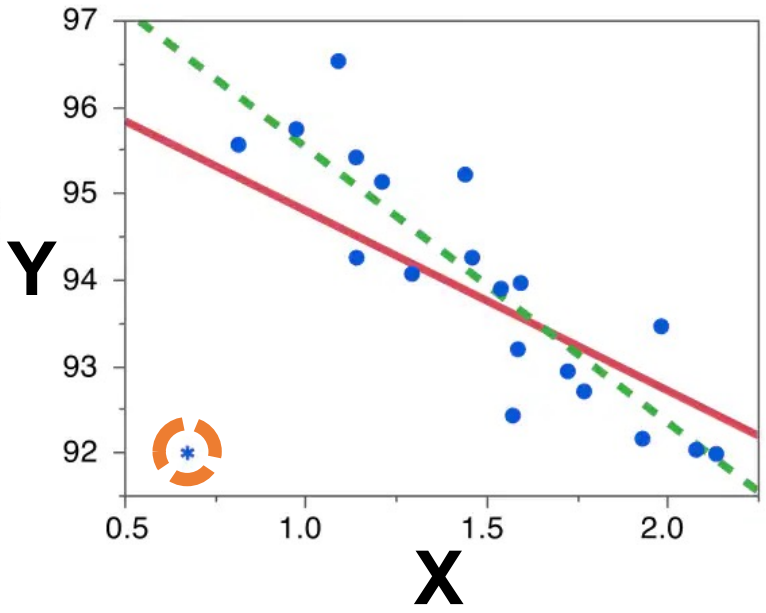
Regularization

Regularization helps avoid overfitting to training data…

$$\text{Model} = \min_{\text{model}} \text{Loss}(\text{Model}, \text{Data}) + \lambda \cdot \text{Regularizer}(\text{Model})$$

Regularization Strength

Regularization Penalty



Red model is without regularization

Green model is with regularization

Regularized Least Squares

A couple regularizers are so common they have specific names

L2 Regularized Linear Regression

- Ridge Regression
- aka Tikhonov Regularization

L1 Regularized Linear Regression

- LASSO
- Stands for “Least Absolute Shrinkage and Selection Operator”

Regularized Least Squares

Ordinary least-squares estimation (no regularizer),

Already know how to solve this…

$$w^{\mathrm{OLS}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

L2-regularized Least-Squares (Ridge)

Quadratic Penalty

$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|^2$$

L1-regularized Least-Squares (LASSO)

Absolute Value (L1) Penalty

$$w^{\mathrm{L1}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda|w|$$

A word on vector norms…

The *L2-norm* (Euclidean norm) of a vector w is,

$$\|w\| = \sqrt{w^T w} = \sqrt{\sum_{d=1}^{D} w_d^2} \qquad \|w\|^2 = \sum_{d=1}^{D} w_d^2$$

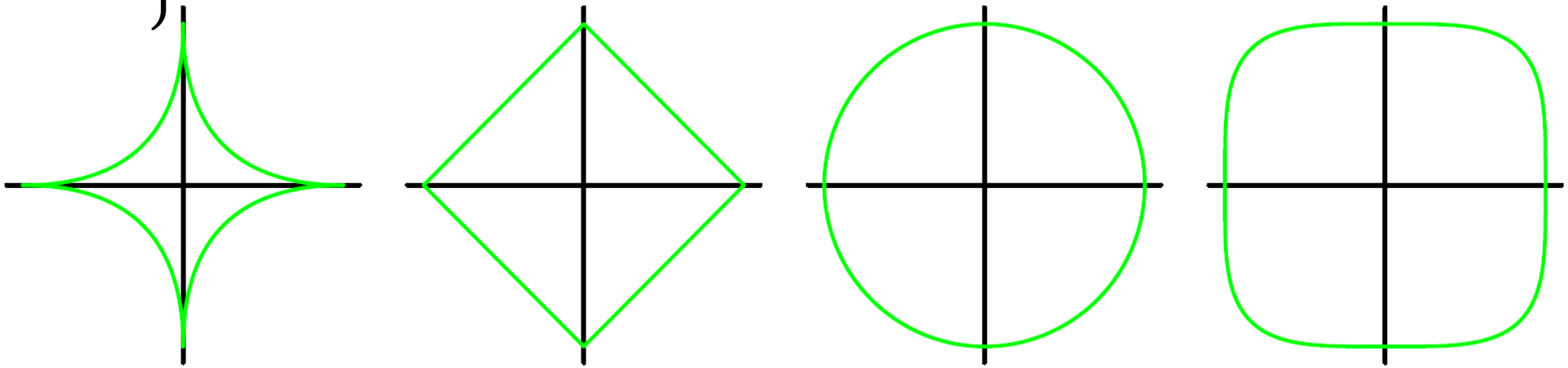
The *L1-norm* (absolute value) of a vector w is,

$$|w| = \sum_{d=1}^{D} |w_d|$$

They are not the same functions…

Other Regularization Terms

$$\left\{ w : \sum_{i=1}^{d} |w_i|^q \leq 1 \right\}$$



$q = 0.5$ $q = 1$ $q = 2$ $q = 4$

q<1 is not a norm, and thus not convex

L1 is non-differentiable

L2 Regularization

A more general regularization penalty:

$$\hat{w} = \arg\min_{w} \frac{1}{2} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{d} |w_i|^q$$

L2 Regularized Least Squares

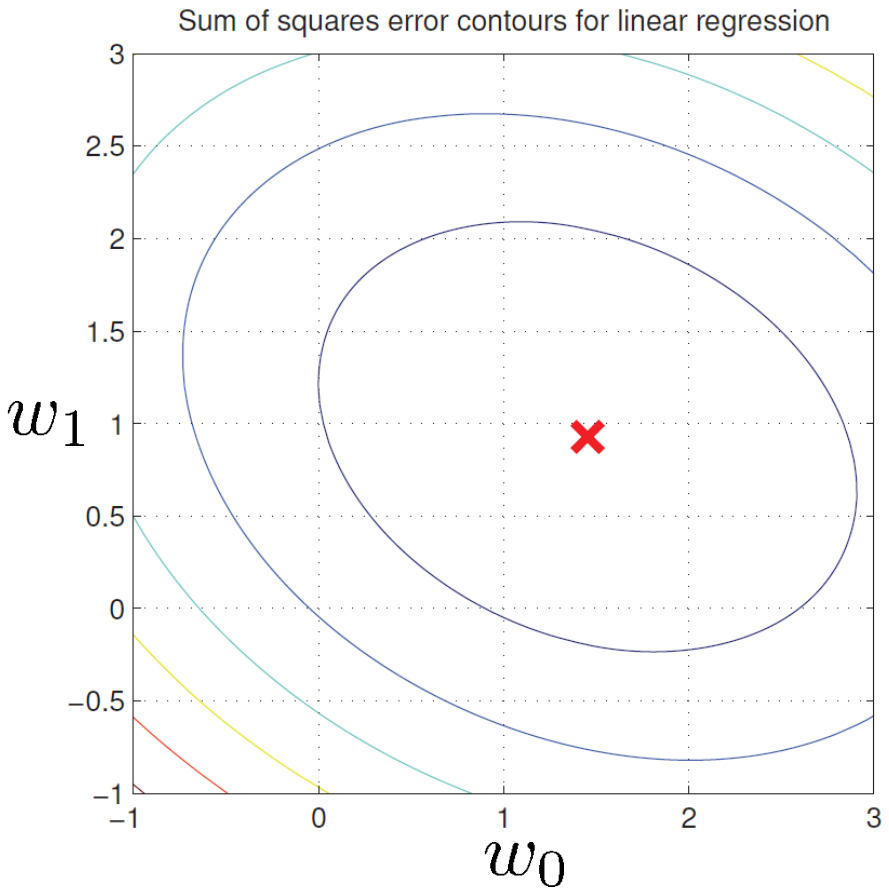
$$w^{\mathrm{L2}} = \arg\min_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|^2$$

Quadratic

Quadratic

Quadratic + Quadratic = Quadratic

- Differentiable
- Convex
- Unique optimum
- Closed form solution



L2 Regularized Least Squares: d=1 Case

$$\frac{d}{dw}\frac{1}{2}\sum_{i=1}^{N}(y_i - wx_i)^2 + \frac{\lambda}{2}\frac{d}{dw}w^2 =$$

Derivative (+ chain rule)

$$= \sum_{i=1}^{N}(y_i - wx_i)(-x_i) + \lambda w \quad = 0 \Rightarrow$$

Distributive Property

$$0 = \sum_{i=1}^{N} y_i x_i - w\sum_{j=1}^{N} x_j^2 - \lambda w$$

Algebra

$$w = \frac{\sum_i y_i x_i}{\lambda + \sum_j x_j^2}$$

L2 Regularized Linear Regression – Ridge Regression

Source: Kevin Murphy's Textbook

$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|^2$$

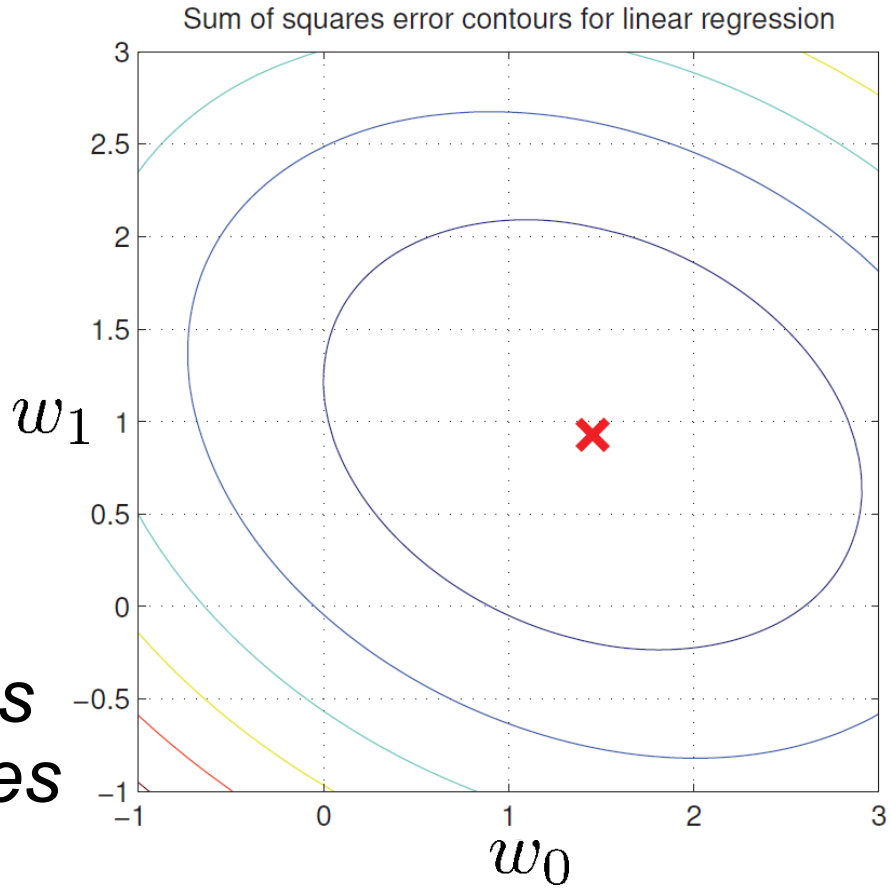
After some algebra…

$$w^{\mathrm{L2}} = (\lambda I + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Compare to ordinary least squares:

$$w^{\mathrm{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Regularized least-squares can be viewed as OLS with additional pseudo-training examples



Notes on L2 Regularization

- Feature weights are “shrunk” towards zero (and each other) – statisticians often call this a “shrinkage” method
- Typically do **not** penalize bias (y-intercept, w_0) parameter,

$$\min_w \sum_i (y_i - w^T x_i - w_0)^2 + \lambda \sum_{d=1}^{D} w_d^2$$

- This way the solution will be invariant to data shifting

- Solutions are **not** invariant to scaling, so typically we standardize features (recall Lec. 5) before fitting model (Sklearn `StandardScaler`)

Least squares solution with general regularization (580 only)

$$\min_{w} \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda ||w||_A^2$$

Here, $||w||_A^2 = w^T A\, w$

Example $A = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \cdots & 1 \end{pmatrix} \Rightarrow ||w||_A^2 = ||w||_2^2 \Rightarrow$ ridge regression

$A = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \cdots & 1 \end{pmatrix} \Rightarrow ||w||_A^2 = \sum_{i=2}^{d} w_i^2 \Rightarrow$ do not penalize w_1

Least squares solution with general regularization (580 only)

Fact The solution to $\min_w \sum_{i=1}^N (w^T x_i - y_i)^2 + \lambda ||w||_A^2$ is

$$(\lambda A + X^T X)^{-1} X^T y$$

Justification: The objective function is

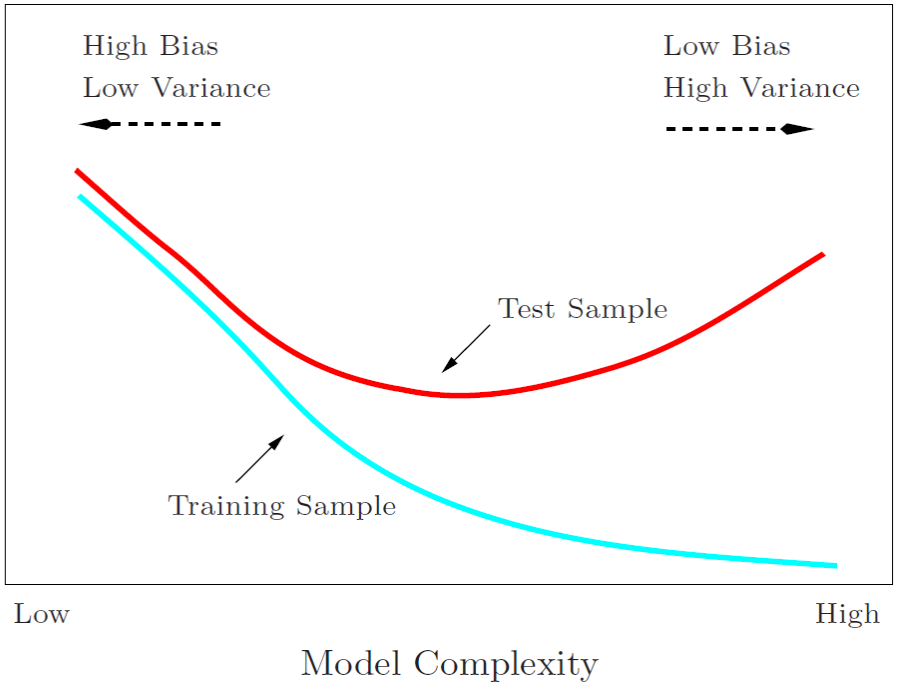
$$R(w) = w^T(\lambda A + X^T X)w - 2w^T X^T y + y^T y$$

The minimizer is stationary: $\nabla R(w) = 2(\lambda A + X^T X)w - 2X^T y = 0$

Choosing Regularization Strength

We need to tune regularization strength to avoid over/under fitting…

$$w^{\mathrm{L2}} = \arg\min_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|^2$$

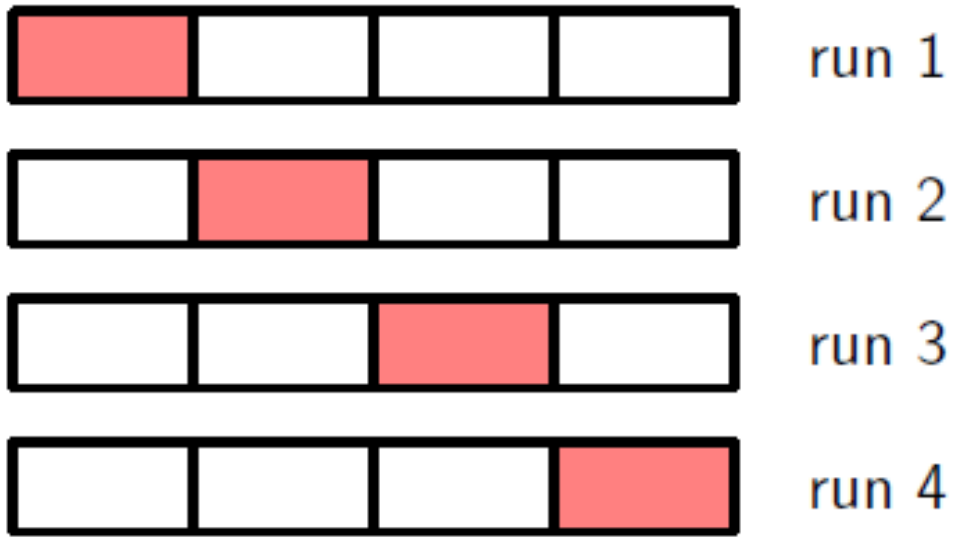


Recall bias/variance tradeoff
Error = Bias^2 + Variance

High regularization *reduces* model complexity: *increases* bias / *decreases* variance

How should we properly tune λ?

Cross-Validation



N-fold Cross Validation Partition training data into N "chunks" and for each run select one chunk to be validation data

For each run, fit to training data (N-1 chunks) and measure accuracy on validation set. Average model error across all runs.

Drawback Need to perform training N times.

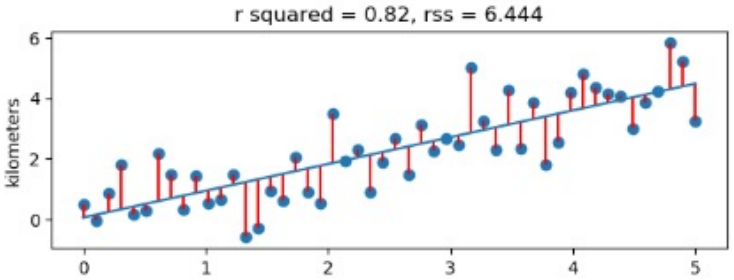
Source: Bishop, C. PRML

Model Selection for Linear Regression

A couple of common metrics for model selection…

Residual Sum-of-squared Errors The total squared residual error on the held-out validation set,

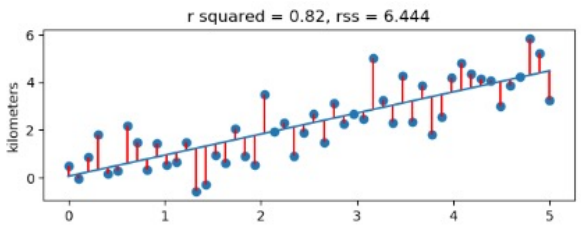
$$\mathrm{RSS} = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$



Coefficient of Determination Also called R-squared or R^2. Fraction of variation explained by the model.

Model selection metrics are known as "goodness of fit" measures

Coefficient of Determination R^2



Variance unexplained by Regression model

Residual Sum-of-Squares

$$R^2 = 1 - \frac{\text{RSS}}{\text{SS}} = 1 - \frac{\sum_{i=1}^{N}(y_i - w^T x_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

Total variance in dataset

Variance using avg. prediction

Where: $\bar{y} = \frac{1}{N}\sum_i y_i$ is the average output

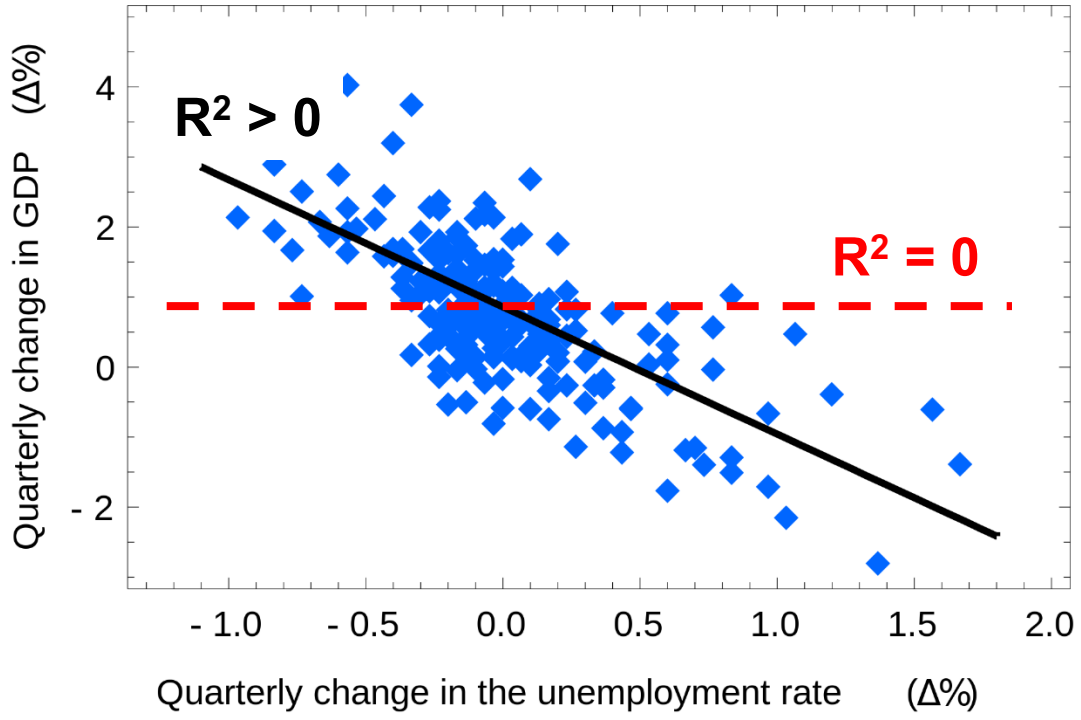
Coefficient of Determination R^2

$$R^2 = 1 - \frac{\text{RSS}}{\text{SS}} = 1 - \frac{\sum_{i=1}^{N}(y_i - w^T x_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

Maximum value R^2=1.0 means model explains *all variation* in the data

R^2=0 means model is as good as predicting average output

R^2<0 means model is worse than predicting average output (unlikely)



“Shrinkage” Feature Selection

Down-weight features that are not useful for prediction…

Quadratic penalty $\lambda\|w\|^2$ down-weights (shrinks) features that are not useful for prediction

Term	LS	Ridge
Intercept	2.465	2.452
lcavol	0.680	0.420
lweight	0.263	0.238
age	−0.141	−0.046
lbph	0.210	0.162
svi	0.305	0.227
lcp	−0.288	0.000
gleason	−0.021	0.040
pgg45	0.267	0.133

Example *Prostate Cancer Dataset* measures prostate-specific cancer antigen with features: age, log-prostate weight (lweight), log-benign prostate hyperplasia (lbph), Gleason score (gleason), seminal vesical invasion (svi), etc.

L2 regularization learns zero-weight for log capsular penetration (lcp)

[Source: Hastie et al. (2001)]

Regularized Least Squares

Ordinary least-squares estimation (no regularizer),

$$w^{\mathrm{OLS}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

L2-regularized Least-Squares (Ridge)

Quadratic Penalty

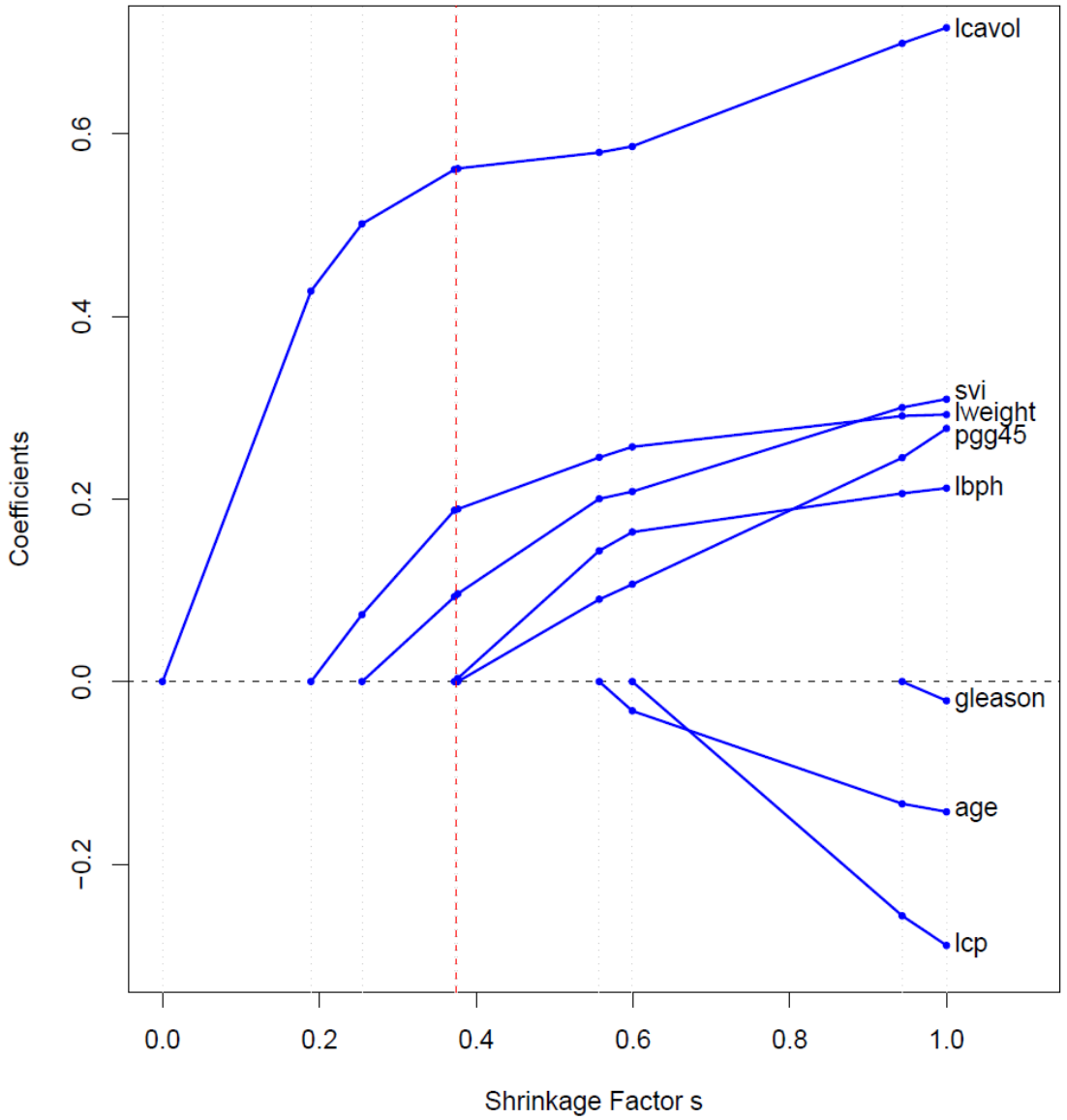
$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|^2$$

L1-regularized Least-Squares (LASSO)

Absolute Value (L1) Penalty

$$w^{\mathrm{L1}} = \arg\min_w \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda|w|$$

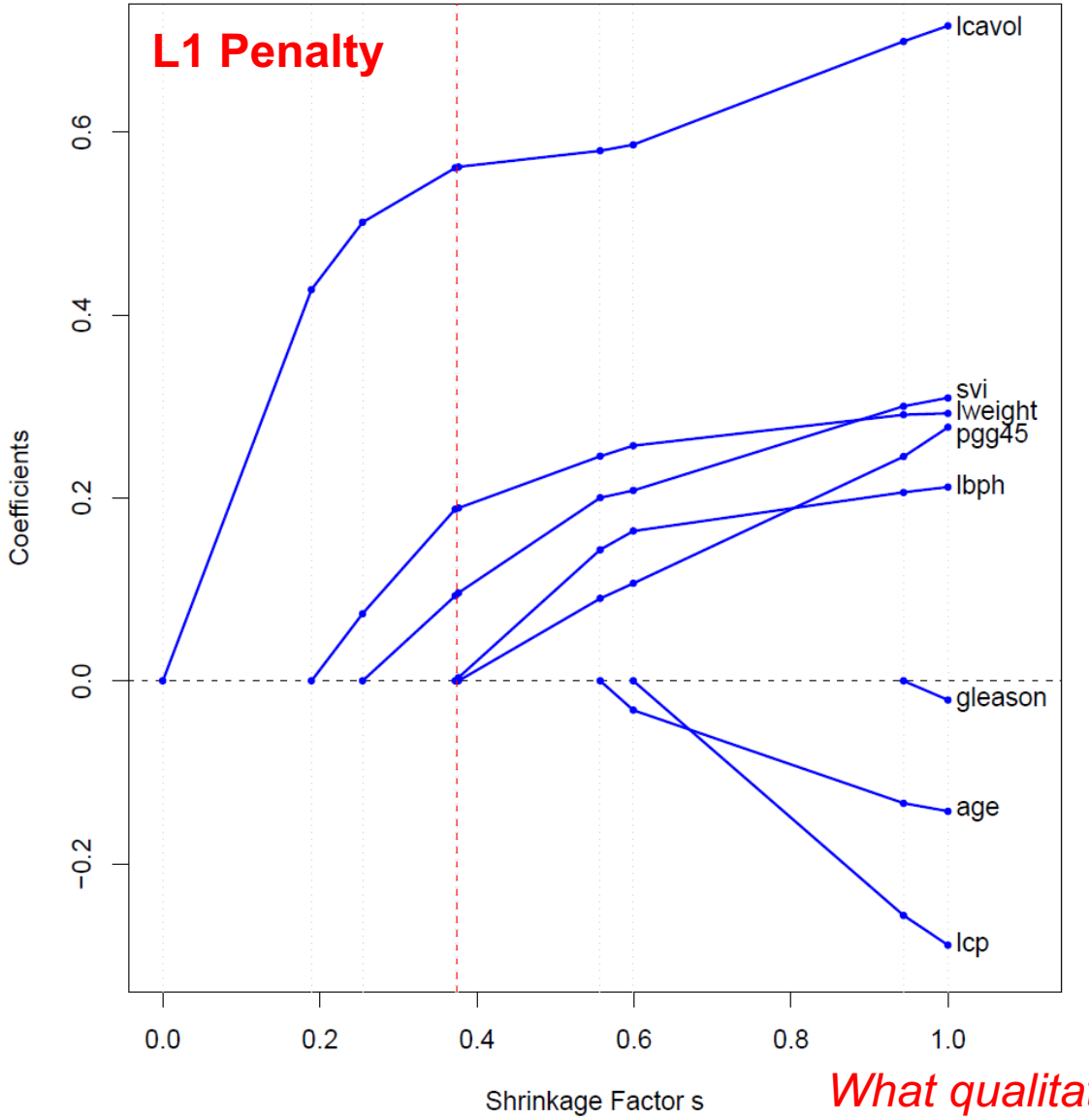
Feature Weight Profiles

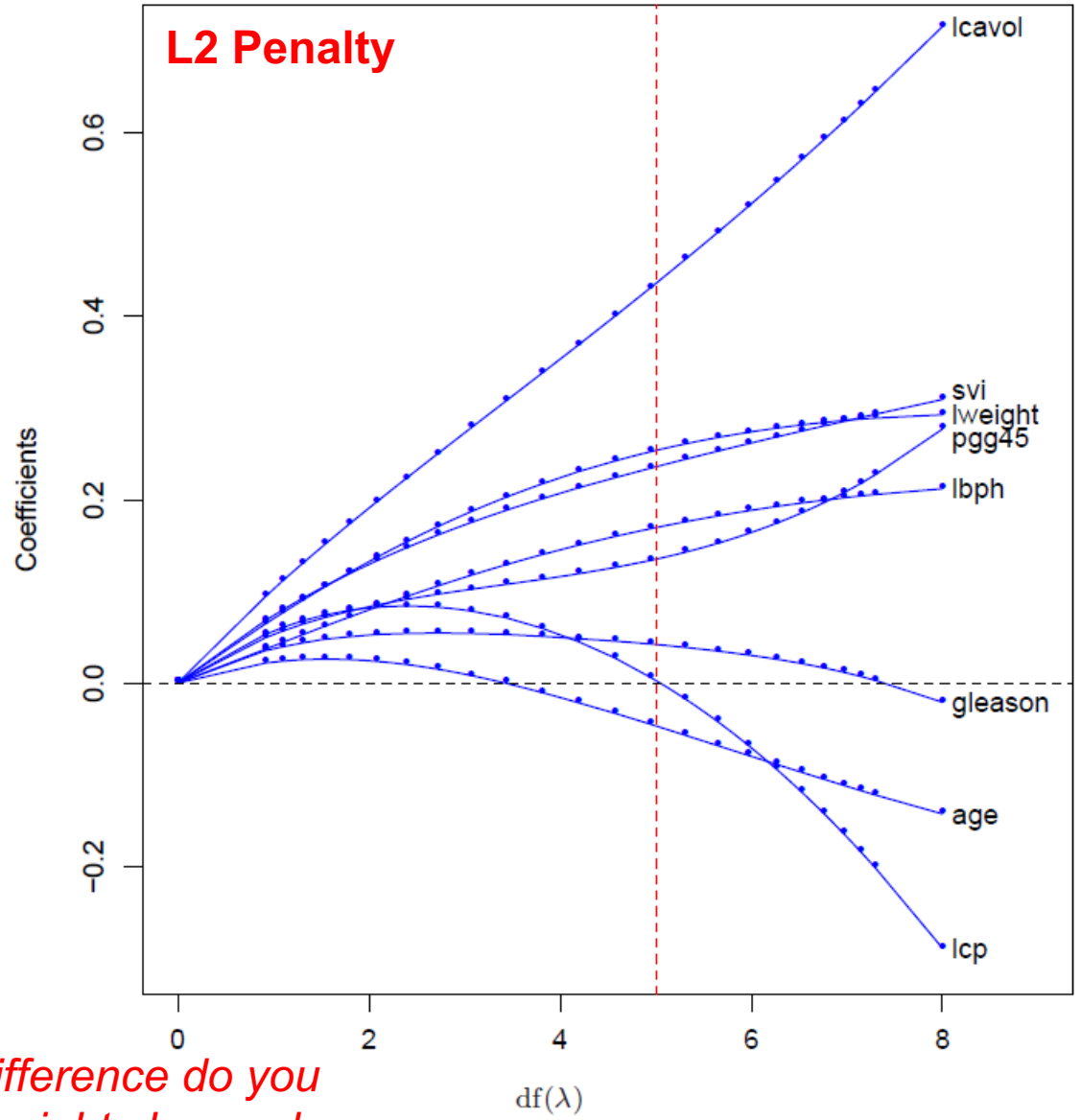


Varying regularization parameter moderates *shrinkage factor*

For moderate regularization strength weights for many features go to zero

Feature Weight Profiles





What qualitative difference do you see between the weights learned with these two penalties?

Example: Prostate Cancer Dataset

Term	LS	Ridge	Lasso
Intercept	2.465	2.452	2.468
lcavol	0.680	0.420	0.533
lweight	0.263	0.238	0.169
age	−0.141	−0.046	
lbph	0.210	0.162	0.002
svi	0.305	0.227	0.094
lcp	−0.288	0.000	
gleason	−0.021	0.040	
pgg45	0.267	0.133	

Best LASSO model learns to ignore several features (age, lcp, gleason, pgg45).

Wait…Is **age** really not a significant predictor of prostate cancer? What's going on here?

Age is highly correlated with other factors and thus *not significant* in the presence of those factors

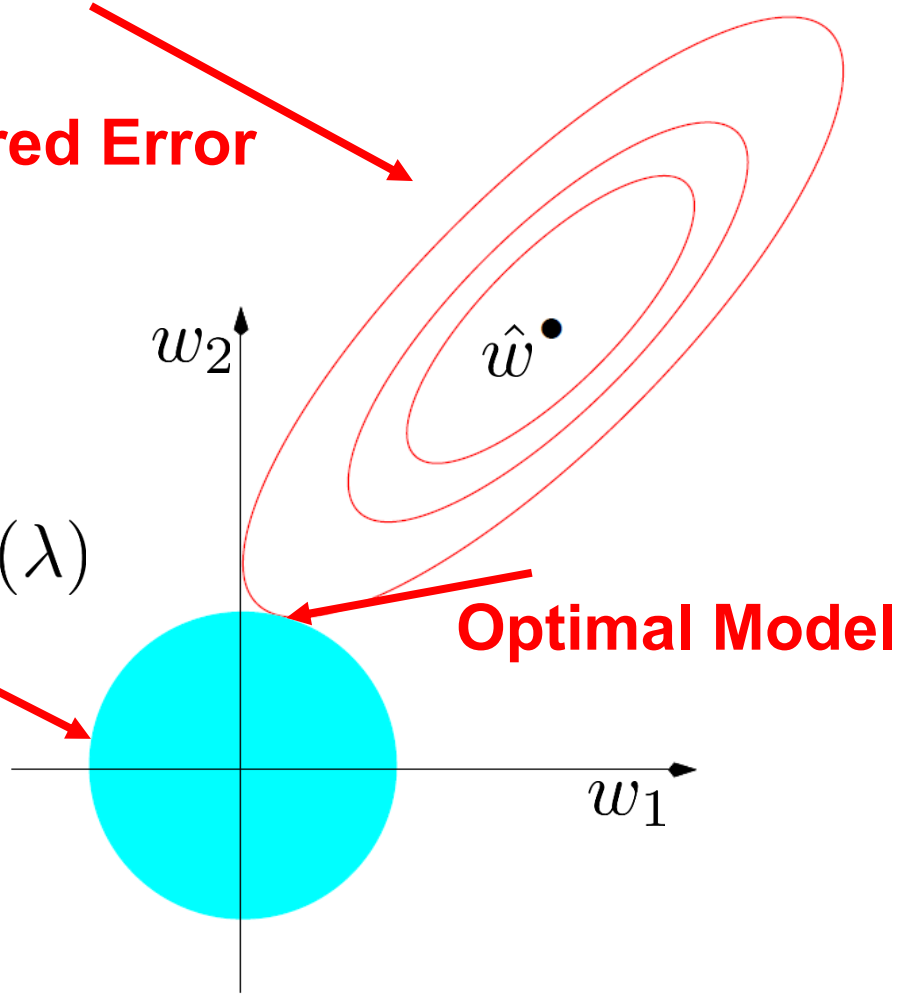
Why LASSO promotes sparsity

$$\min_w \sum_i (y_i - w^T x)^2$$

Squared Error

Total Weight Norm

$$\|w\|^2 = \delta(\lambda)$$



Optimal Model

- **Fact:** the solution of

$$\arg\min_w \ \|Xw - y\|_2^2 + \lambda\|w\|_2 ,$$

Is equivalent to the solution of the constrained optimization problem:

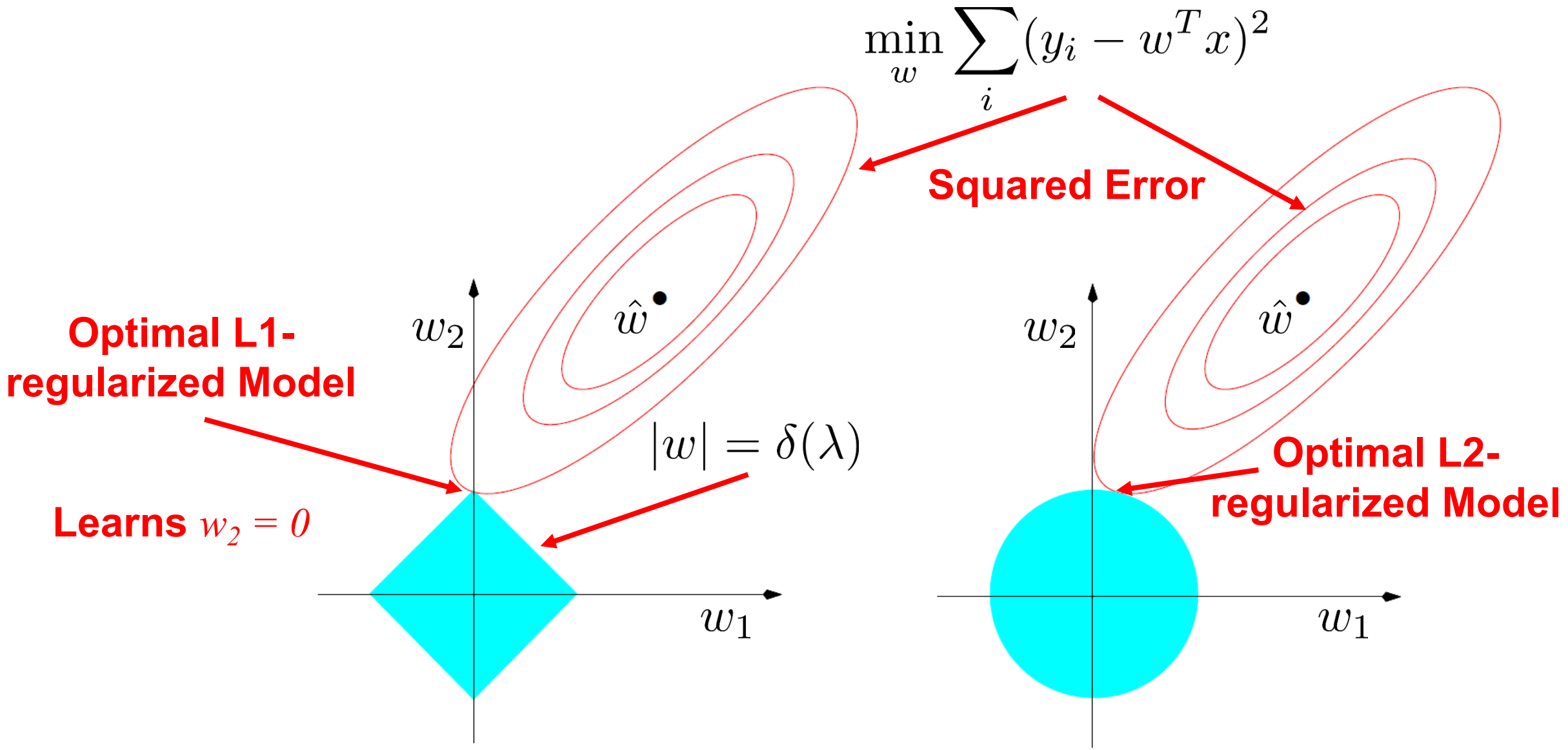
$$\arg\min_{w:\|w\|_2 \le \delta(\lambda)} \ \|Xw - y\|_2^2$$

for some $\delta(\lambda)$

L2 penalized regression rarely learns feature weight that are *exactly zero…*

[Source: Hastie et al. (2001)]

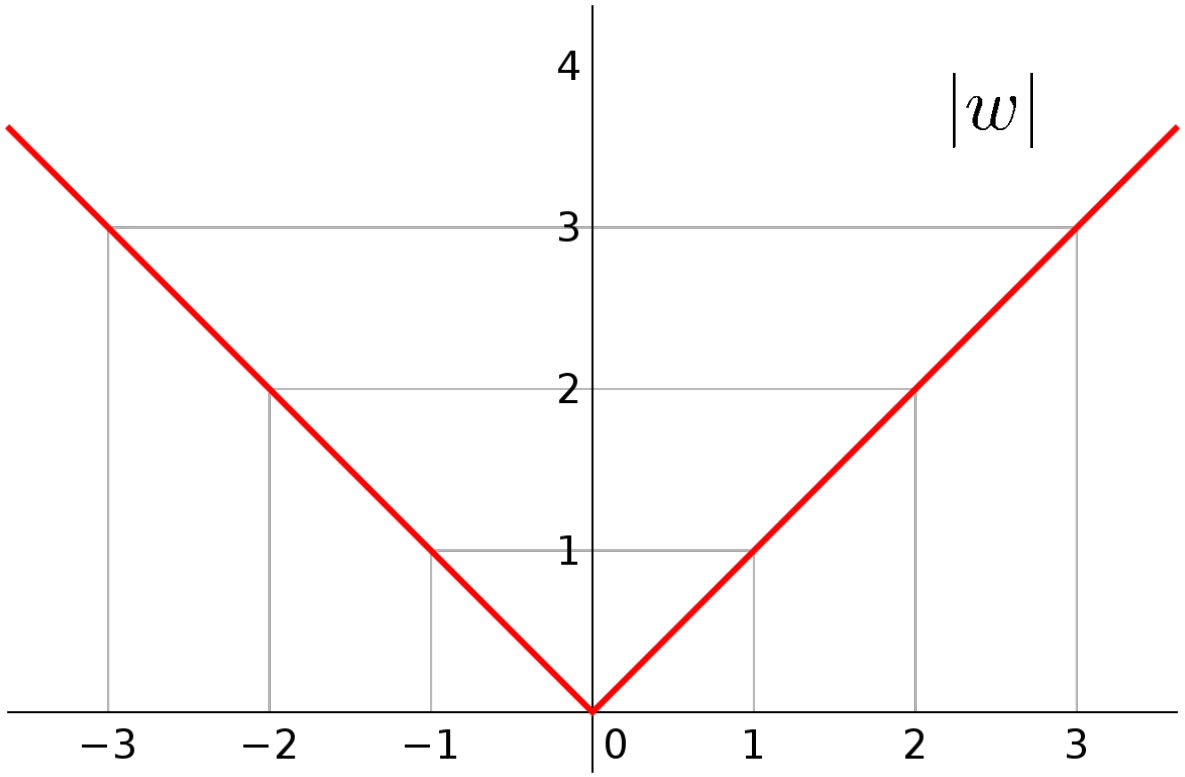
Why LASSO promotes sparsity



LASSO is able to zero-out weights that are not predictive…

Learning L1 Regularized Least-Squares

$$w^{\mathrm{L1}} = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda |w|$$



Not differentiable…

$$\frac{d}{dx}|x|$$

…doesn't exist at x=0

Can't set derivatives to zero as in the L2 case!

Learning L1 Regularized Least-Squares

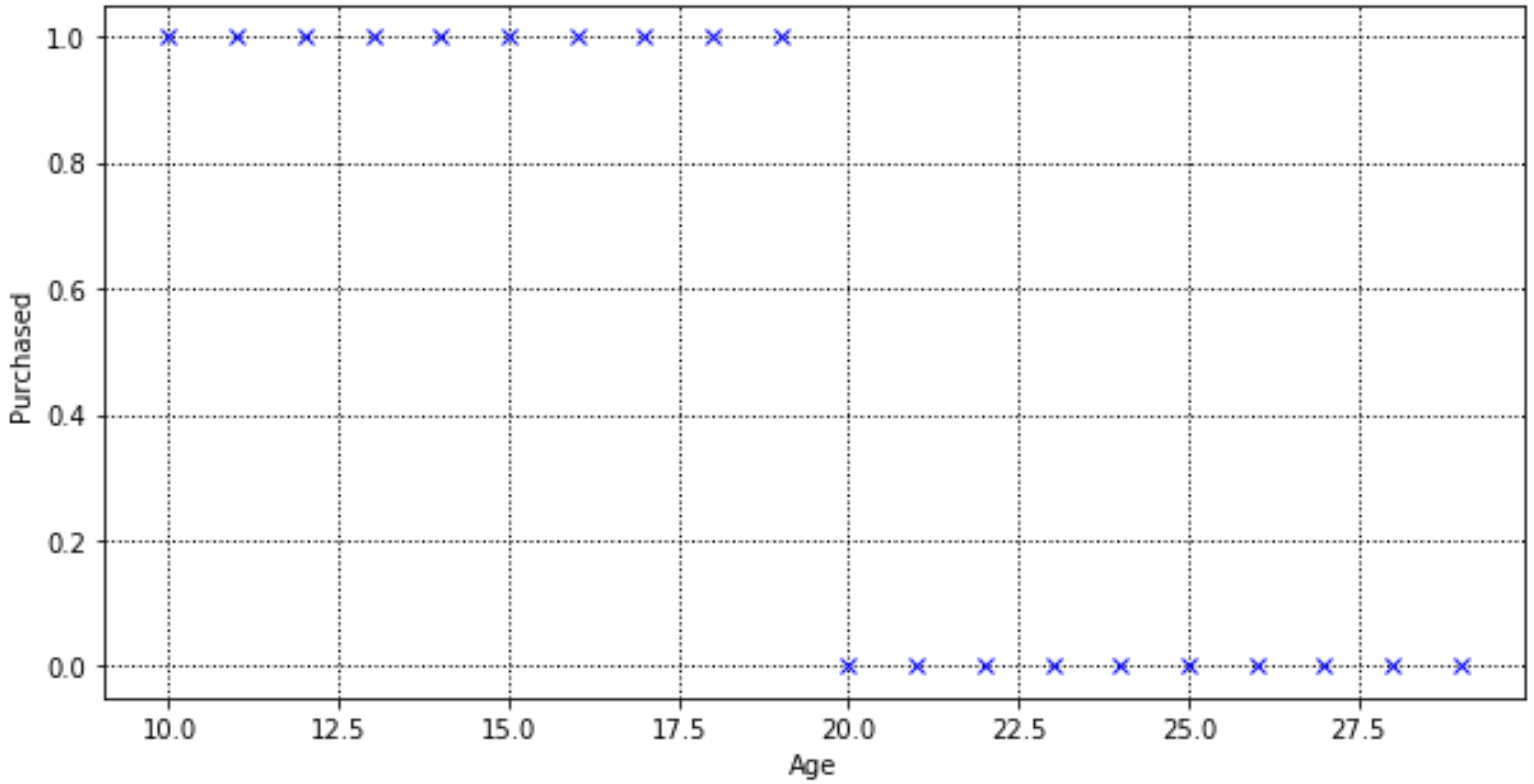
- Not differentiable, no closed-form solution
- But it is convex! Can be solved by standard convex optimization packages (e.g. CVXPY)
- Efficient optimization algorithms exist
- *Least Angle Regression* (LAR) computes **full solution path** for a range of λ values
- Can be solved as efficiently as L2 regression

Outline

- Linear Models for Regression
  - Least Squares Estimation
  - Regularized Least Squares
- Linear Models for Classification
  - Logistic Regression

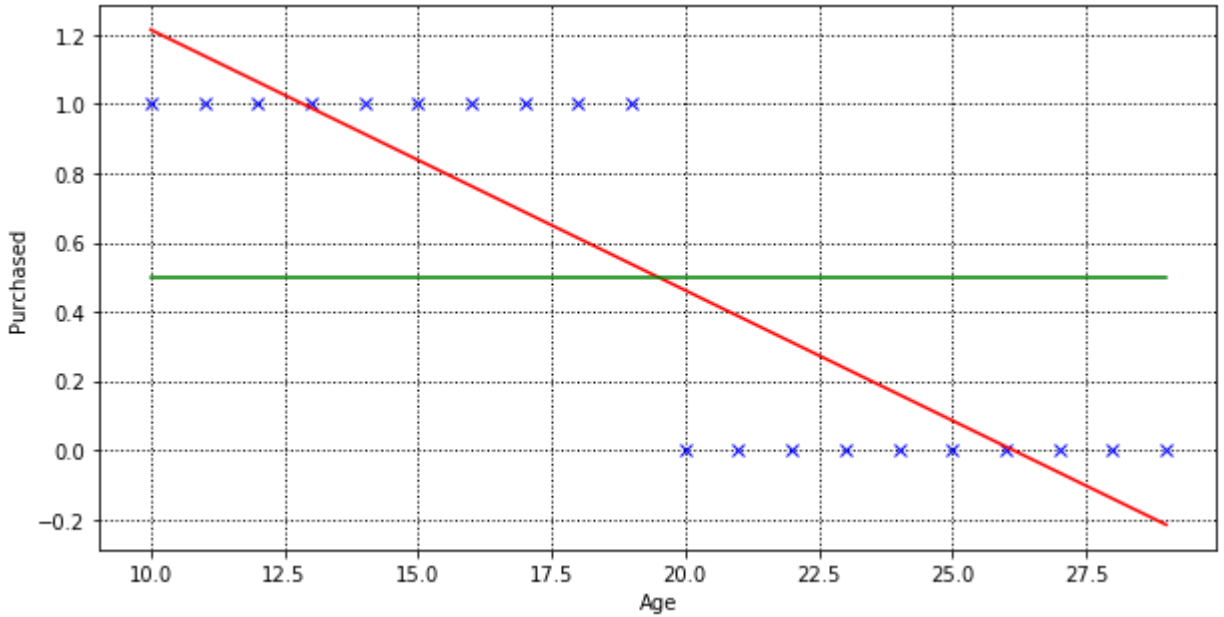
Classification as Regression

Suppose our labels are binary, y={0,1}. How can we use linear regression ideas to solve this classification problem?



https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

Least squares classification: classification as regression



First Idea Fit a least-square linear regressor w to the data (red). Classify points based on whether they are *above* or *below* the midpoint (green).

$$h(x) = I(w^T x \geq 0.5)$$

- This is a *discriminant* function, since it discriminates between classes
- It is a linear function and so is a *linear discriminant*
- We can call this approach **least squares classification**

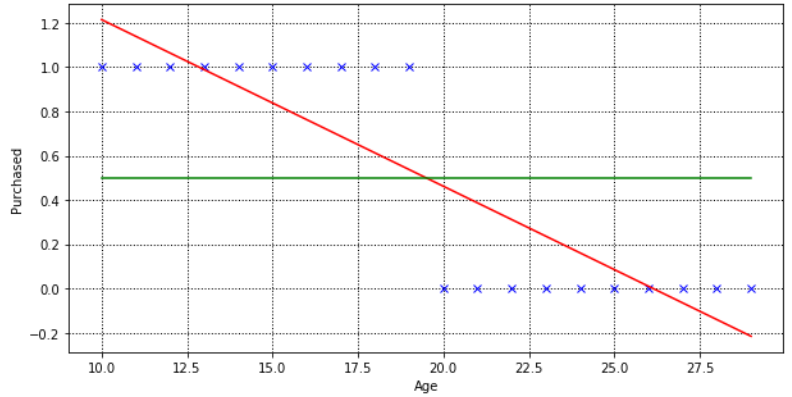
https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

Least squares classification: why it works

Recall: Bayes optimal classifier

$$h^*(x) = I\left(P(y = 1 \mid x) \geq 0.5\right)$$

I(A) = 1 if A is true, = 0 otherwise



(Why?)

Recall the linear regression model, $p(y \mid x) = \mathcal{N}(w^T x, \sigma^2)$

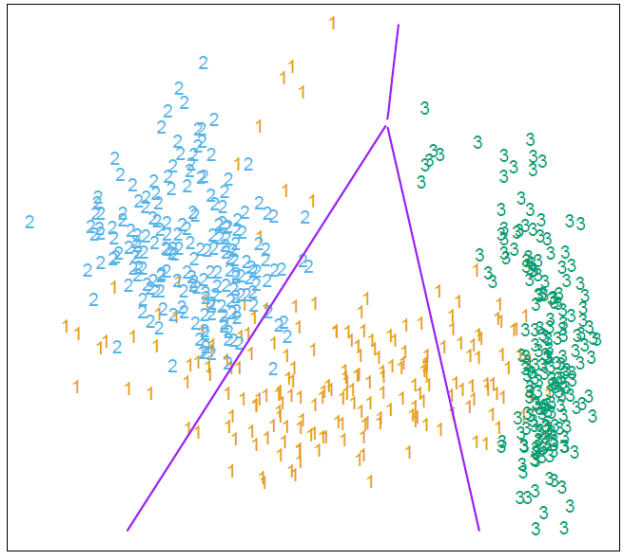
A wrong, but useful model..

So linear regression aims at predicting the expected value of y given x,

$$w^T x \approx \mathbf{E}[y \mid x] \quad = 1 \times P(y = 1 \mid x) + 0 \times P(y = 0 \mid x) \quad = P(y = 1 \mid x)$$

Thus, we expect $I(w^T x \geq 0.5)$ to closely approximate the Bayes optimal classifier

Least squares classification: multiclass setting



Suppose we have K classes. Each example's label is represented by an *indicator vector*,

$$Y = (Y_1, \ldots, Y_K)$$

With $Y_k = 1$ if example is in class k,
e.g. for K=5, class 3, Y=(0,0,1,0,0).

For N training inputs, create NxK matrix of outputs $\mathbf{Y}$ and solve,

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

W is DxK matrix of K linear regression models
each column is for a different class

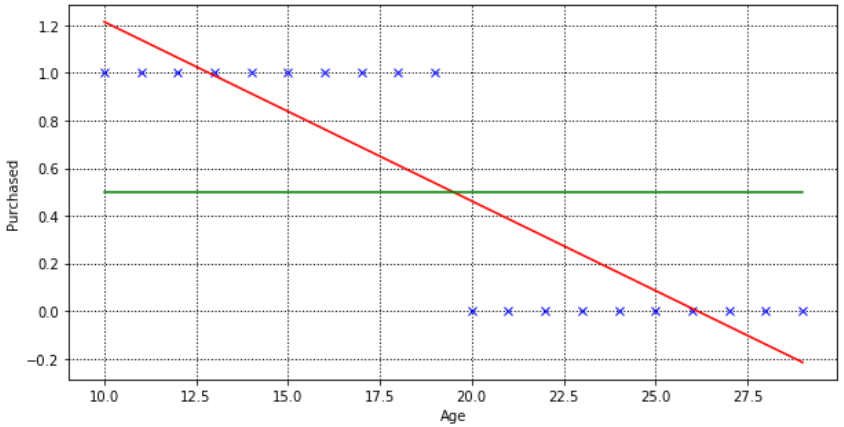
- Compute fitted output $f(x) = x^\top \mathbf{W}$ a K-vector
- Identify largest component and classify as,

$$C = \arg\max_k \; f_k(x)$$

This is an instance of multi-output linear regression

[Image: Hastie et al. (2001)]

Linear Probability Models



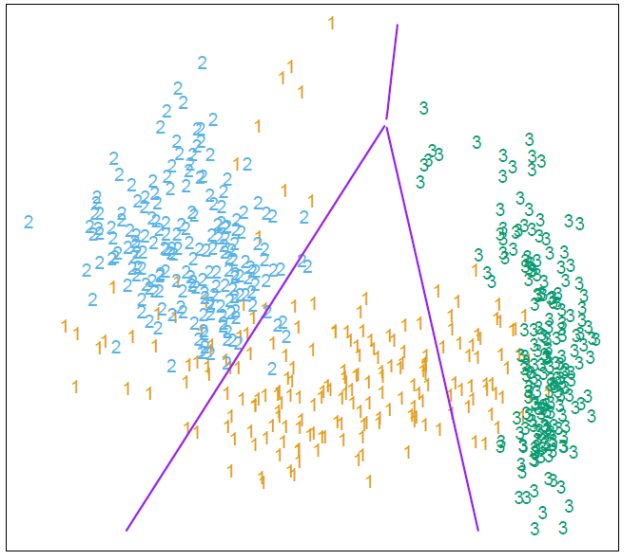
Binary Classification Linear model approximates probability of class assignment,

$$w^T x \approx p(y = 1|x)$$

Multiclass Classification Multiple decision boundaries, each approximated by the class-specific linear model,

$$\hat{f}_k(x) = W_{k:}^\top x$$

Where $W_{k:}$ is k[th] column of W



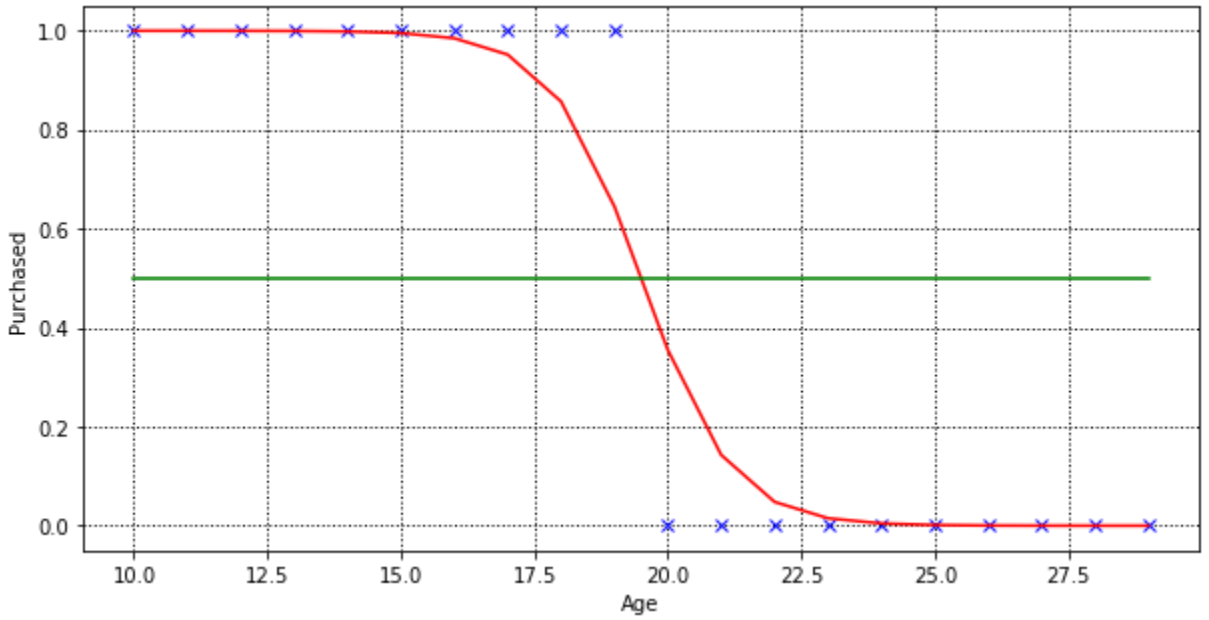
Approximates probability of class assignment,

$$\hat{f}_k(x) \approx p(y = k \mid x)$$

Any drawback with this approach?

$w^T x$, $W_{k,:}^T x$ are not guaranteed to be valid probabilities in [0,1]!

Logistic Regression

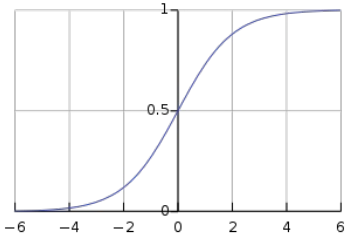


Idea Distort the prediction score in some way to map to [0,1] so that it is actually a probability.

$$f(x) = \sigma(w^T x)$$

Uses the *logistic* function,

$$\sigma(z) = \frac{\exp(z)}{1+\exp(z)}$$



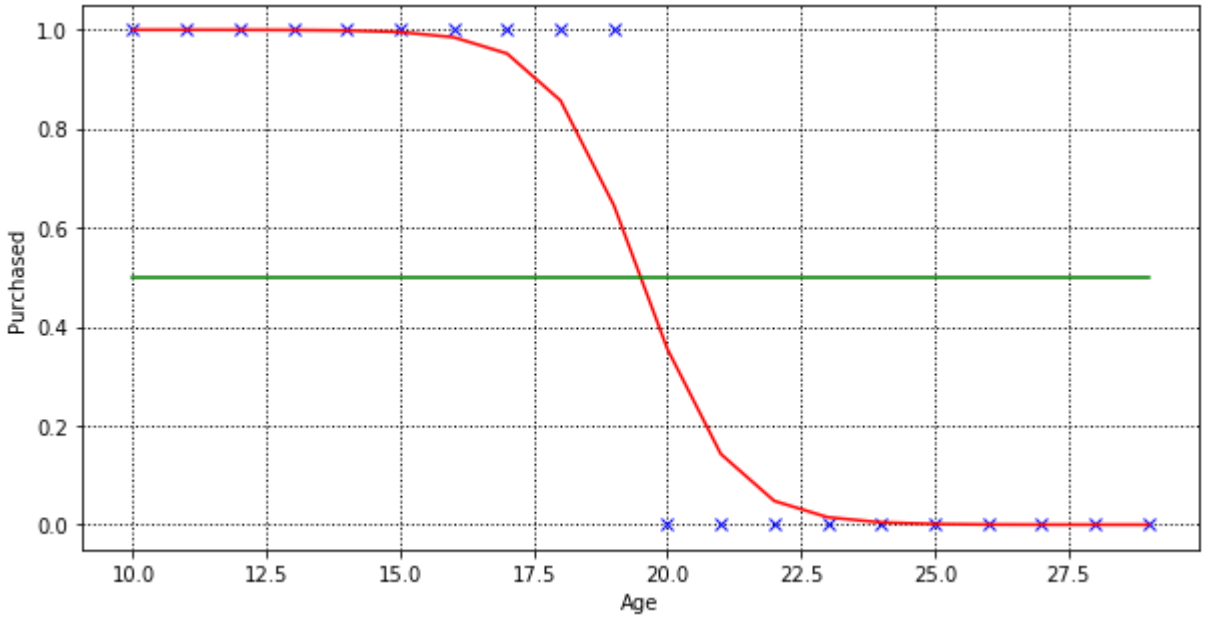
- Logistic function is a type of *sigmoid* or *squashing function*, since it maps any value to the range [0,1]

- Prediction now actually maps to a valid probability

$$f(x) = \sigma(w^T x) = p(y = 1|w, x)$$

https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

Logistic Regression: Decision Boundary



Bayes optimal prediction on example x:
Predict 1 $\Leftrightarrow P(y = 1 \mid x) \geq 0.5$

$$\Leftrightarrow (\text{odd ratio}) \frac{P(y = 1 \mid x)}{P(y = 0 \mid x)} \geq 1$$

$$\Leftrightarrow \ln \frac{P(y = 1|x)}{P(y = 0|x)} \geq 0$$

logistic regression posits that for some w: $p(y = 1|w, x) = \sigma(w^T x) = \frac{e^{w^T x}}{1+e^{w^T x}}$

$$\Longrightarrow \ln \frac{p(y = 1 \mid w, x)}{p(y = 0 \mid w, x)} = w^T x$$

This induces a *linear decision boundary*

Logistic regression learns w, which gives a *linear classifier* $I(w \cdot x \geq 0)$

Logistic vs. Logit Transformations

Logistic function maps the linear prediction score to the interval [0,1],

$$\sigma(w^T x) = \frac{\exp(w^T x)}{1 + \exp(w^T x)}$$

Logit function is defined for probability values p in [0,1] as,

$$\text{logit}(p) = \log \frac{p}{1-p}$$

Logit is the *inverse* of the logistic function,

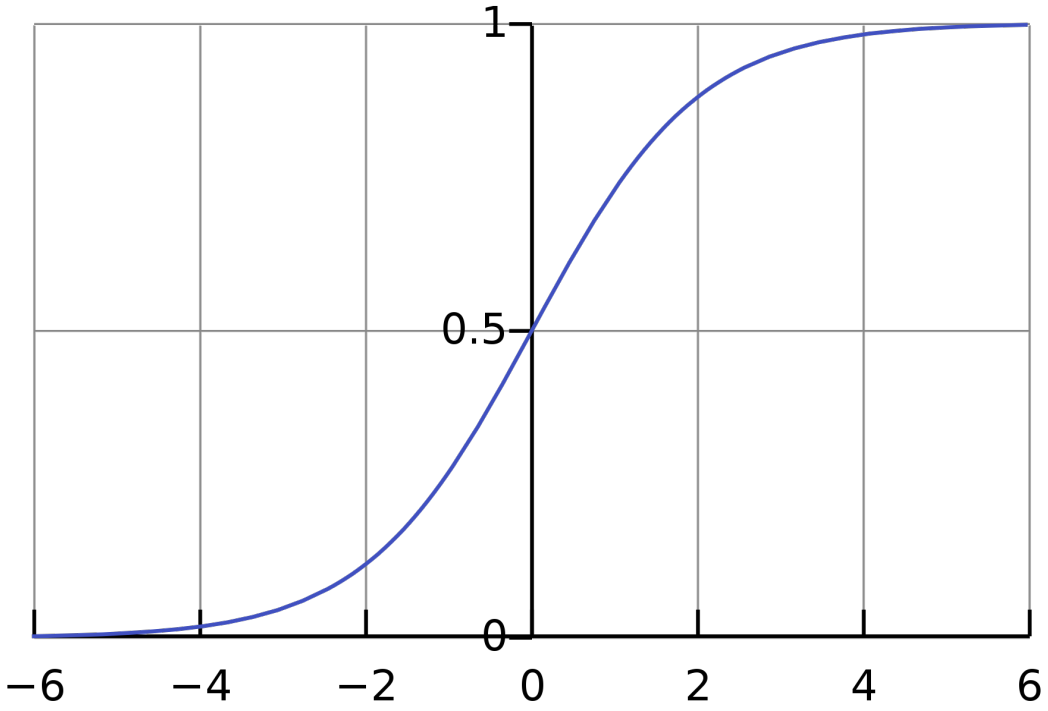
$$\text{logit}(\sigma(w^T x)) = w^T x$$

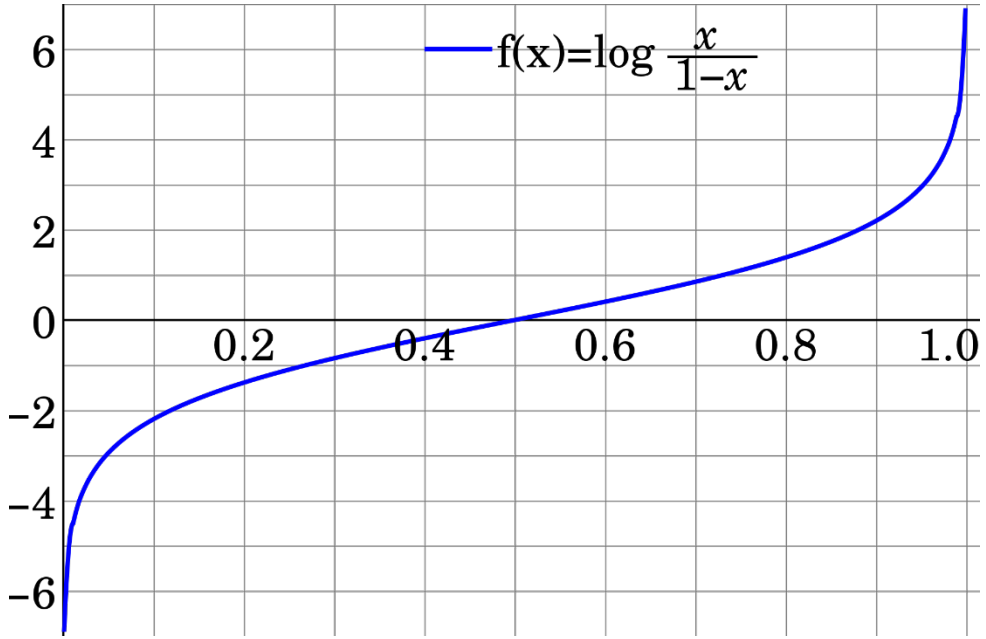
Logit is also the log odd, and thus induces decision boundary for our binary classifier

Logistic vs. Logit Transformations

Logistic Function ⟷ **Logit Function**

inverse function





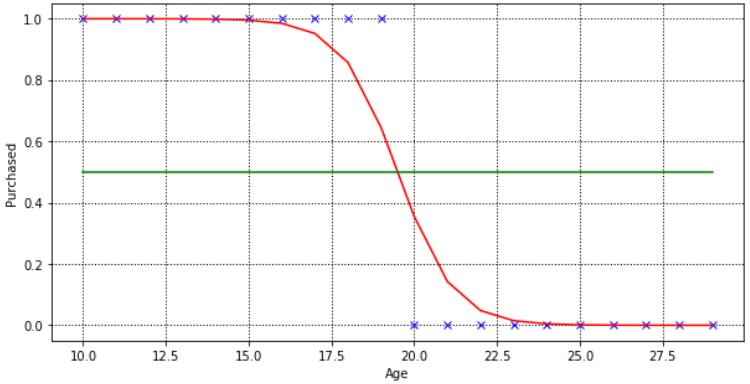
Maps $(-\infty, \infty)$ to [0,1]

Maps [0,1] to $(-\infty, \infty)$

Logistic function is also widely used in Neural Networks – for classification the last layer is typically a logistic function

logistic regression: summary

- Following the probabilistic ML recipe:
  1. **Model** For $z = (x, y)$, assume $P(y = 1|x) = \sigma(w^T x)$



  2. **Training:** Learn the model parameter $\hat{w}$ using maximum likelihood estimation (MLE): $\text{maximize}_w \log \prod_{i=1}^{n} P(y_i \mid x_i, w)$,

How? We will come back to it

  3. **Test:** Make prediction based on the learned model $P(y = 1|x) = \sigma(\hat{w}^T x)$ given a new example x, predict its label by $I(\hat{w}^T x \geq 0)$

Multiclass Logistic Regression

Classification decision based on log-odd-ratio compared to final class,

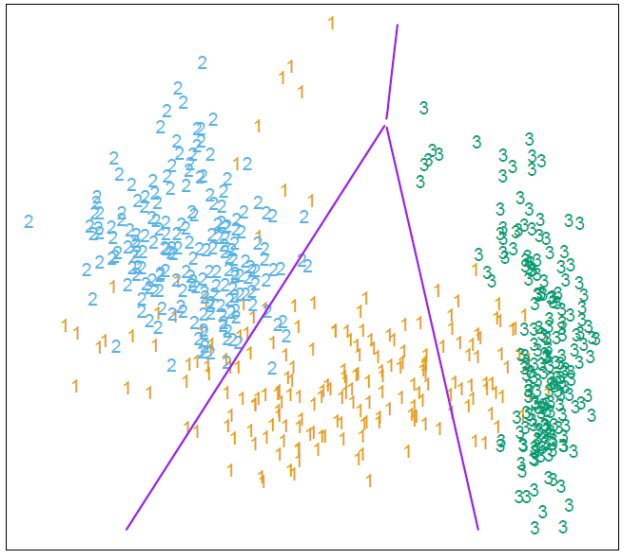
$p(C = i \mid x)$'s sum to 1

$$\log \frac{p(C = 1 \mid x)}{p(C = K \mid x)} = w_1^T x$$

$$\log \frac{p(C = 2 \mid x)}{p(C = K \mid x)} = w_2^T x$$

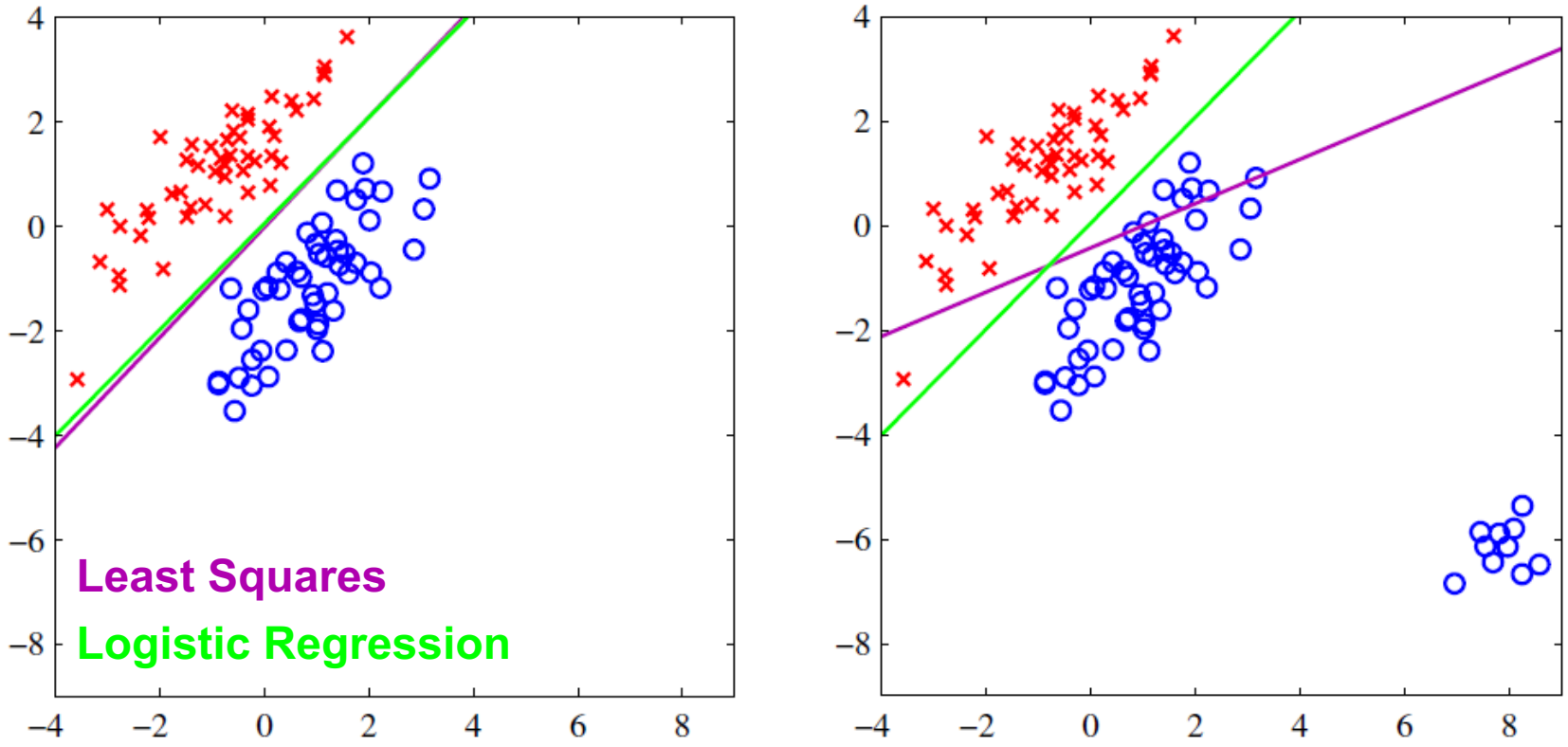
$$\vdots$$

$$\log \frac{p(C = K - 1 \mid x)}{p(C = K \mid x)} = w_{K-1}^T x$$



Choice of denominator class is arbitrary, but use K by convention

Least Squares vs. Logistic Regression

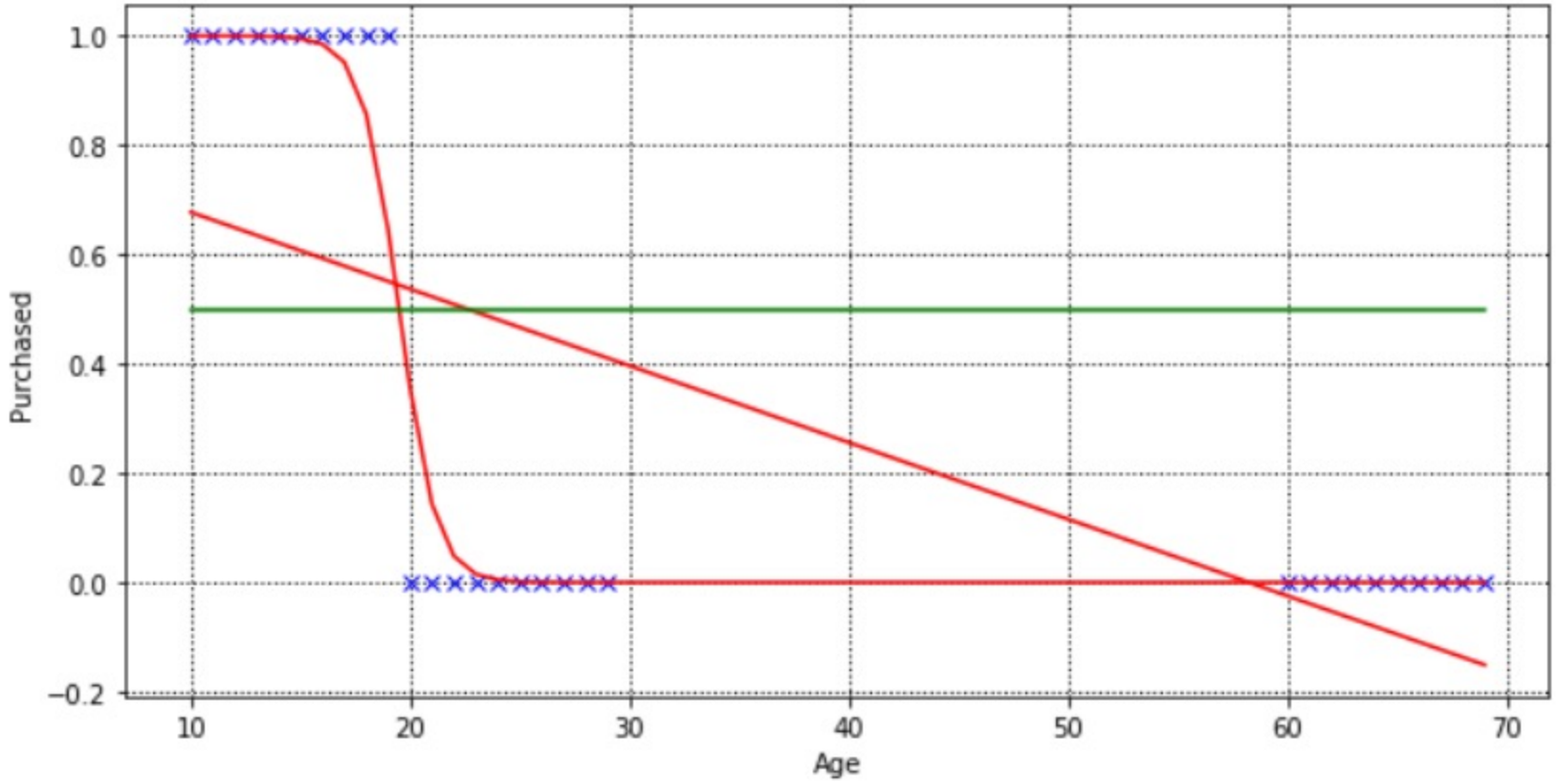


- Both models learn a linear decision boundary
- Least squares can be solved in closed-form (convex objective)
- Least squares is sensitive to outliers (need to do regularization)

[Source: Bishop “PRML”]

Least Squares vs. Logistic Regression

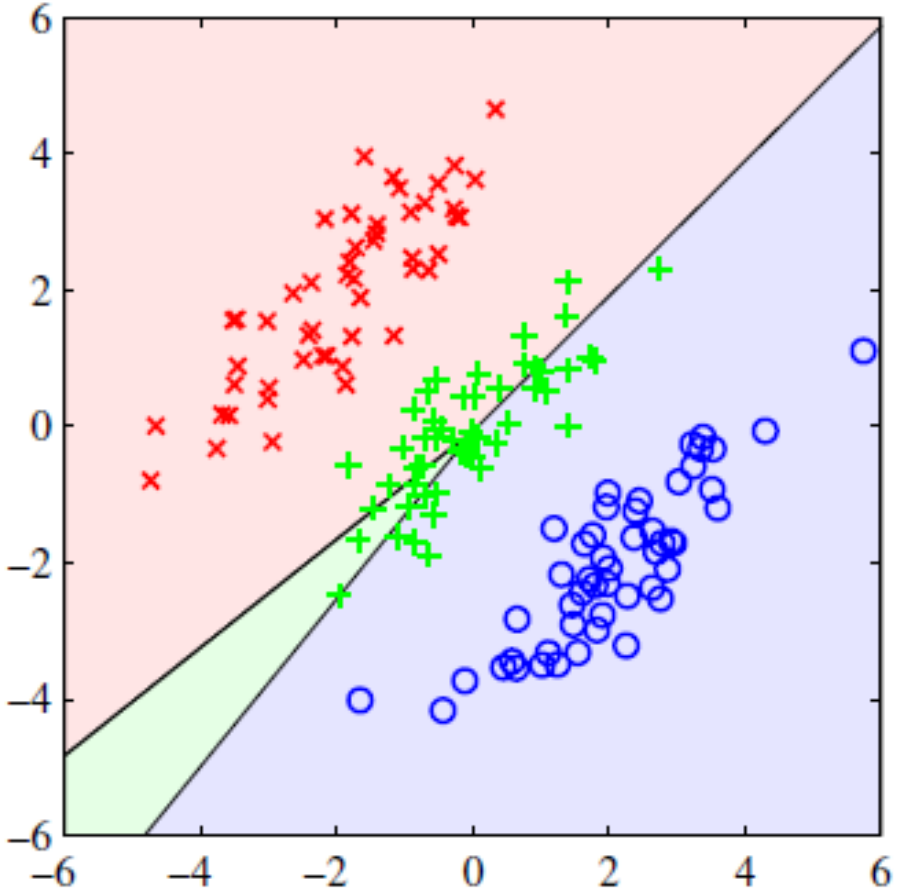
Similar qualitative comparisons in 1-dimension



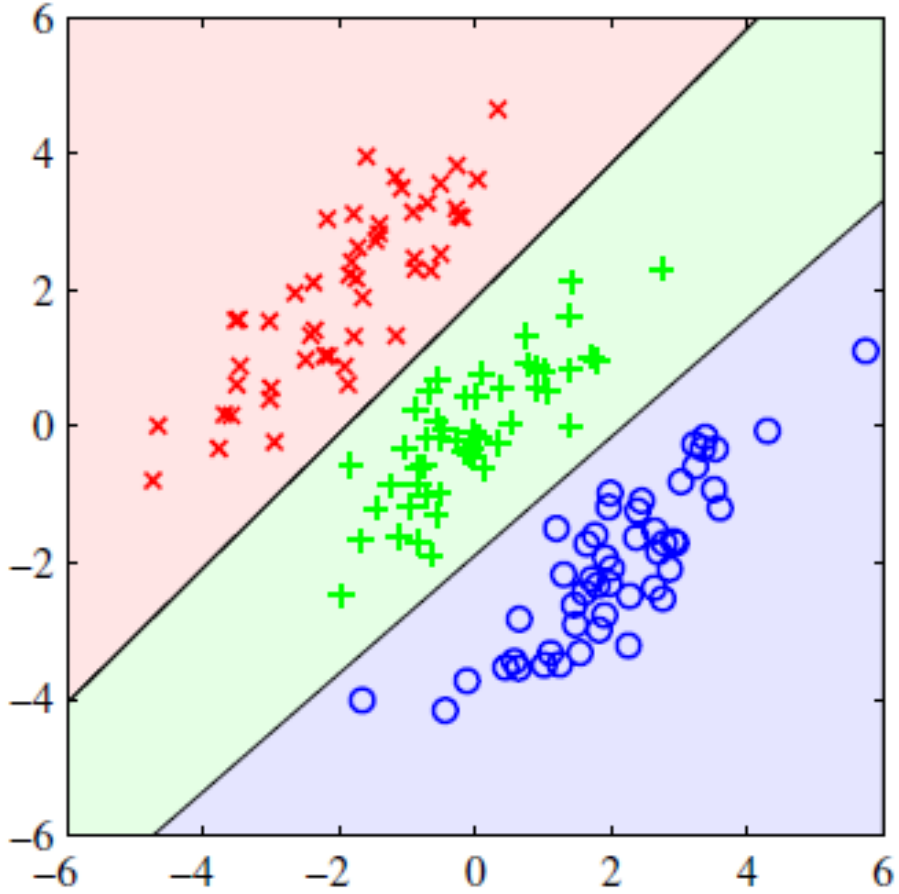
https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

Least Squares vs. Logistic Regression

Least Squares



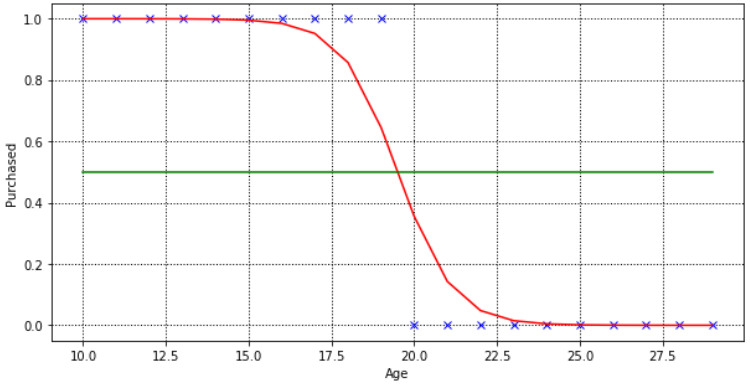
Logistic Regression



[Source: Bishop “PRML”]

logistic regression: summary

- Following the probabilistic ML recipe:
  1. **Model** For $z = (x, y)$, assume $P(y = 1|x) = \sigma(w^T x)$



  2. **Training:** Learn the model parameter $\hat{w}$ using maximum likelihood estimation (MLE): $\text{maximize}_w \log \prod_{i=1}^{n} P(y_i \mid x_i, w)$,

Let's see how we can do this step…

  3. **Test:** Make prediction based on the learned model $P(y = 1|x) = \sigma(\hat{w}^T x)$ given a new example x, predict its label by $I(\hat{w}^T x \geq 0)$

Logistic Regression: Model Training

Fit by maximum likelihood—start with the *binary* case

Recall: how is this defined? $\sigma(w^T x)$

A convenient way to write $P(y \mid x, w)$:

$$p(y \mid x, w) = p(y = 1 \mid x, w)^y (1 - p(y = 1 \mid x, w))^{(1-y)}$$

This is true because y can only take 1 or 0

Given N iid training data pairs the log-likelihood function is,

$$\begin{aligned}
\mathcal{L}_N(w) &= \sum_{i=1}^{N} \log p(y_i \mid x_i, w) \\
&= \sum_i \{y_i \log p(y_i = 1 \mid x_i, w) + (1 - y_i) \log p(y_i = 0 \mid x_i, w)\} \\
&= \sum_i \{y_i \log \sigma(w^T x_i) + (1 - y_i) \log(1 - \sigma(w^T x_i))\} \\
&= \sum_i \left\{y_i w^T x_i - \log\left(1 + e^{w^T x_i}\right)\right\}
\end{aligned}$$

Fitting Logistic Regression

$$w^{\mathrm{MLE}} = \arg\max_{w} \sum_{i} \left\{ y_i w^T x_i - \log\left(1 + e^{w^T x_i}\right) \right\}$$

Fact: This is a convex optimization problem => stationary points are optimal
Computing the derivatives with respect to each element w_d,

$$\frac{\partial \mathcal{L}}{\partial w_d} = \sum_{i} x_{di} \left(y_i - \frac{e^{w^T x_i}}{1 + e^{w^T x_i}} \right) = 0$$

- For D features this gives us D equations and D unknowns
- But equations are nonlinear and can't be solved
- Can use standard convex optimization toolbox (CVXPY) to solve it
- Can also be solved with Newton's method (Iterative Weighted Least Squares)

https://retostauffer.github.io/Rfoehnix/articles/logisticregression.html

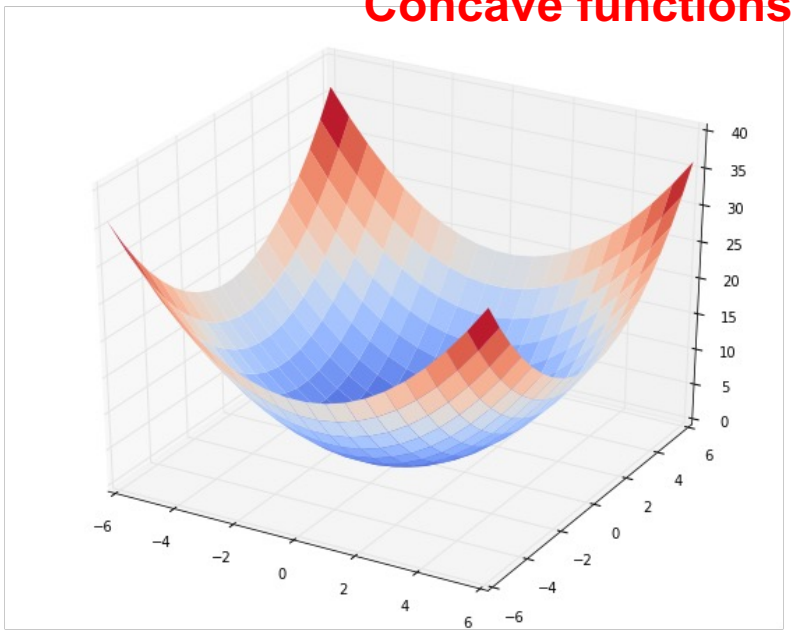
Fitting Logistic Regression

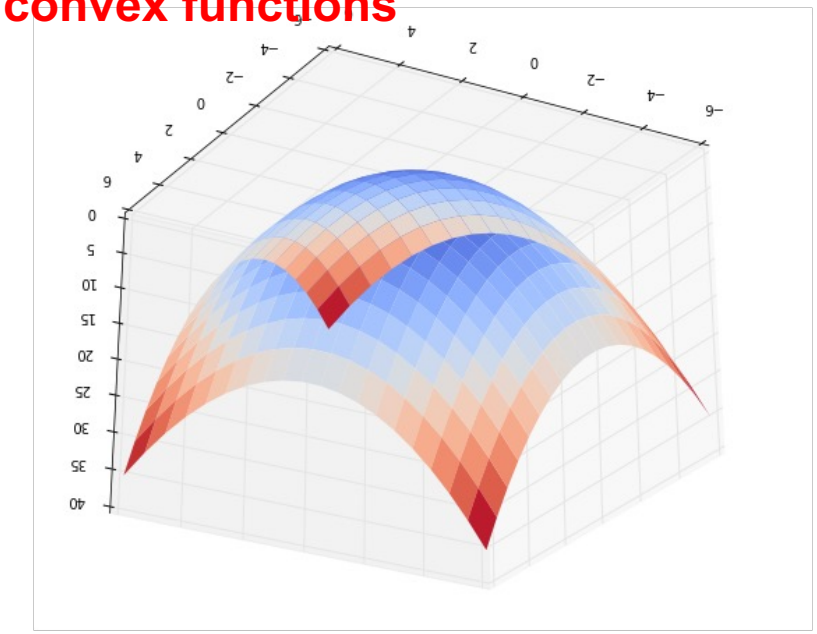
$$w^{\mathrm{MLE}} = \arg\max_{w} \sum_{i} \left\{ y_i w^T x_i - \log\left(1 + e^{w^T x_i}\right) \right\}$$

This is a *convex optimization problem*, which includes

- minimizing a convex function
- maximizing a concave function (for logistic regression)

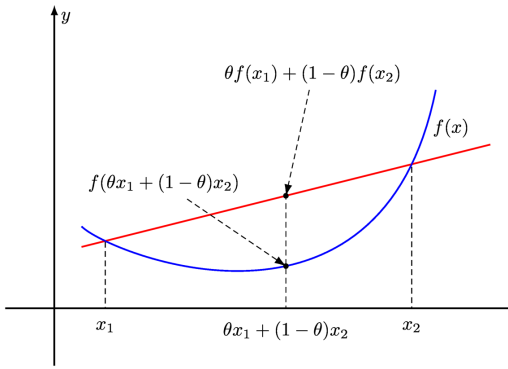
Concave functions = negative of convex functions



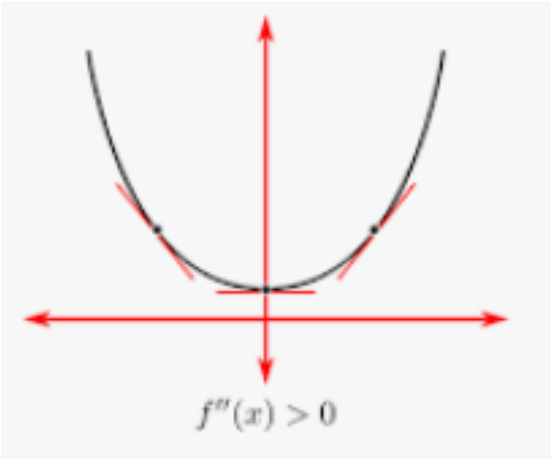


Checking convexity – a toolkit

- How to check if function f is convex?
  - Idea 1: checking definition **Doable, but cumbersome..**
    - for all $u, v, \alpha \in [0,1]$, $f(\alpha u + (1-\alpha)v) \leq \alpha f(u) + (1-\alpha) f(v)$

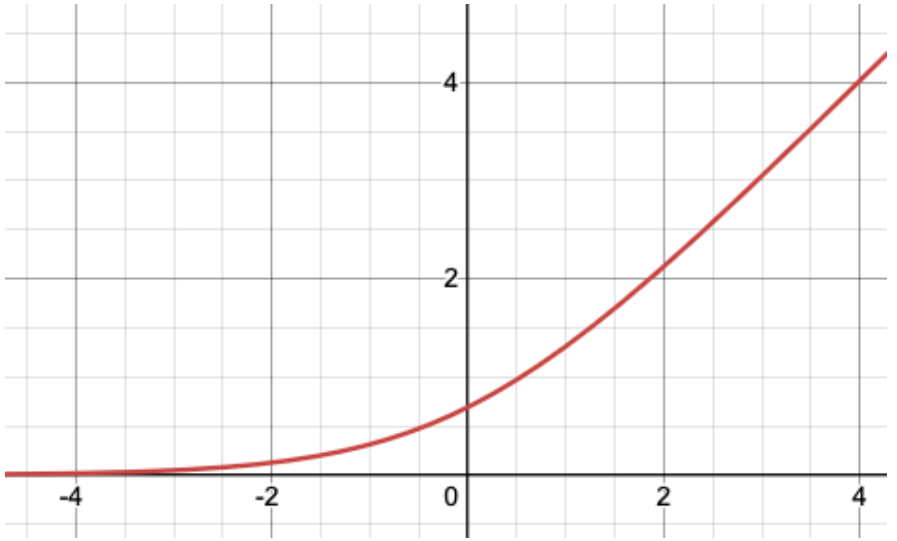


  - Idea 2: checking second order derivative
    - **Fact:** for univariate, 2nd order differentiable f, f is convex $\Leftrightarrow f''(u) \geq 0$ for all u



Checking convexity – a toolkit

Example Is $f(z) = \ln(1 + e^z)$ convex?



Let's find $f''(z)$..

- $f'(z) = \frac{e^z}{1+e^z}$
- $f''(z) = \frac{e^z}{(1+e^z)^2}$

Why? Chain rule: $f = \ln(w), w = 1 + e^z$

Why? Again, chain rule: $g = 1 - \frac{1}{w}, w = 1 + e^z$

- $e^z \geq 0, (1 + e^z)^2 \geq 0, \Rightarrow f''(z) \geq 0$ always, thus f is convex
- How about multivariate f?

Checking convexity – a toolkit

- **Fact:** for multivariate, 2nd order differentiable f, f is convex $\Leftrightarrow$ its Hessian $\nabla^2 f(u) \succcurlyeq 0$ for all u

This is a DxD matrix, with entries being f's partial derivatives

- Notation: $A \succcurlyeq 0 \Leftrightarrow A$ is positive semidefinite (PSD)
- This means for any w, $w^T A w \geq 0$

PSD are natural generalizations of nonnegativity

- Exercise: verify that

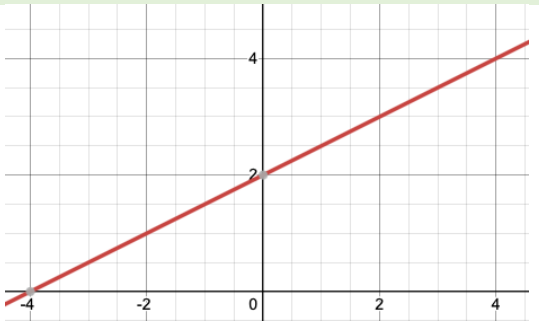
$$F(w) = \sum_i \left\{ y_i w^T x_i - \log\left(1 + e^{w^T x_i}\right) \right\}$$

  is concave in w by checking its Hessian

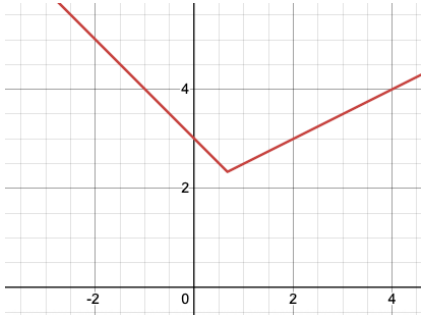
quite cumbersome.. first compute the matrix, then verify the PSD-ness

- Is there an easier way?

Checking convexity – a toolkit



1. Linear functions are both convex and concave
2. Norms are convex
3. If f, g are convex, then
   a) max{f(x), g(x)} is convex
   b) f(x) + g(x) is convex
   c) if g is nondecreasing, then h(x) := g(f(x)) is convex => e.g., $h(w) = \|w\|^2$



4. Convexity is invariant under affine maps:
   if f is convex, then $f(Ax + b)$ is also convex

- Example: show that logistic regression's minimization objective is convex:

Property 3b) $\sum_{i=1}^{n} \left(-w^T x_i + \ln\left(1 + e^{w^T x_i}\right) \right)$

Property 1 **Property 1**

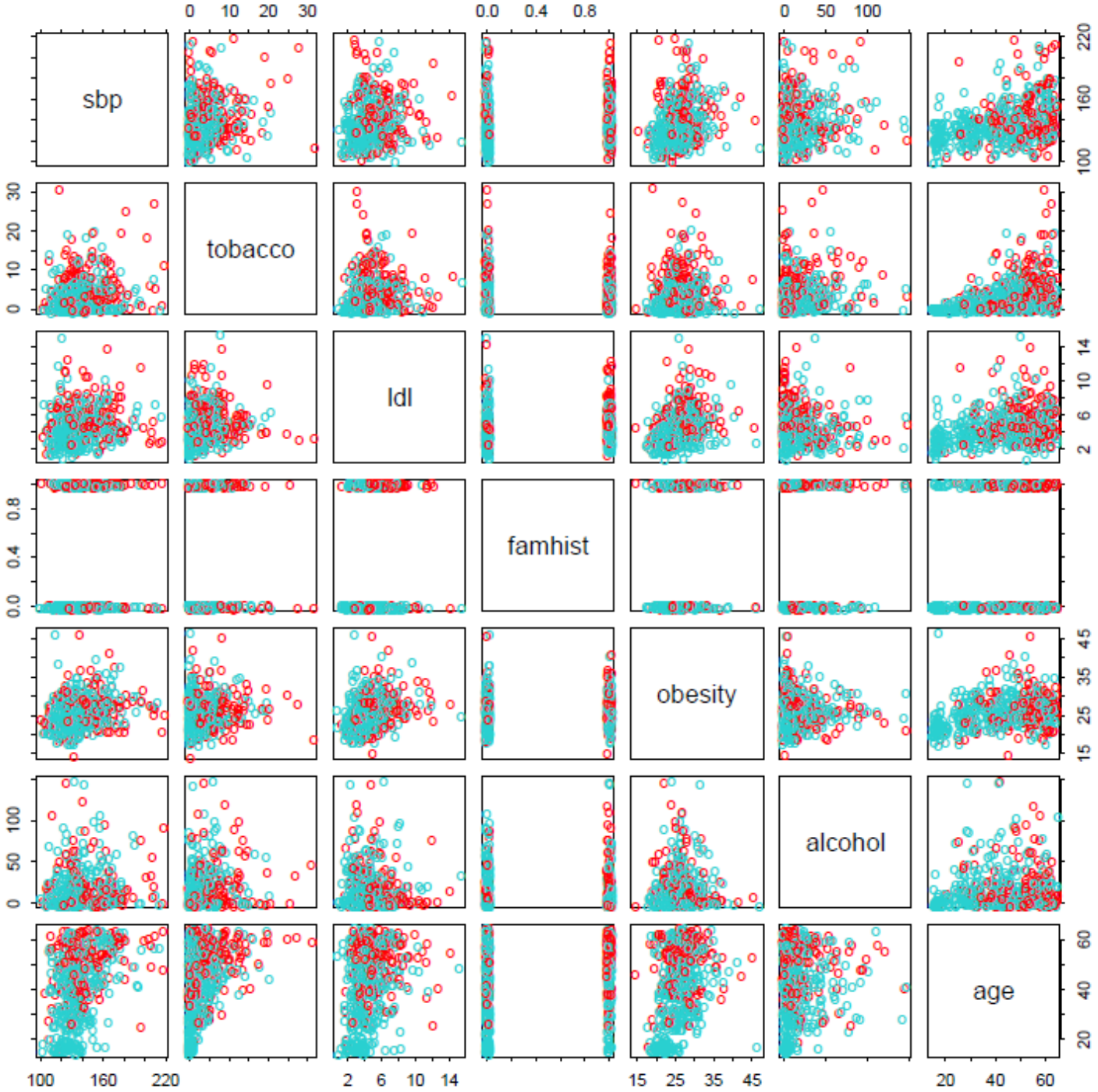
Property 4, and convexity of $\ln(1 + e^z)$

Using Logistic Regression

The role of Logistic Regression differs in ML and Data Science,

- In Machine Learning we use Logistic Regression for building predictive classification models
- In Data Science (Explorative Data Analysis) we use it for understanding & interpreting how features relate to data classes / categories

Example South African Heart Disease (Hastie et al. 2001)
Data result from Coronary Risk-Factor Study in 3 rural areas of South Africa. Data are from white men 15-64yrs and response is presence/absence of *myocardial infraction (MI).* How predictive are each of the features?



Looking at Data

Each scatterplot shows pair of risk factors. Cases **with** MI (**red**) and **without** (cyan)

Features

- Systolic blood pressure
- Tobacco use
- Low density lipoprotein (ldl)
- Family history (discrete)
- Obesity
- Alcohol use
- Age

[Source: Hastie et al. (2001)]

Example: African Heart Disease

	Coefficient	Std. Error	Z Score
(Intercept)	−4.130	0.964	−4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.939	0.225	4.178
obesity	-0.035	0.029	−1.187
alcohol	0.001	0.004	0.136
age	0.043	0.010	4.184

z-score of a feature: how significant the feature influences the label

(>1.96 usually indicates significant)

Finding Systolic blood pressure (sbp) is **not a significant predictor**

Obesity is not significant and negatively correlated with heart disease in the model

Remember All correlations / significance of features are based on presence of *other features*. We must always consider that features are strongly correlated.

Example: African Heart Disease

w_{tobacco}'s 95% confidence interval:

[0.081-1.96*0.026, 0.081+1.96*0.026]

	Coefficient	Std. Error	Z score
(Intercept)	−4.204	0.498	−8.45
tobacco	0.081	0.026	3.16
ldl	0.168	0.054	3.09
famhist	0.924	0.223	4.14
age	0.044	0.010	4.52

Doing some feature selection we find a model with 4 features: tobacco, ldl, family history, and age

How to interpret coefficients? (e.g. tobacco → 0.081)

- Tobacco is measured in total lifetime usage (in kg) $\ln \frac{p(y=1 \mid x)}{p(y=0 \mid x)} = w^T x$
- Thus, increase of 1kg of lifetime tobacco yields

$$\exp(0.081) = 1.084$$

Or 8.4% increase in odds of coronary heart disease

- 95% CI is 3% to 14% since $\exp(0.081 \pm 2 \times 0.026) = (1.03, 1.14)$

Next lecture

- Nonlinear models: kernel methods
- Assigned reading: CIML Chap. 11