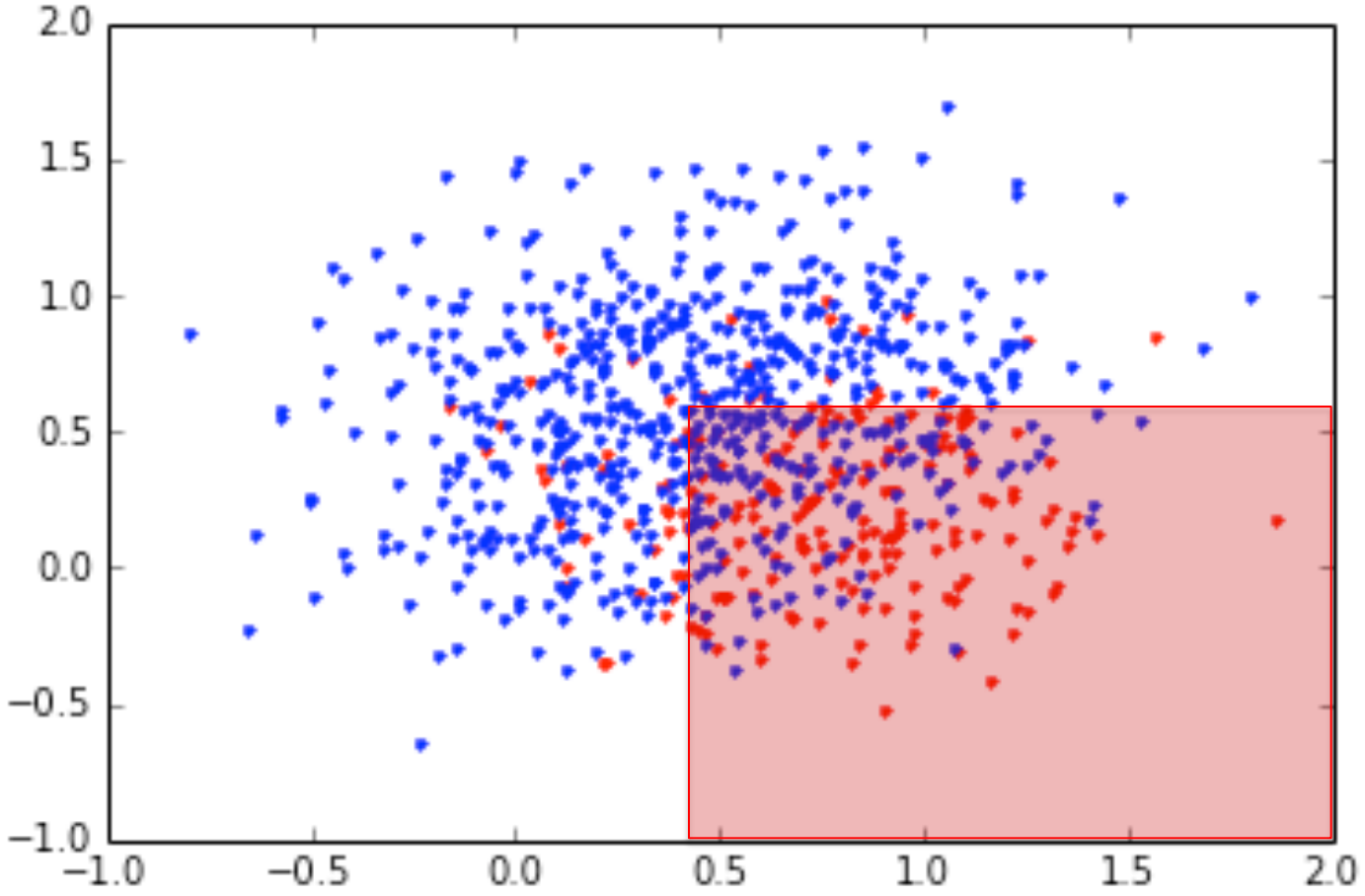
Foundations of Data Science
Lecture 4
Module 2

Rumi Chunara, PhD

CS6053

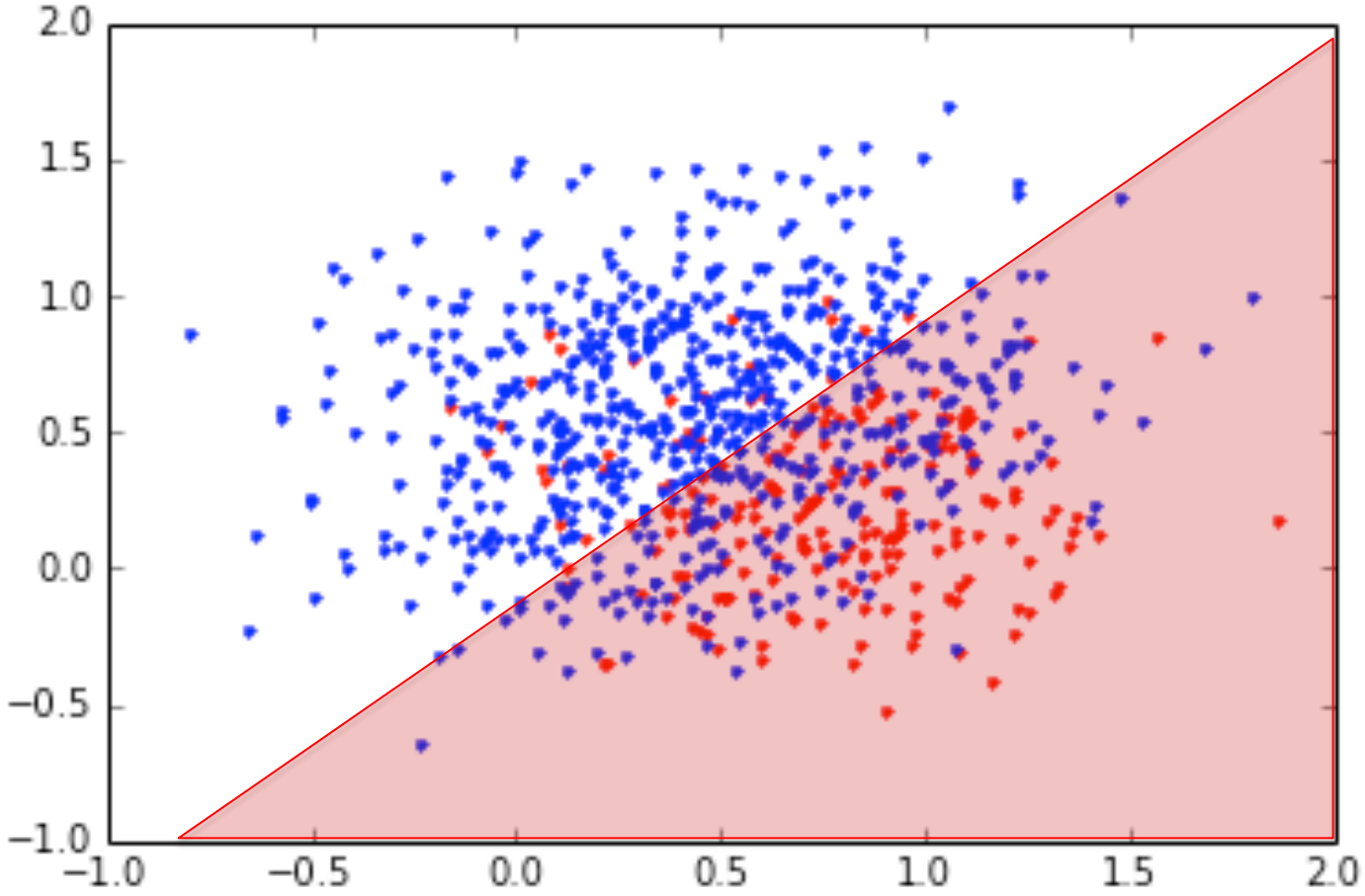
Review – how to classify

Decision Trees use a greedy algorithm to minimize the entropy. It does this by using Boolean operators to partition the feature space.



Linear models

Linear models also partition the feature space, but do so by finding an optimal linear separating hyper-plane, which is found by using principles of statistical learning theory.



Statistical Learning Theory

Let's first set up some notation and ideas:

- Let $X \in \mathbb{R}^p$ be a p-dimensional real valued vector of predictor variables
- Let Y be a target variable, where
  - $Y \in \mathbb{R}$ is real valued
  - $Y \in C$ is an element in some set of classes $C = \{c_1, c_2, \ldots c_k\}$
- X and Y are governed by a joint distribution $P(Y, X)$ (that we likely don't know)
- We seek a function $f(X)$ for predicting Y, given X, whose output can be
  - Real valued, i.e. $f(X) = E[Y|X]$
  - Discrete valued, i.e. $f(x) \in \{c_1, c_2, \ldots c_k\}$

Statistical Learning Theory

Second, let's define two more things

- $\mathbb{F}$ is a family of functions, such that $f(x) \in \mathbb{F}$, examples are:
  - All linear hyper-planes, such that $f(x) = \alpha + \beta x$
  - All quadratic polynomials, such that $f(x) = \alpha + \beta_1 x + \beta_2 x^2$
  - All decision trees with max(depth)=k

- A loss function $\mathbb{L}(f(X), Y)$ that measures how well $f(X)$ approximates Y.
  - Squared Loss: $\mathbb{L}(f(x), y) = (f(x) - y)^2$
  - 0-1 Loss: $\mathbb{L}(f(x), y) = \mathbb{I}(f(x) == y)$
  - Logistic Loss: $\mathbb{L}(f(x), y) = -[y * Ln(f(x)) + (1 - y) * Ln(1 - f(x))]$
  - Hinge Loss: $\mathbb{L}(f(x), y) = max(0, 1 - f(x) * y)$

In this lecture we'll focus on linear models with either Logistic Loss (i.e., logistic regression) or Squared Loss (linear regression).

Statistical Learning Theory

The main goal of Supervised Learning can be stated using the Empirical Risk Minimization framework of Statistical Learning.

We are looking for a function $f \in \mathbb{F}$ that minimizes the expected loss:

$$E[\mathbb{L}(f(x), y)] = \int \mathbb{L}(f(x), y)\, P(x, y)\, \mathrm{d}x\mathrm{d}y$$

Because we don't know the distribution $P(X, Y)$, we can't minimize the expected loss. However, we can minimize the empirical loss, or risk, by computing the average loss over our training data.

Thus, in Supervised Learning, we choose the function $f(X)$ that minimizes the loss over training data:

$$f^{opt} = \underset{f \in \mathbb{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(f(x_i), y_i)$$

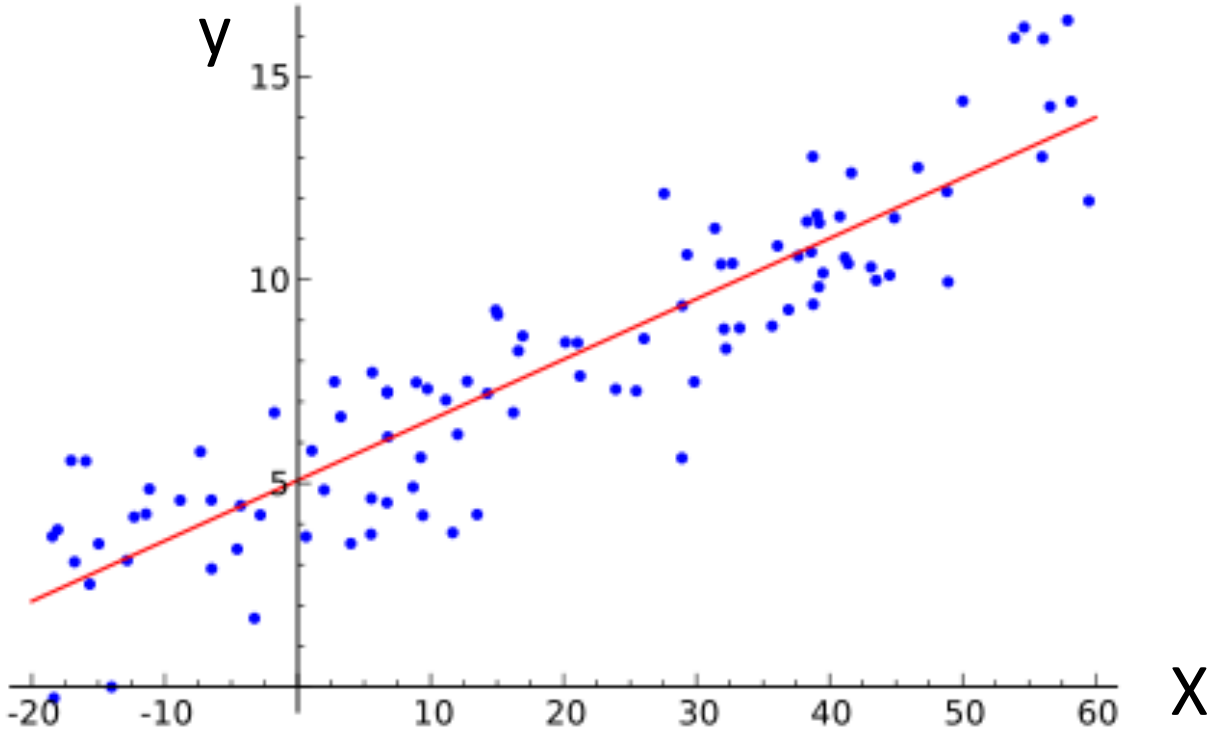
This concept of Empirical Risk Minimization will the basis for understanding the linear models presented later, as well as for understanding the very important principle of bias-variance tradeoffs.

Linear Regression

- LR: a very simple approach for supervised learning.
- LR useful for predicting a quantitative response.
- Been around for a while, useful for large data sets also (quick)

Linear Regression

We want to find the best line (linear function y=f(X)) to explain the data.



Linear Regression

The predicted value of y is given by:

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

The vector of coefficients $\hat{\beta}$ is the regression model.

If $X_0 = 1$, the formula becomes a matrix product:

$$\hat{y} = \mathrm{X}\,\hat{\beta}$$

Linear Regression in Matrix Form

- Consider writing an equation for each observation: $Y_1 = \beta_0 + \beta_1 X_1 + \varepsilon_1$

- Model in Matrix Form:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \dots & \dots \\ 1 & X_3 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_n \end{bmatrix}$$

Residual Sum-of-Squares

To determine the model parameters $\hat{\beta}$ from some data, we can write down the Residual Sum of Squares:

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

or symbolically $\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$. To minimize it, take the derivative wrt β which gives:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

And if $\mathbf{X}^T\mathbf{X}$ is non-singular, the unique solution is:

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

Stochastic Gradient

A very important set of iterative algorithms use **stochastic gradient** updates.

They use a **small subset or mini-batch X** of the data, and use it to compute a gradient which is added to the model

$$\beta' = \beta + \alpha \nabla$$

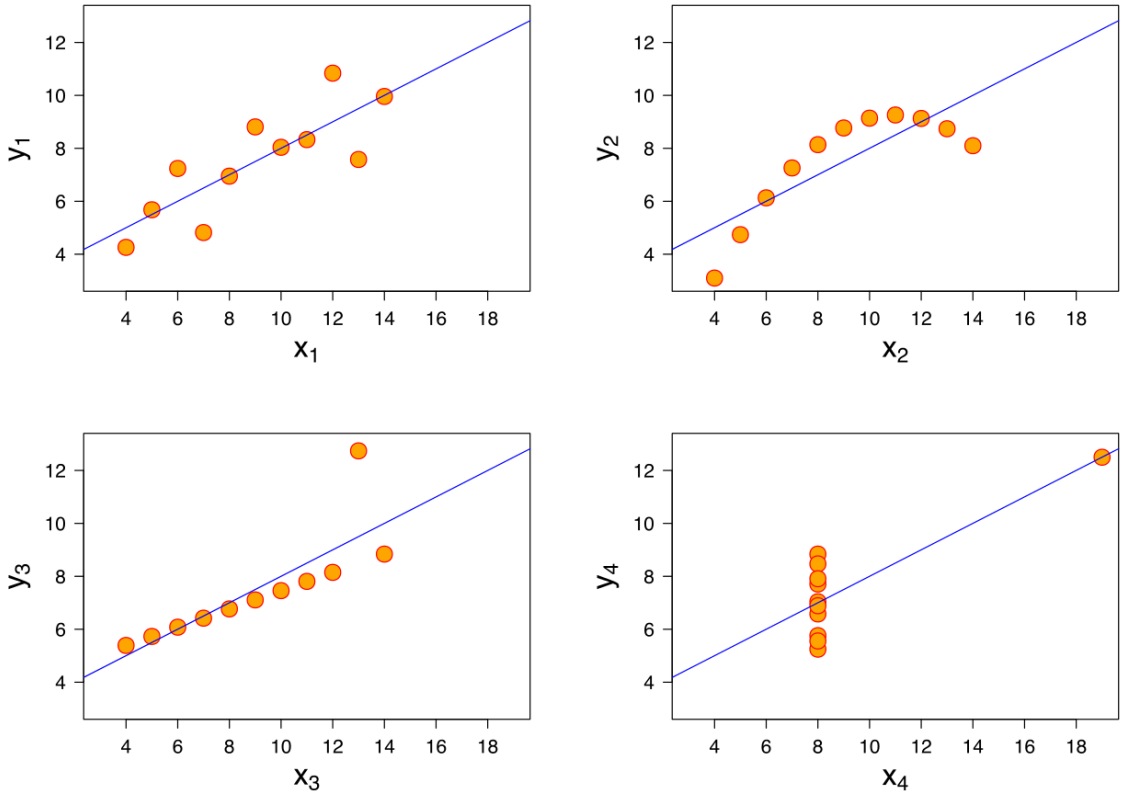
Where α is called the **learning rate**.

These updates happen **many times** in one pass over the dataset.

Its possible to compute high-quality models with very few passes, sometime with less than one pass over a large dataset.

R^2-values and P-values

We can **always** fit a linear model to any dataset, but how do we know if there is a **real linear relationship**?



R^2-values and P-values

Approach: Use a hypothesis test. The null hypothesis is that there is no linear relationship ($\beta = 0$).

Statistic: Some value which should be small under the null hypothesis, and large if the alternate hypothesis is true.

R-squared: a suitable statistic. Let $\hat{y} = \mathrm{X}\,\hat{\beta}$ be a predicted value, and $\bar{y}$ be the sample mean. Then the R-squared statistic is

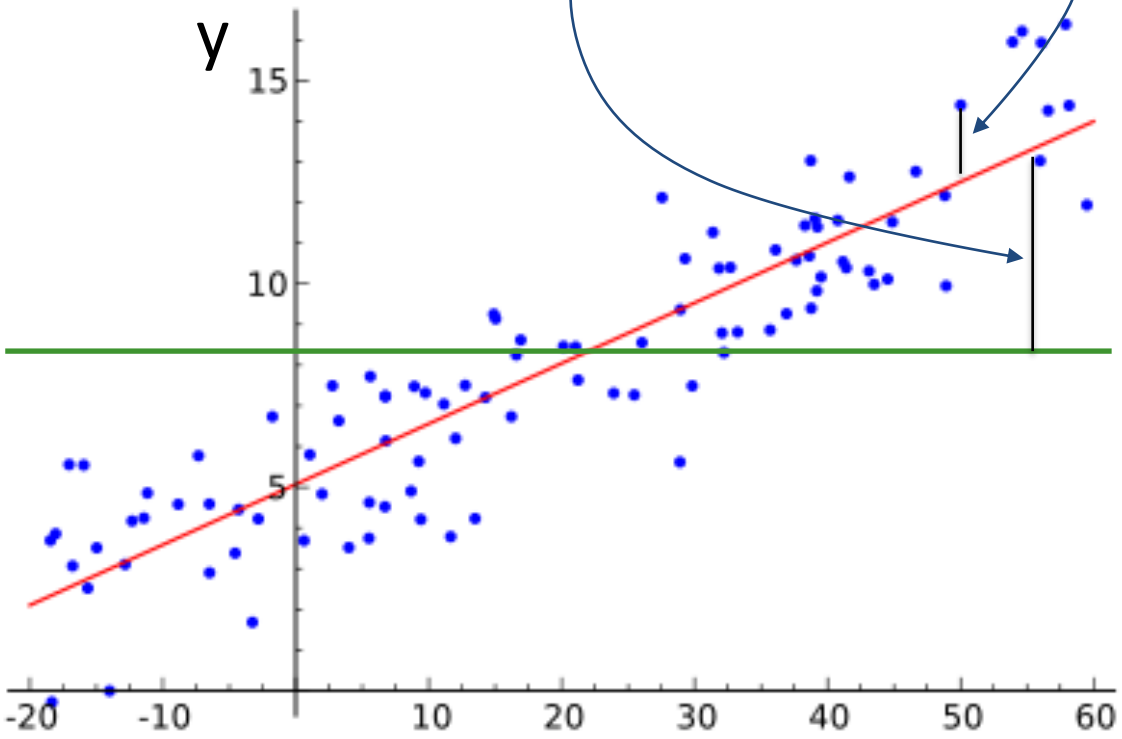
$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

And can be described as the fraction of the total variance not explained by the model.

R-squared

$$R^2 = 1 - \frac{RSS}{TSS} \qquad = 1 - \frac{\sum_i (y_i - \hat{y})^2}{\sum_i (y_i - \bar{y})^2}$$

Small if good fit



Line of $\hat{y}$

Line of $\bar{y}$

R^2-values and P-values

Statistic: From R-squared we can derive another statistic (using degrees of freedom) that has a standard distribution called an **F-distribution**.

From the CDF for the F-distribution, we can derive a **P-value** for the data.

$$F = \frac{(TSS - RSS)}{(p-1) * RSS * (n-p)}$$

The P-value is, as usual, the probability of observing the data under the null hypothesis of no linear relationship.

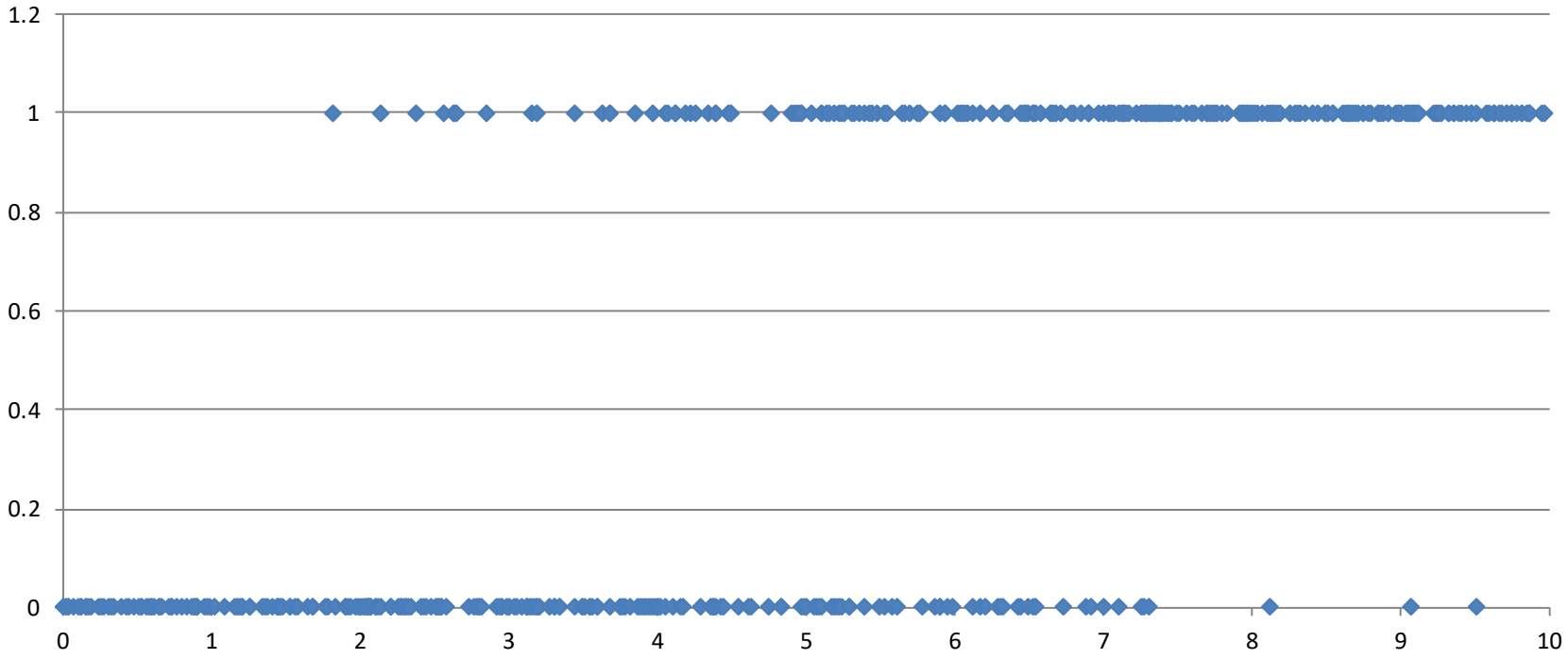
If **p is small**, say less than 0.05, we conclude that **there is a linear relationship**.

Logistic Regression

A Binary Regression?

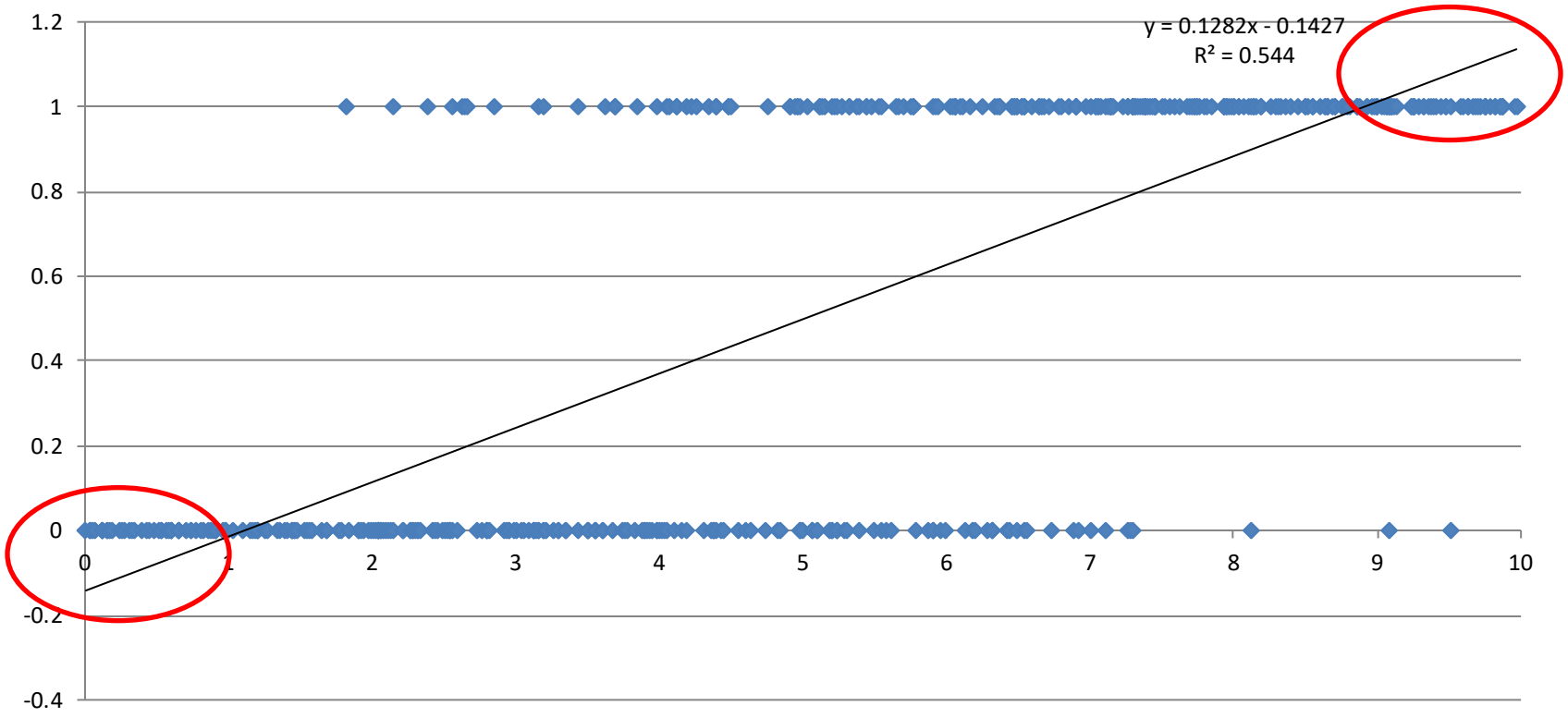
Logistic Regression was created as a binary extension of Ordinary Least Squares Regression.

What linear function fits this data?



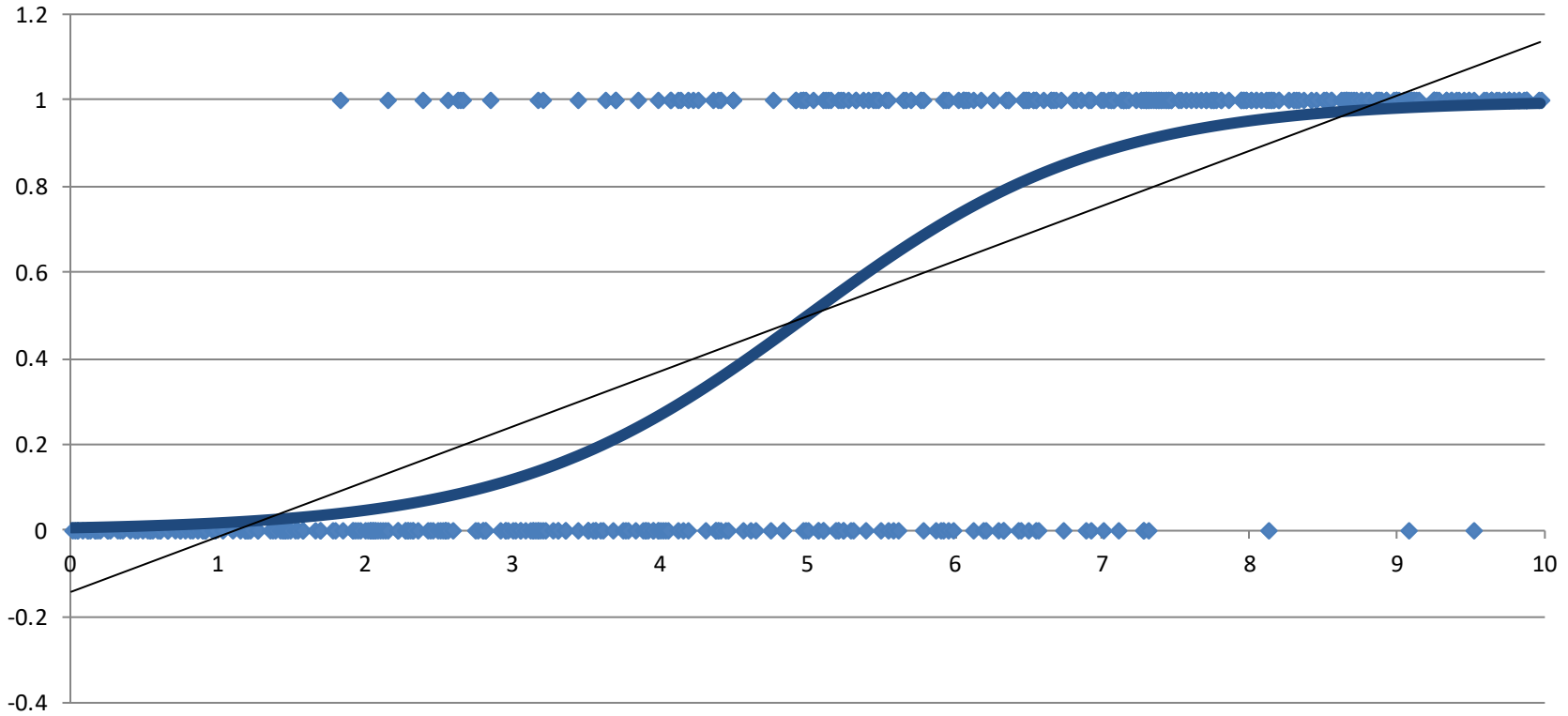
Linear Least Squares – Not So Good

The linear least squares curve gives us a reasonable fit in certain areas, but is not constrained to the interval [0,1]. This is bad when you want to estimate a probability.



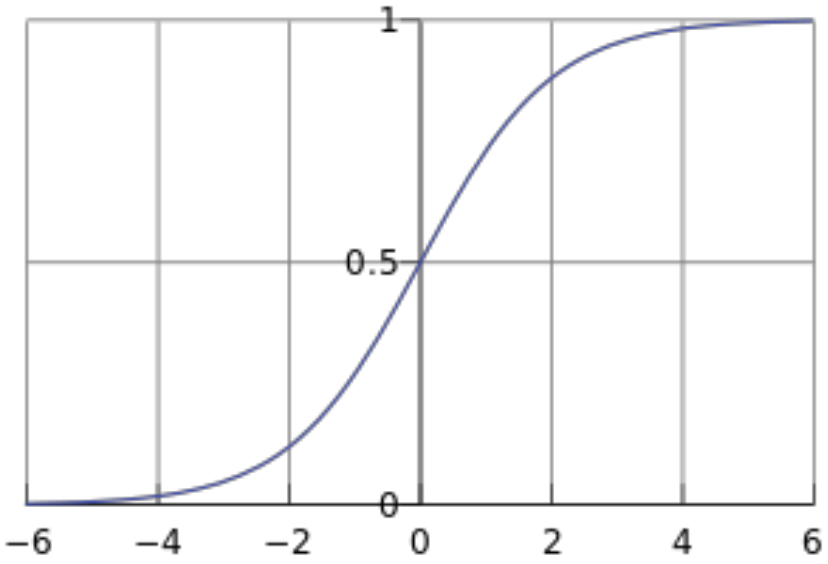
Something Better?

It would be better if we had some function that was linear in its parameters, but behaved better as a probability estimator.



The Inverse Logit

The inverse logit is just the function we are looking for.



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

Logistic Regression

Logistic Regression: a member of the class of generalized linear models (glm) using the logit as its link function.

The goal of Logistic Regression is to model the posterior probability of membership in class c_i as a function of X. I.e.,

$$P(c_i|x) = f(x) = \frac{1}{1+e^{-(\alpha+\beta x))}}$$

To make this a linear model in X, we take the log of the odds ratio of p (called the log-odds):

$$ln\ \frac{P(c_i|x)}{1-P(c_i|x)} = ln\ \frac{1}{e^{-(\alpha+\beta x))}} = \alpha + \beta x$$

And effectively we do a linear regression against the log-odds of $P(c_i|x)$ (though we don't use least squares).

Logistic Regression as ERM

How do we fit Logistic Regression into the ERM framework?

We find the parameters α and β using the method of Maximum Likelihood Estimation.

If we consider each observation to be an indepedent Bernoulli draw with $p_i = P(y_i|x_i)$, then the likelihood of each draw can be defined as: $p_i^{y_i}(1-p_i)^{1-y_i}$, with p_i given by the inverse logit function. In MLE, we wish to maximize the likelihood of observing the data as a function of the independent parameters of the model (i.e., α and β). The total likelihood function looks like:

$$L(\alpha, \beta|X, Y) = \prod_{i=1}^{n} P(x_i, y_i|\alpha, \beta) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

This is actually a difficult equation to maximize directly, so we do a little trick. We take the negative log and call this our loss function for ERM!

$$\mathbb{L}(f(X), Y) = -Ln[L(\alpha, \beta|X, Y)] = -\sum_{i=1}^{n} y_i \, ln\,(p_i) + (1-y_i)\, ln\,(1-p_i)$$

LR: stats vs. Machine Learning?

MLE is the method traditionally described in statistics (finding the model that best fits the data) while ERM is used in machine learning (empirically learning from data). An advantage of "statistical" point of view is that we can do hypothesis testing on the parameter estimates of the model.

	coef	std err	z	P>|z|	[95.0% Conf. Int.]
isbuyer	0.8421	0.562	1.499	0.134	-0.259 1.943
buy_freq	0.0588	0.397	0.148	0.882	-0.720 0.838
visit_freq	0.0469	0.026	1.828	0.067	-0.003 0.097
buy_interval	0.0320	0.020	1.591	0.112	-0.007 0.071
sv_interval	-0.0047	0.010	-0.488	0.626	-0.024 0.014
expected_time_buy	-0.0337	0.025	-1.372	0.170	-0.082 0.014
expected_time_visit	-0.0241	0.009	-2.801	0.005	-0.041 -0.007
last_buy	0.0038	0.006	0.634	0.526	-0.008 0.016
last_visit	-0.0518	0.006	-8.636	0.000	-0.064 -0.040
multiple_buy	-0.6713	1.099	-0.611	0.541	-2.825 1.482
multiple_visit	0.0493	0.278	0.177	0.859	-0.495 0.593
uniq_urls	-0.0107	0.002	-5.147	0.000	-0.015 -0.007
num_checkins	-6.112e-05	0.000	-0.481	0.631	-0.000 0.000

Model Statistical Analysis

coef – the estimate for β

std err – the standard error for the estimate of β

z – the z-score for hypothesis testing on the estimate of β

P>|z| – the p-value (prob of type 1 error) for asserting that β!= 0.

[95% Conf Int] – the 95% conf. interval for the estimate of β

Interpreting Betas

A Practical Aside

What exactly does the estimate of β really mean? How can we interpret it?

Recall that $Ln\ \frac{p}{1-p} = \alpha + \beta x$. This means that a unit change in the value of x changes the log-odds by the value of β.

So what can we learn by looking at betas?

Some helpful tips, garnered from theory and experience:

- $|\beta_1| > |\beta_2|$ does not guarantee that feature X_1 is more predictive than X_2. The magnitude of β is inversely proportional to the scale of X, so comparing betas only makes sense when the features have the same scale (such as binary features).

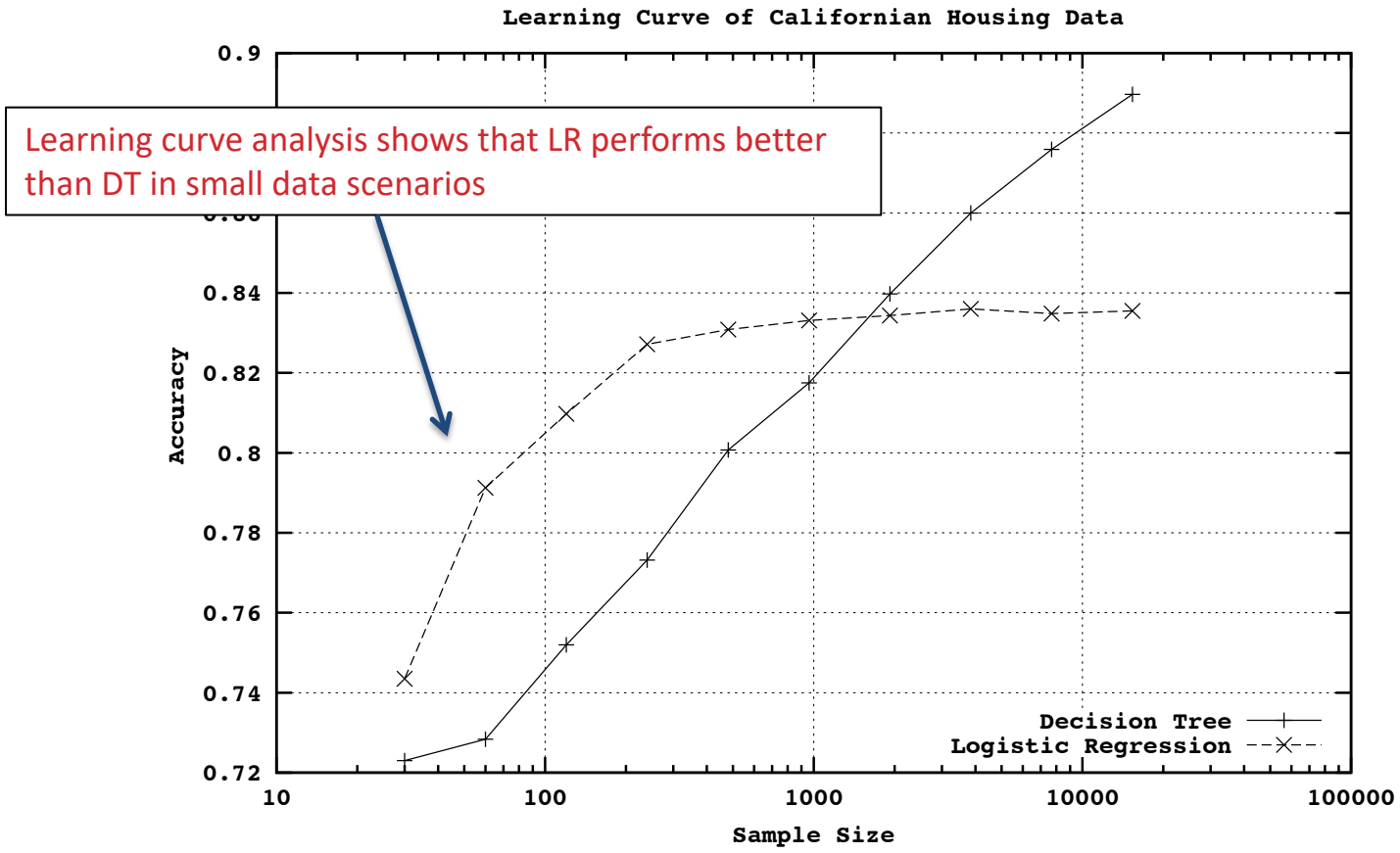
- Likewise, the z-score of β is influences by sample size and should not be used to rank features by predictiveness

- sign(β) does tell you whether Y is positively or negatively correlated with X. However, if the features have a lot of multi-collinearity, sign(β) can be misleading.

- Multi-collinearity in X means the betas will have covariance with each other. The betas will "split" the effect. Sometimes they'll split the effect as positive numbers (i.e. 1=0.5+0.5) and other times they'll split as negatives (i.e., 1=2-1). This makes interpreting β that much more difficult.

Robustness of LR

Not all scenarios involve “Big Data.” Logistic Regression has been proven over and over to be very robust in small data problems.



Source: Tree Induction vs. Logistic Regression, a Learning Curve Analysis. Perlich, Provost and Simonoff

Today

- Intro to ML – what is it
- Two Basic Algorithms
  - Linear Regression
  - kNN