Introduction to Data Science

BRIAN D'ALESSANDRO

ADJUNCT PROFESSOR, NYU

FALL 2016

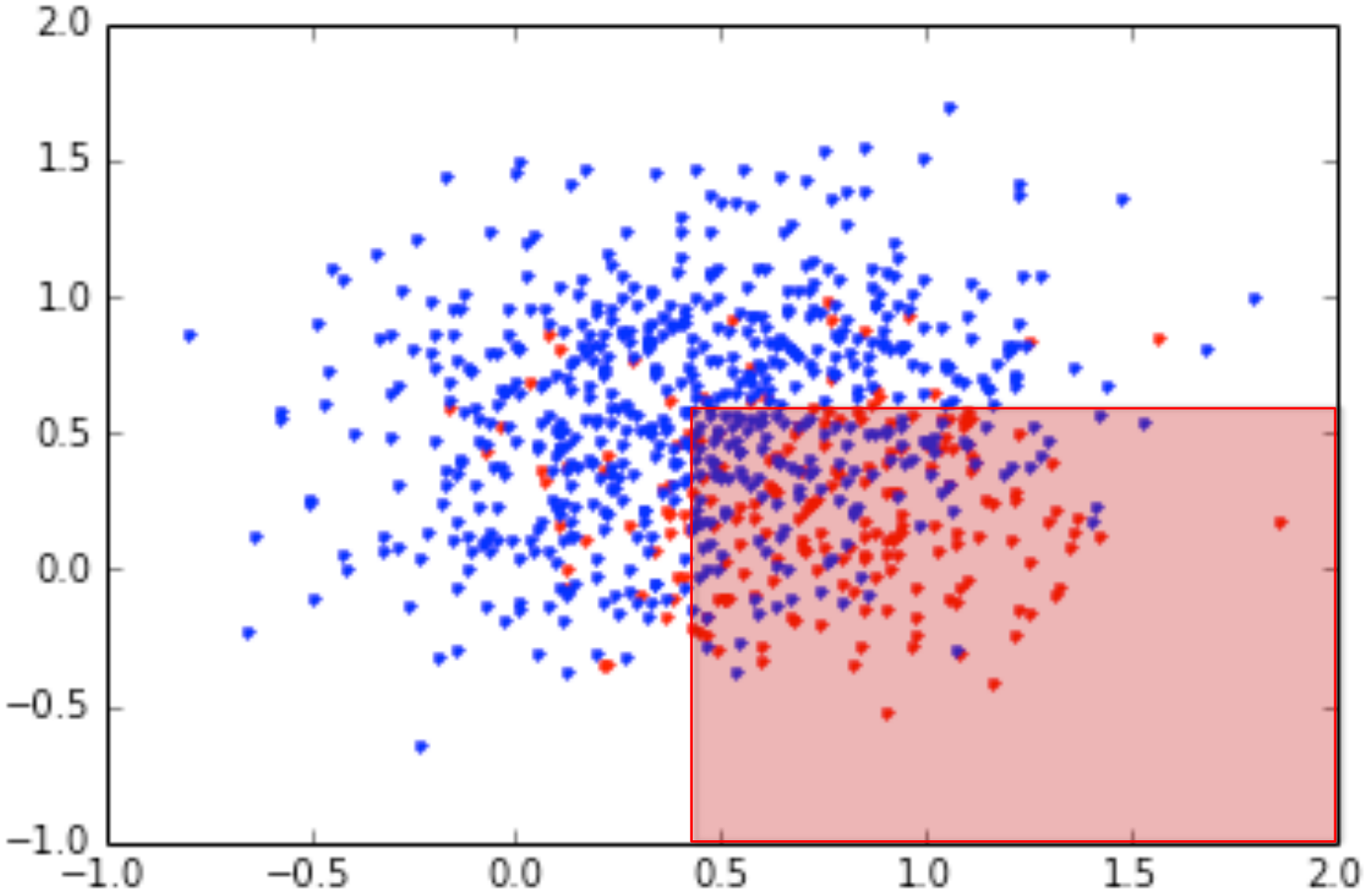
Fine Print: these slides are, and always will be a work in progress. The material presented herein is original, inspired, or borrowed from others' work. Where possible, attribution and acknowledgement will be made to content's original source. Do not distribute, except for as needed as a pedagogical tool in the subject of Data Science.

LINEAR MODELING:
STATISTICAL LEARNING THEORY
LOGISTIC REGRESSION
SUPPORT VECTOR MACHINES

NYU – Intro to Data Science

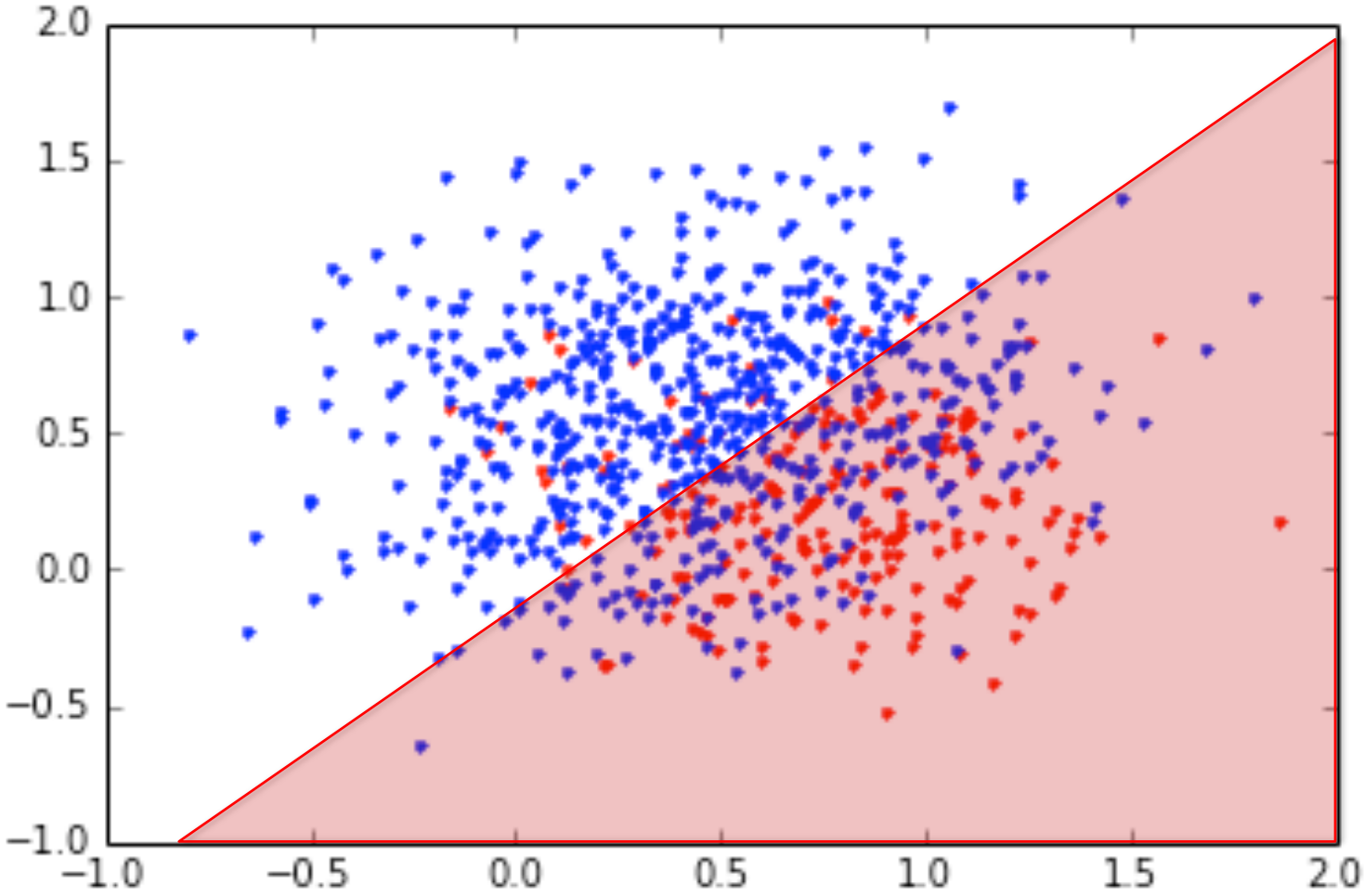
REVIEW – HOW TO CLASSIFY

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 Hinge Loss (Support Vector Machine).

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:

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

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:

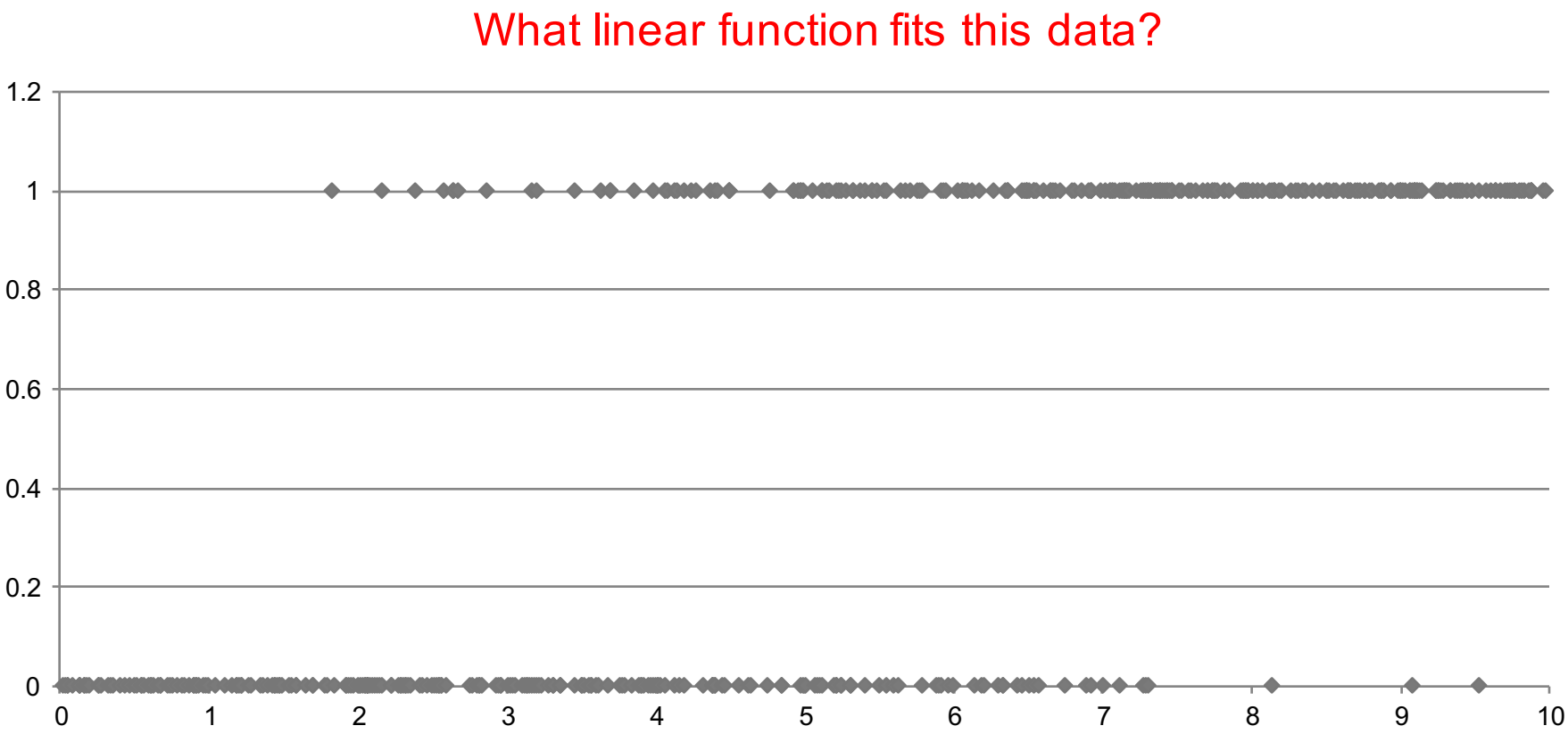
$$f^{opt} = \underset{f \in \mathbb{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(f(x_i), y_i)$$

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.

LOGISTIC REGRESSION

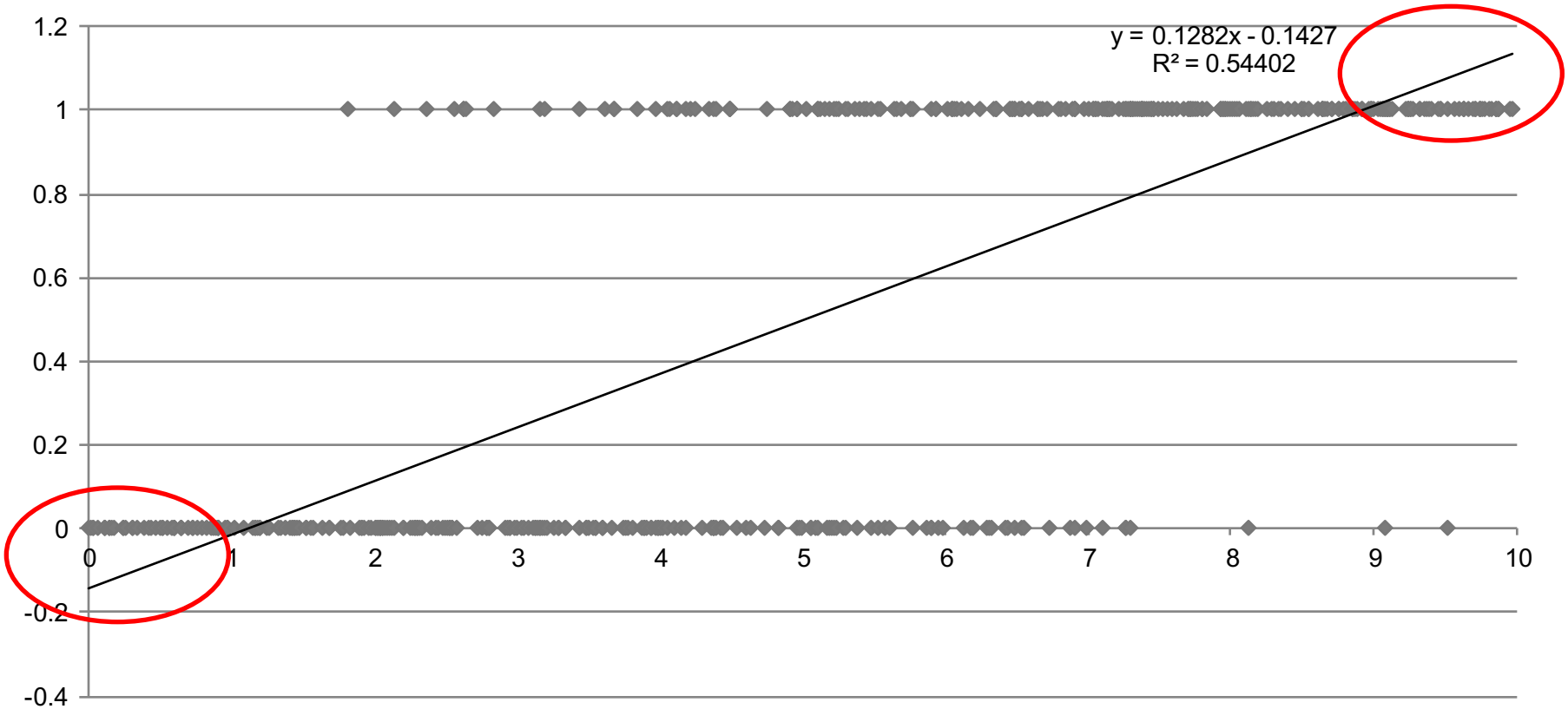
A BINARY REGRESSION?

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



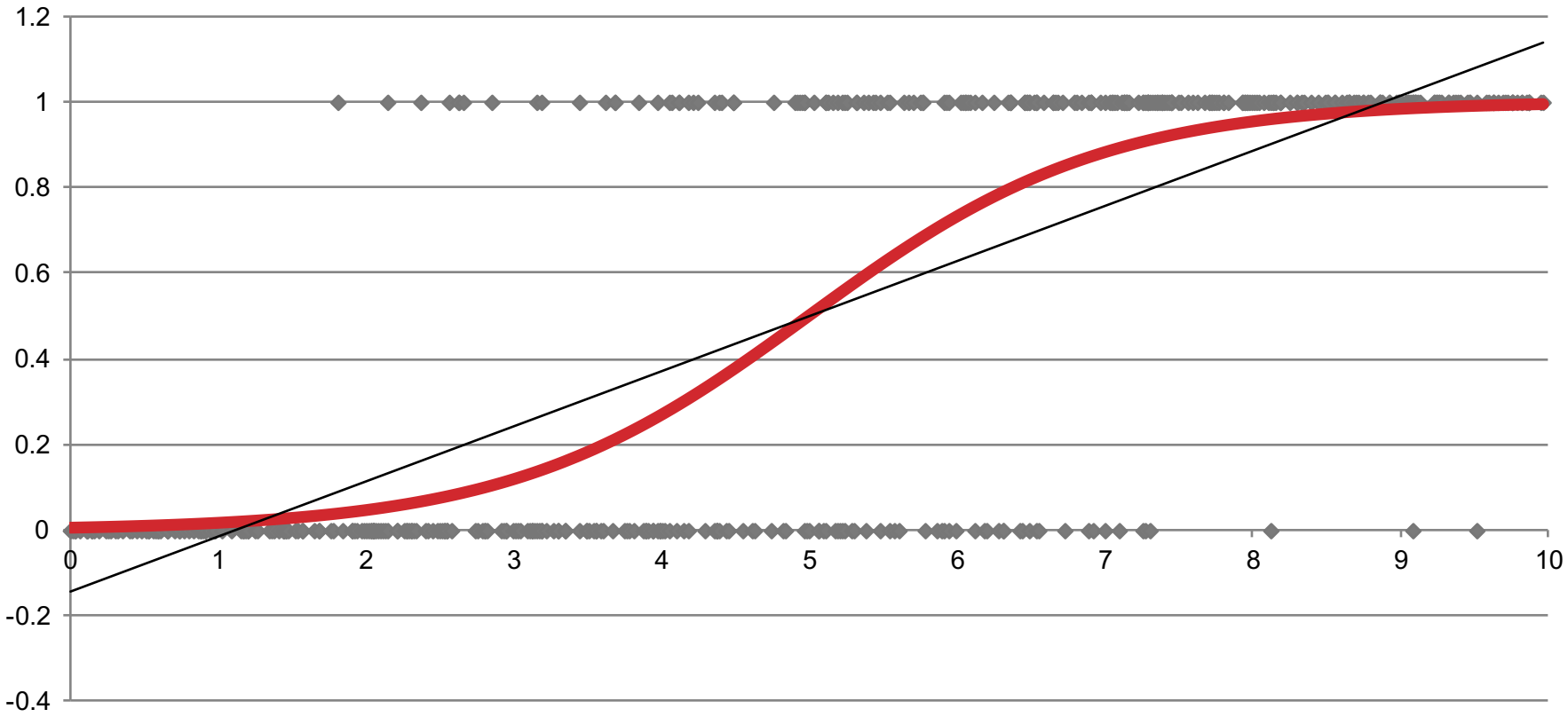
LINEAR LEAST SQUARES – NOT SO GOOD

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.



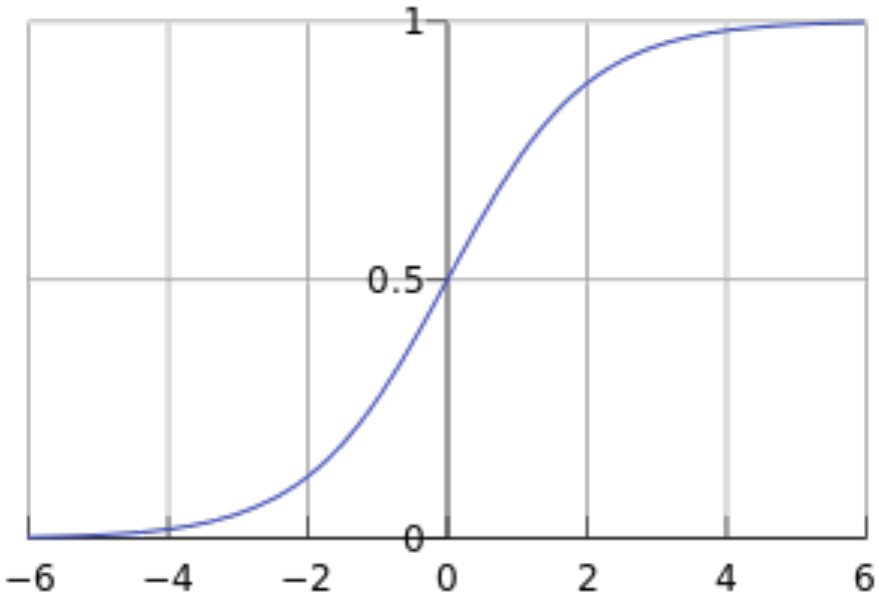
SOMETHING BETTER?

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 a method used traditionally in statistics while ERM is used in machine learning. Logistic Regression works in both disciplines. 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

Note: I used pythons statsmodel package as opposed to scikit-learn to get this summary. See accompanying ipython notebook for examples.

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. This is a mathematical statement that IMHO does not offer much intuitive value.

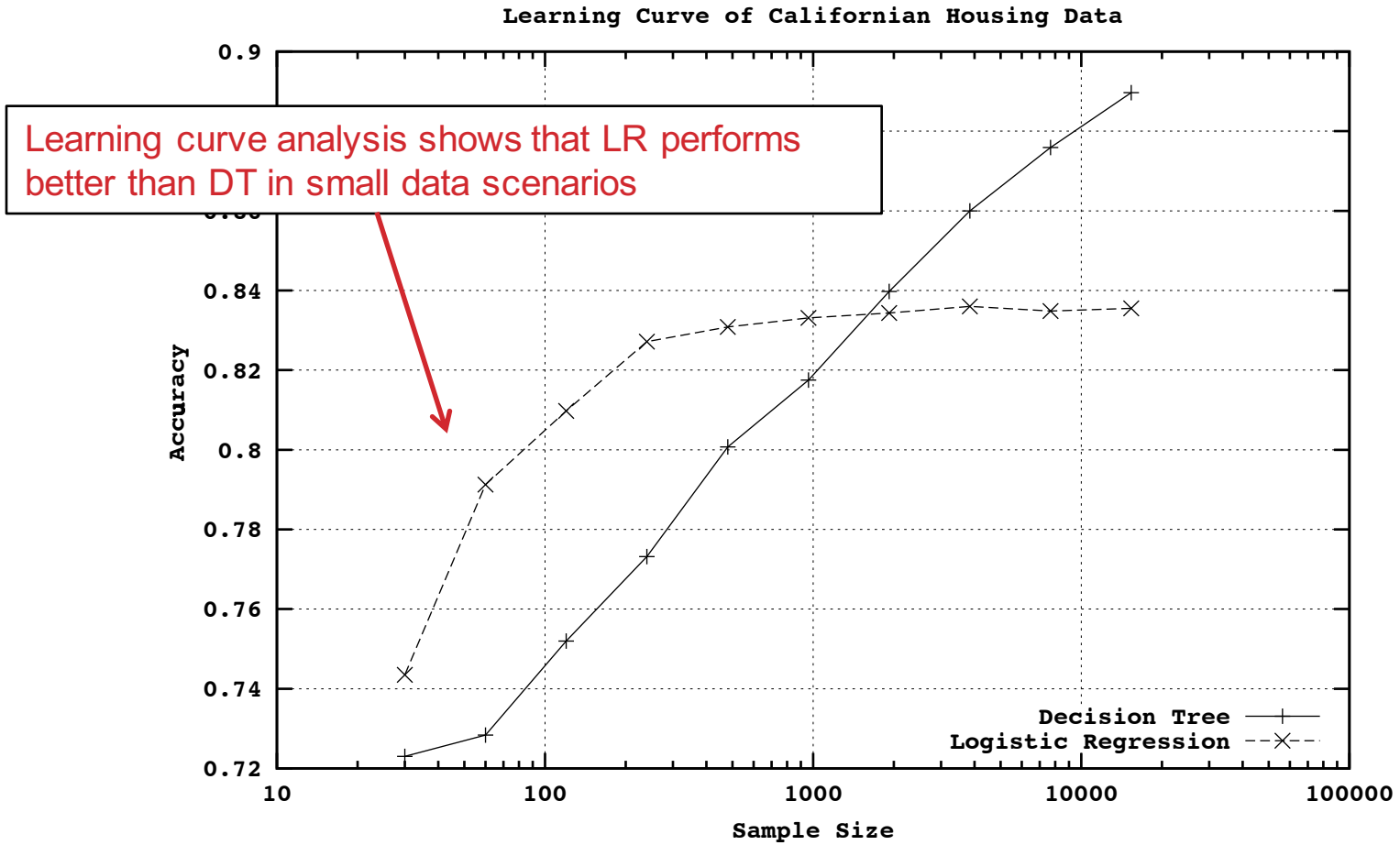
So what can we learn by looking at betas? (IMHO, not much!)

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.



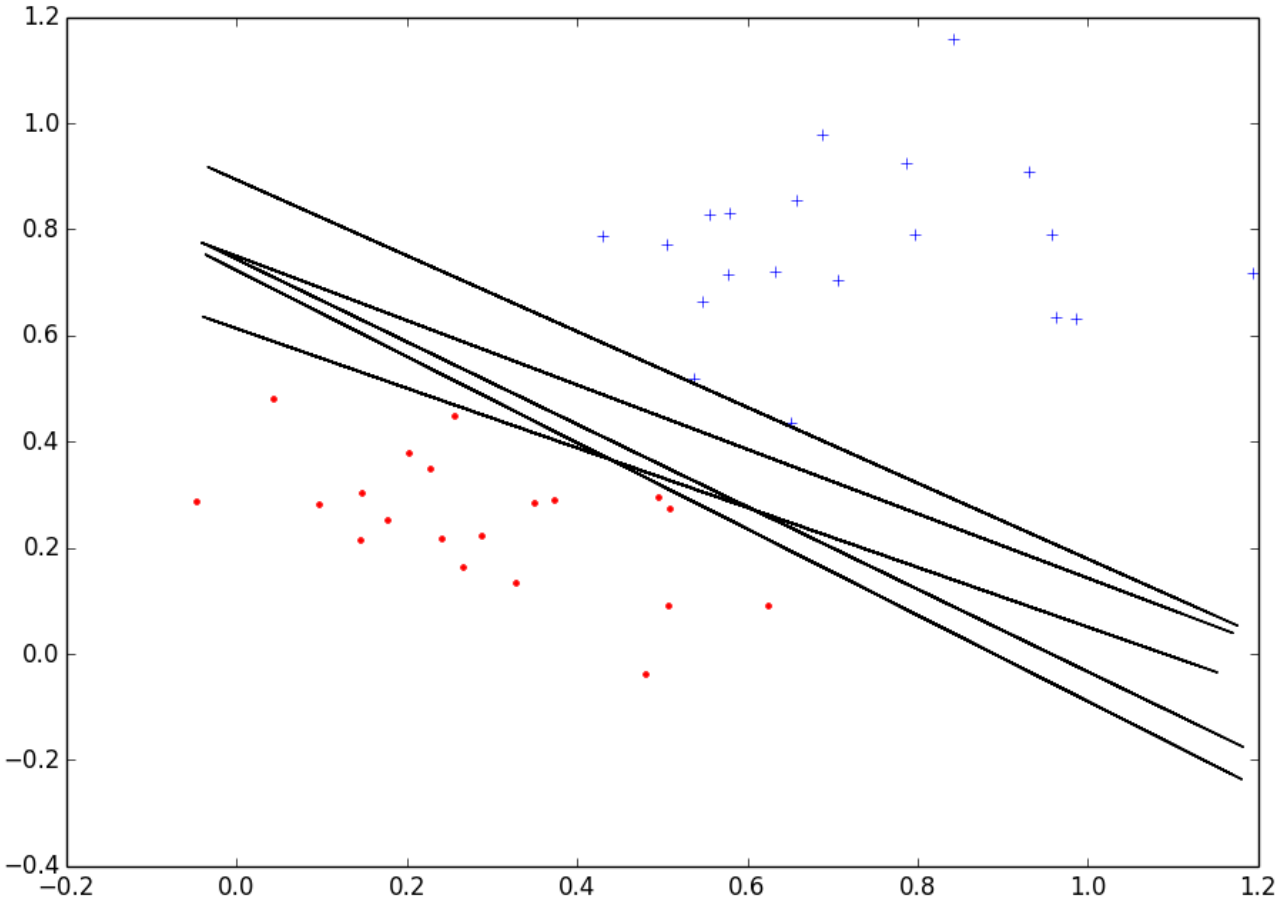
Source: Tree Induction vs. Logistic Regression, a Learning Curve Analysis. Perlich, Provost and Simonoff

SUPPORT VECTOR MACHINES

SEPARATING HYPERPLANES

When data is separable, there is often more than one hyper-plane that can perfectly separate the data.

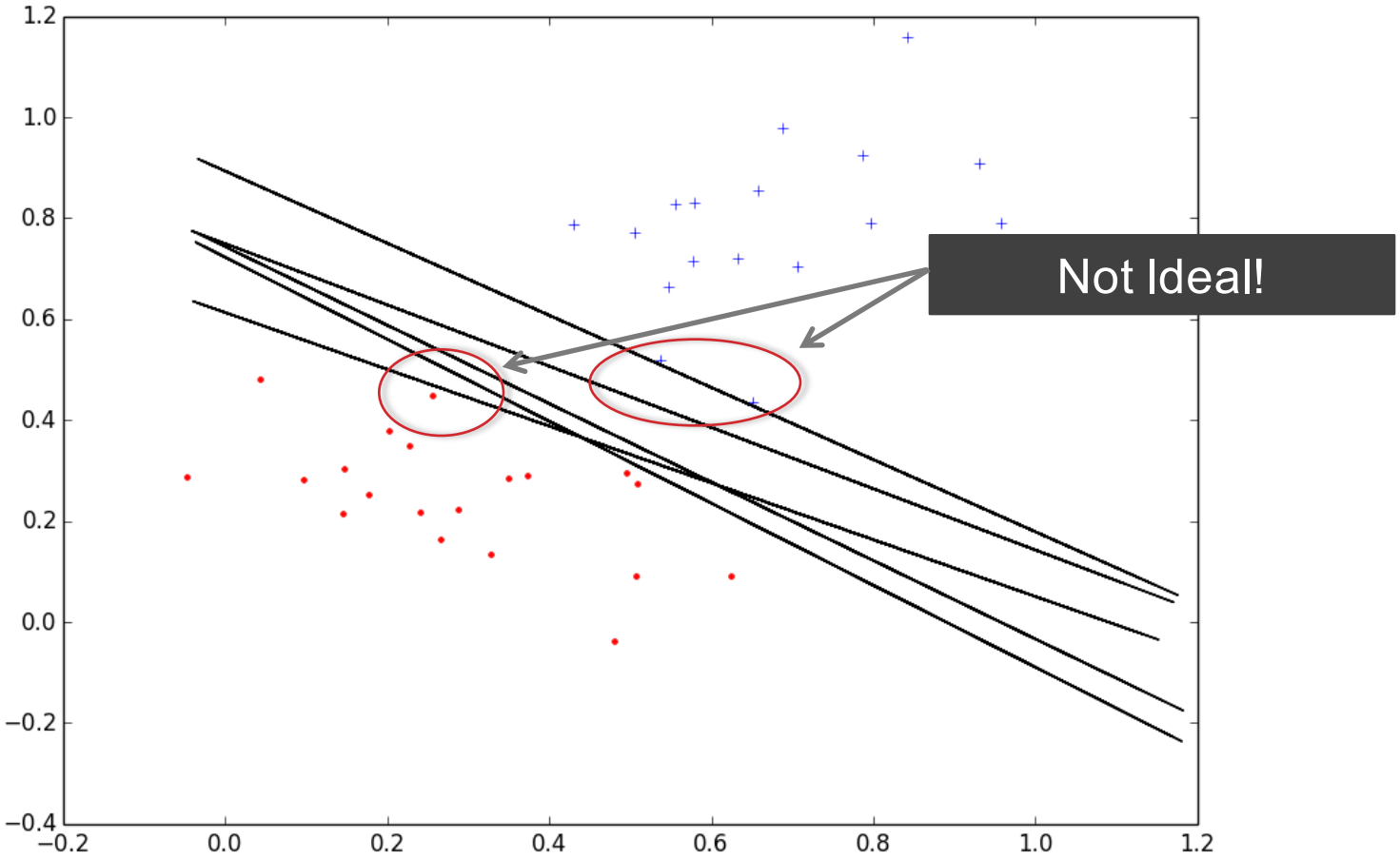
Given so many choices, it would be nice to have a principled way to choose an optimal one.



SEPARATING HYPERPLANES

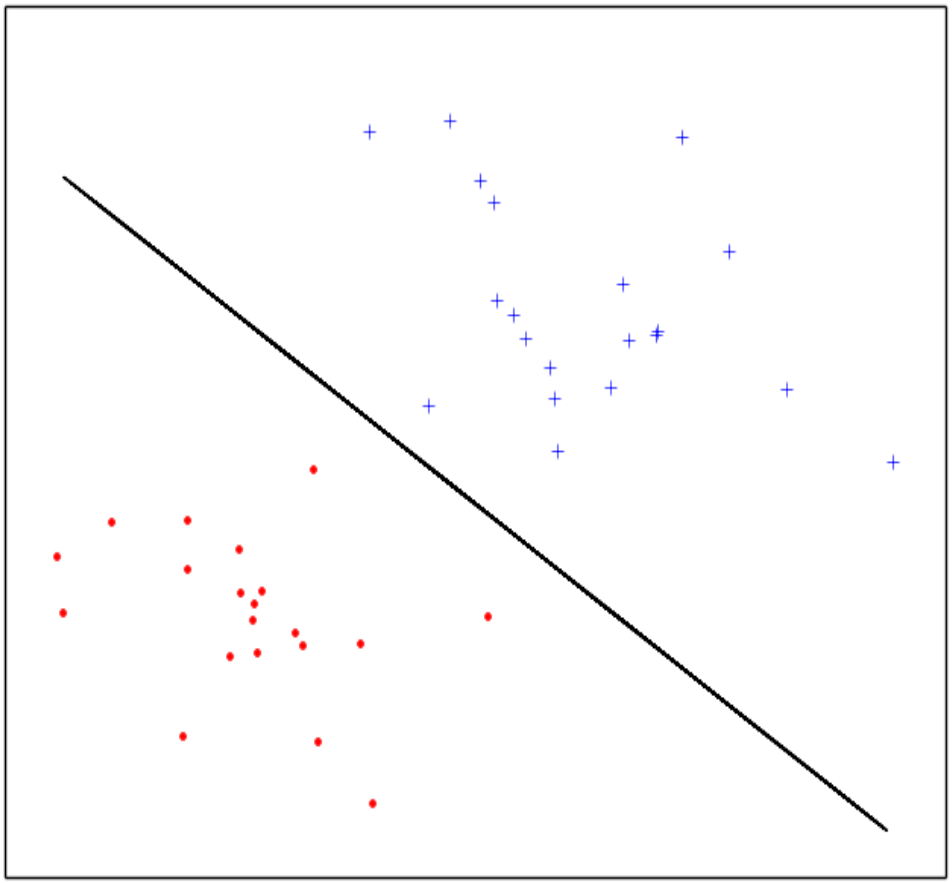
Ideally, we want a line that is as far from all points as possible.

A safe bet would be a line that sits as close to the middle of the empty separating the two classes



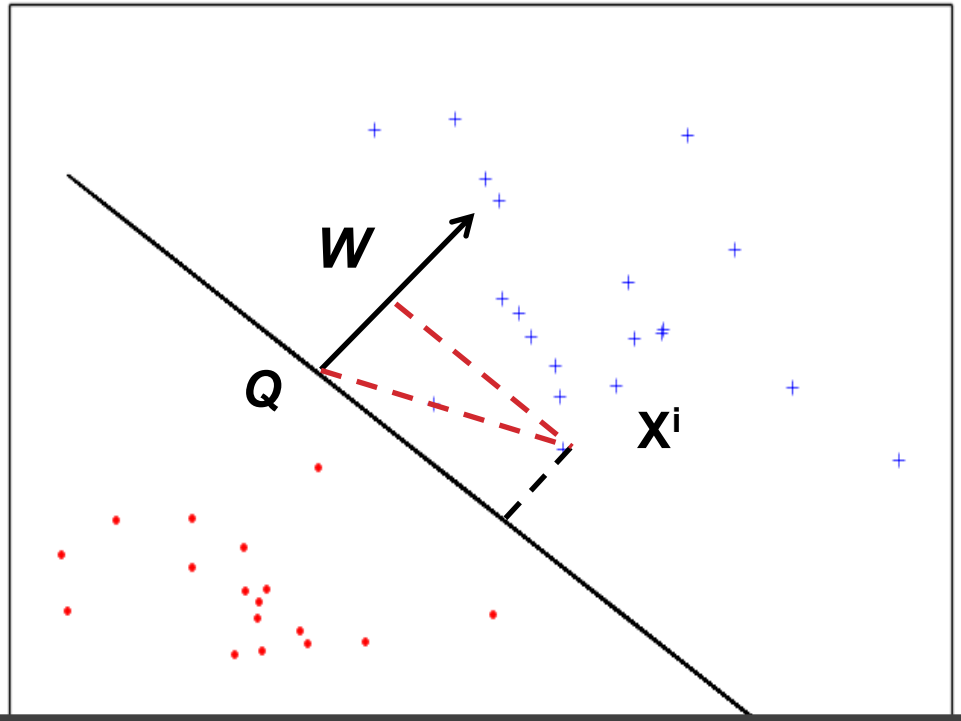
MAXIMAL MARGIN HYPERPLANE

We can formalize our intuition of searching for a plane that is maximally far from as many points as possible.



SOME FUNDAMENTALS

We'll start with some basic linear algebra



The distance of X^i to the line is given by the projection of the segment QX^i onto W and is given by:

We have a feature space:

$$X =< x_1, ...x_k >$$

A hyper-plane:

$$W \cdot X + t = 0$$

A normal vector:

$$W =< w_1, ...w_k >$$

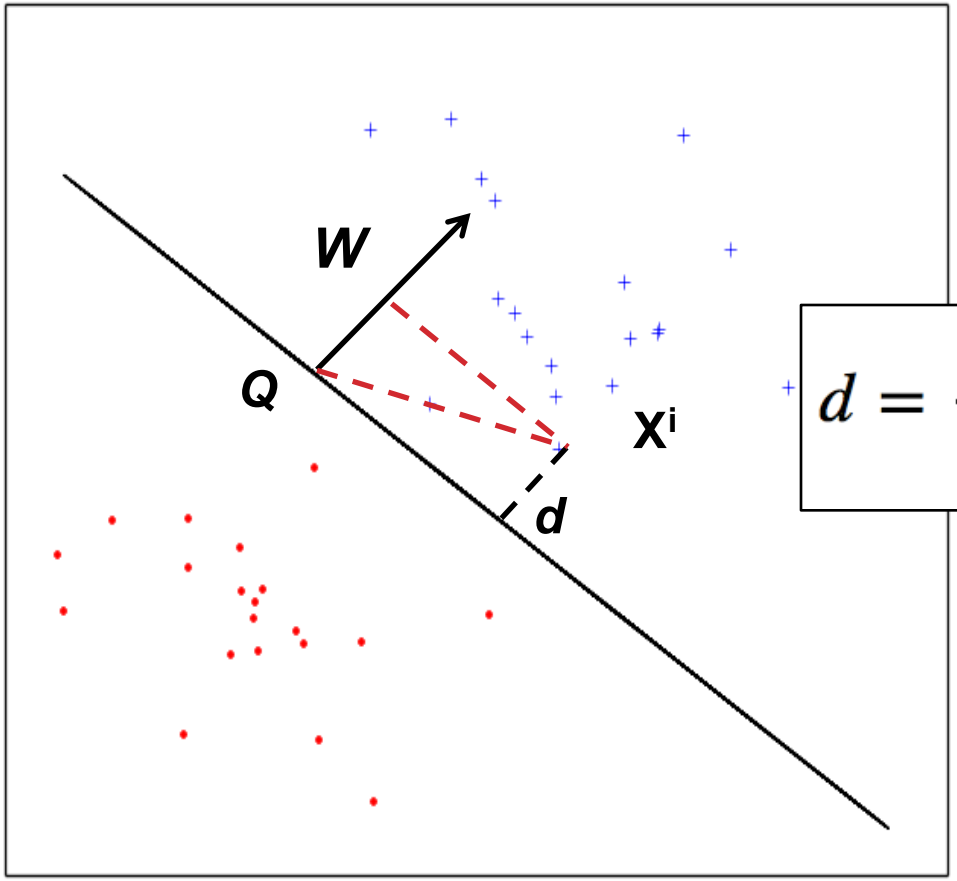
Distance(X^i, plane)

$$\frac{W \cdot X^i + t}{\|W\|_2}$$

DISTANCE TO THE PLANE

How did we get that distance?

If we define a point ***Q*** on the line, then the distance from $\mathbf{X^i}$ to the line is equal to the orthogonal projection of the vector $\boldsymbol{QX^i}$ to the normal vector ***W***.

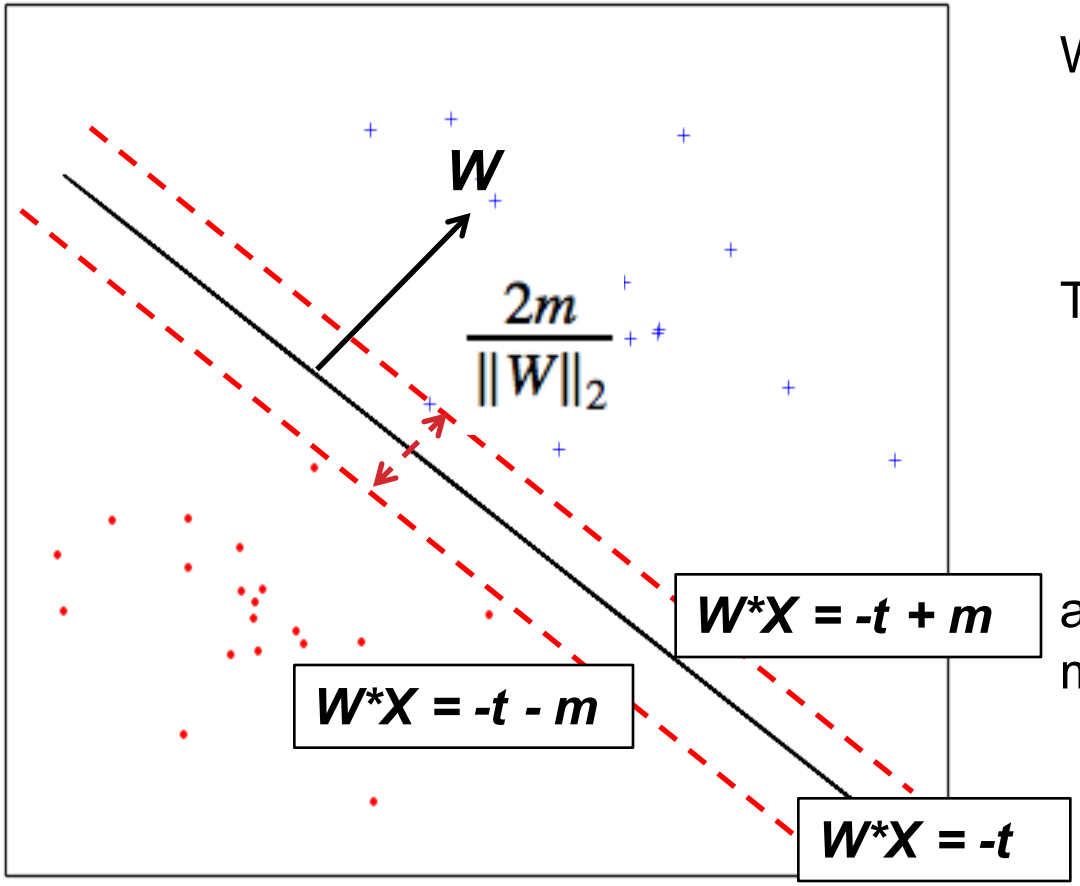


$$d = \frac{\overrightarrow{QX^i} \cdot W}{\|W\|_2} = \frac{W \cdot X^i - W \cdot Q}{\|W\|_2} = \frac{W \cdot X^i + t}{\|W\|_2}$$

ADDING CONSTRAINTS

If we add some constraints, the problem becomes easier to solve.

M^+ = distance of closest positive point to W
M^- = distance of closest negative point to W



With the constraint:

$$M^+ = M^- = m = 1$$

The margin then becomes:

$$\frac{2m}{\|W\|_2}$$

and this is what we seek to maximize.

THE OPTIMIZATION PROBLEM

Maximizing: $\frac{2m}{\|W\|_2}$ is equivalent to the following optimization problem:

$$W^*, t^* = \underset{W,t}{\operatorname{argmin}} \frac{1}{2} \|W\|_2^2 \quad \text{subject to } y_i(W * X_i - t) \geq 1$$

- The constraints ensure that all points are on the correct side of the line.
- This can be solved with Convex Optimization and the method of Langrange Multipliers (beyond the scope of this class)

DUAL FORM

- There is a “dual-form” that is equivalent to searching for the support vectors – i.e., the points on the lines WX = -t +/- m

$$\underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j X_i \cdot X_j$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

- For most input points, $a_i = 0$, leading to a sparse solution composed only of the ‘support vectors’ (those points on the margin).

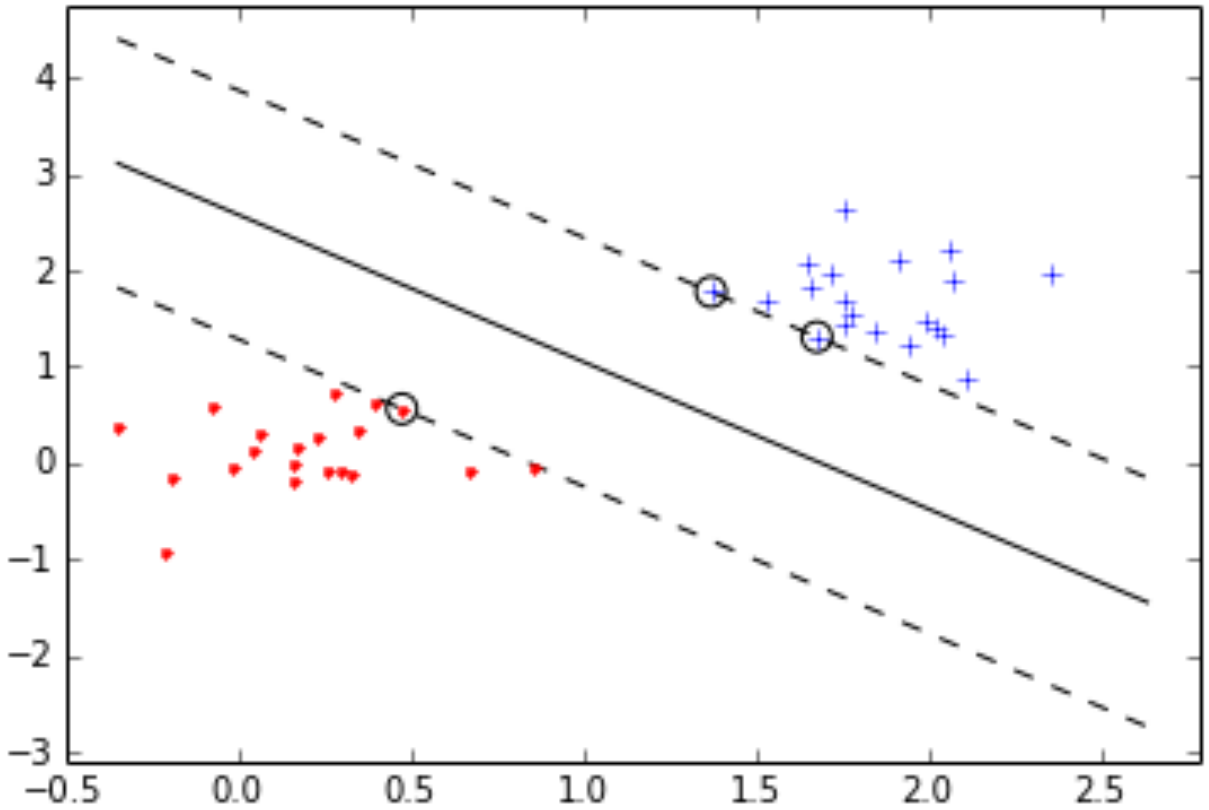
$$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i.$$

ET VOILA!

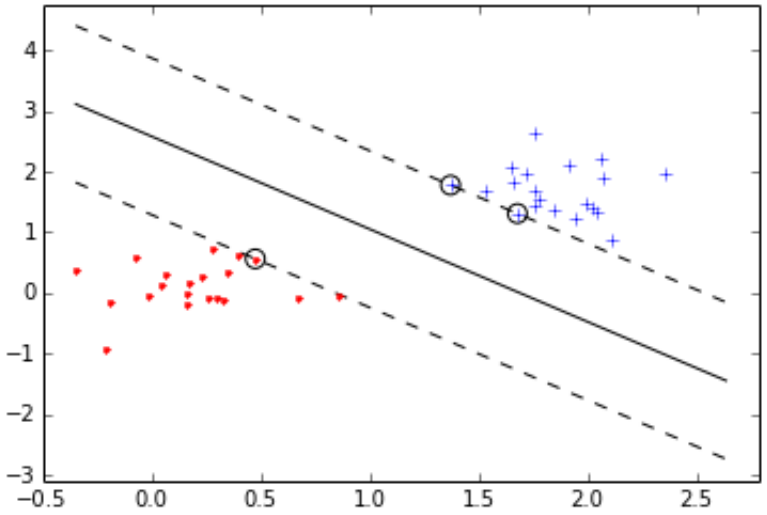
An optimal separating hyper-plane along with margin planes and support vectors identified.

Classification of a new point X then becomes: f(X) = sign(W*X + t).

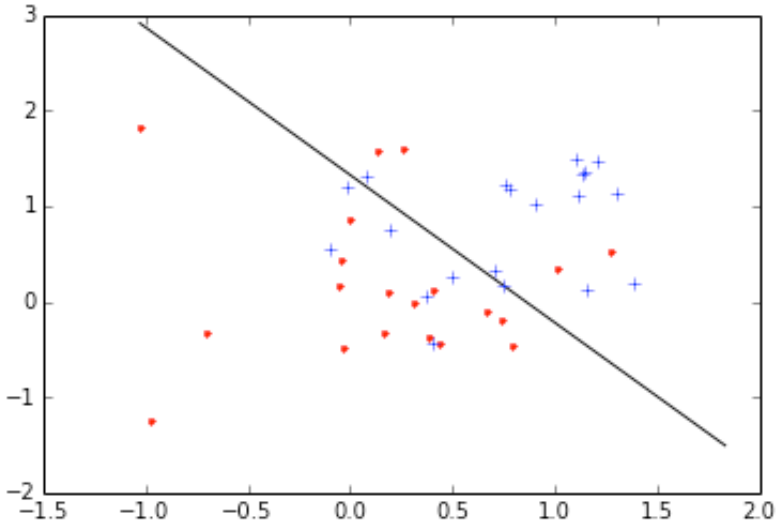
Effectively, we are finding which side of the line a new point is on.



REALITY



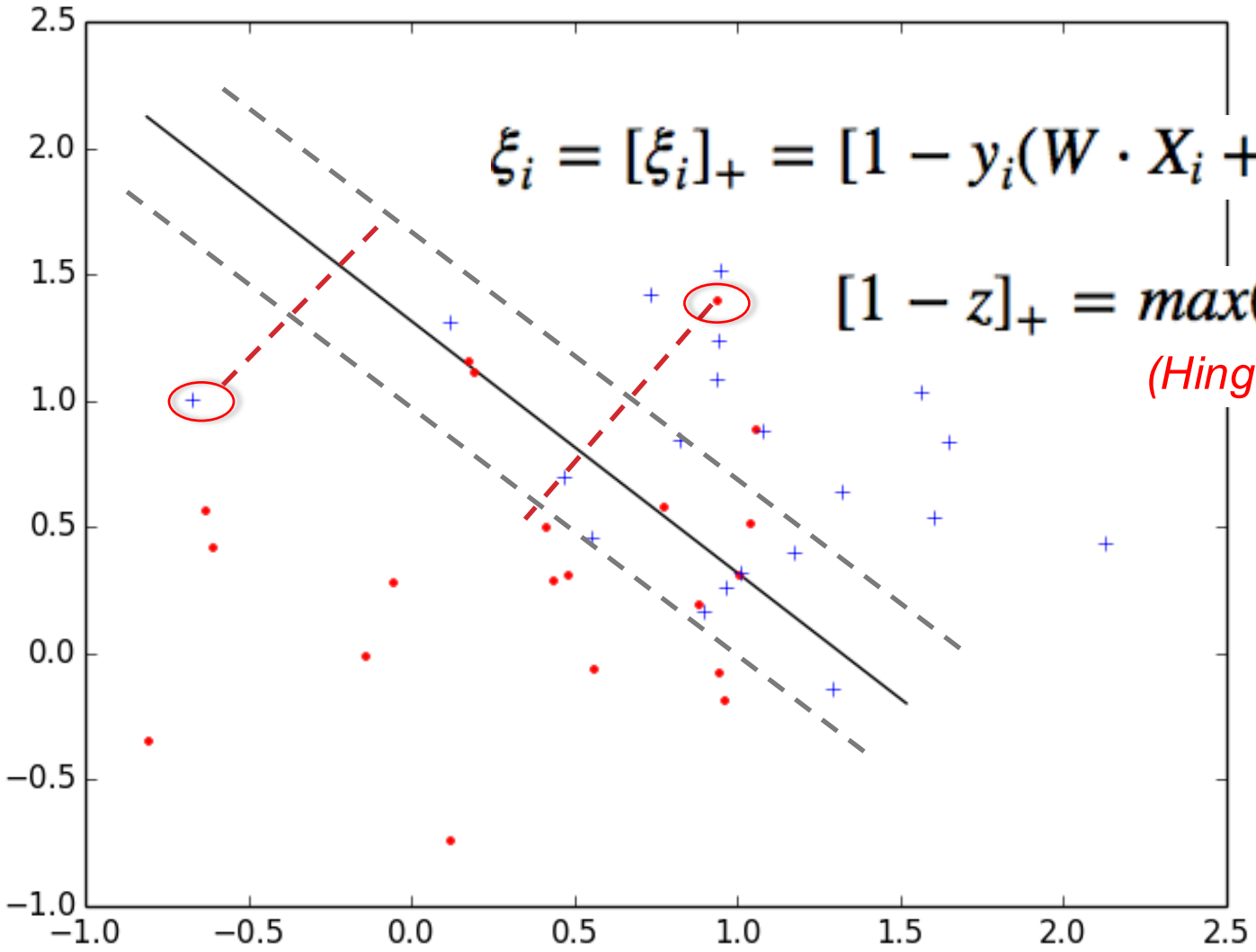
This is somewhat of an unrealistic scenario. Data is seldom this cleanly separated.



Real data looks more like this.

INTRODUCING SLACK VARIABLES

We can account or errors in our problem by introducing slack variables, which measure distance of a misclassified points to the margin.



$$\xi_i = [\xi_i]_+ = [1 - y_i(W \cdot X_i + t)]_+$$

$$[1 - z]_+ = max(0, 1 - z)$$

(Hinge Loss)

SOFT MARGIN SVM

Our optimization problem now can be written:

$$W^*, t^* = \underset{W,t}{\operatorname{argmin}} \frac{1}{2} \|W\|_2^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i(W \cdot X_i + t) \geq 1 - \xi_i$$

Which can also be expressed as:

$$W^*, t^* = \underset{W,t}{\operatorname{argmin}} \lambda \|W\|_2^2 + \sum_{i=1}^{n} [1 - y_i(W \cdot X_i + t)]_+$$

Where $\lambda = 1/2C$

SOFT MARGIN SVM AS ERM

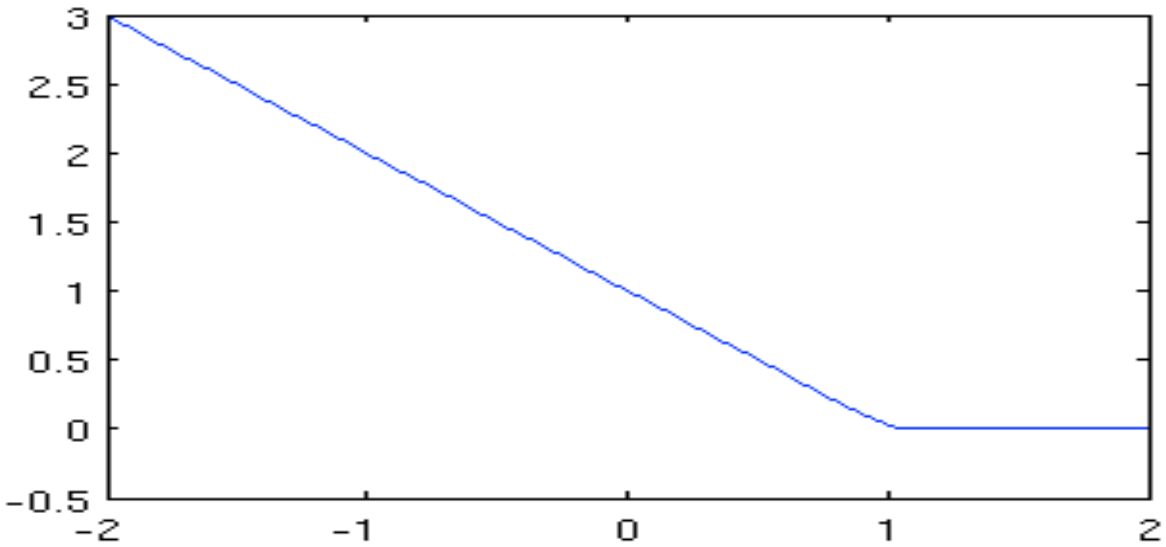
With this formulation we are trying to do two things:

$$W^*, t^* = \underset{W,t}{\operatorname{argmin}} \lambda \|W\|_2^2 + \sum_{i=1}^{n} [1 - y_i(W \cdot X_i + t)]_+$$

Maximize the margin

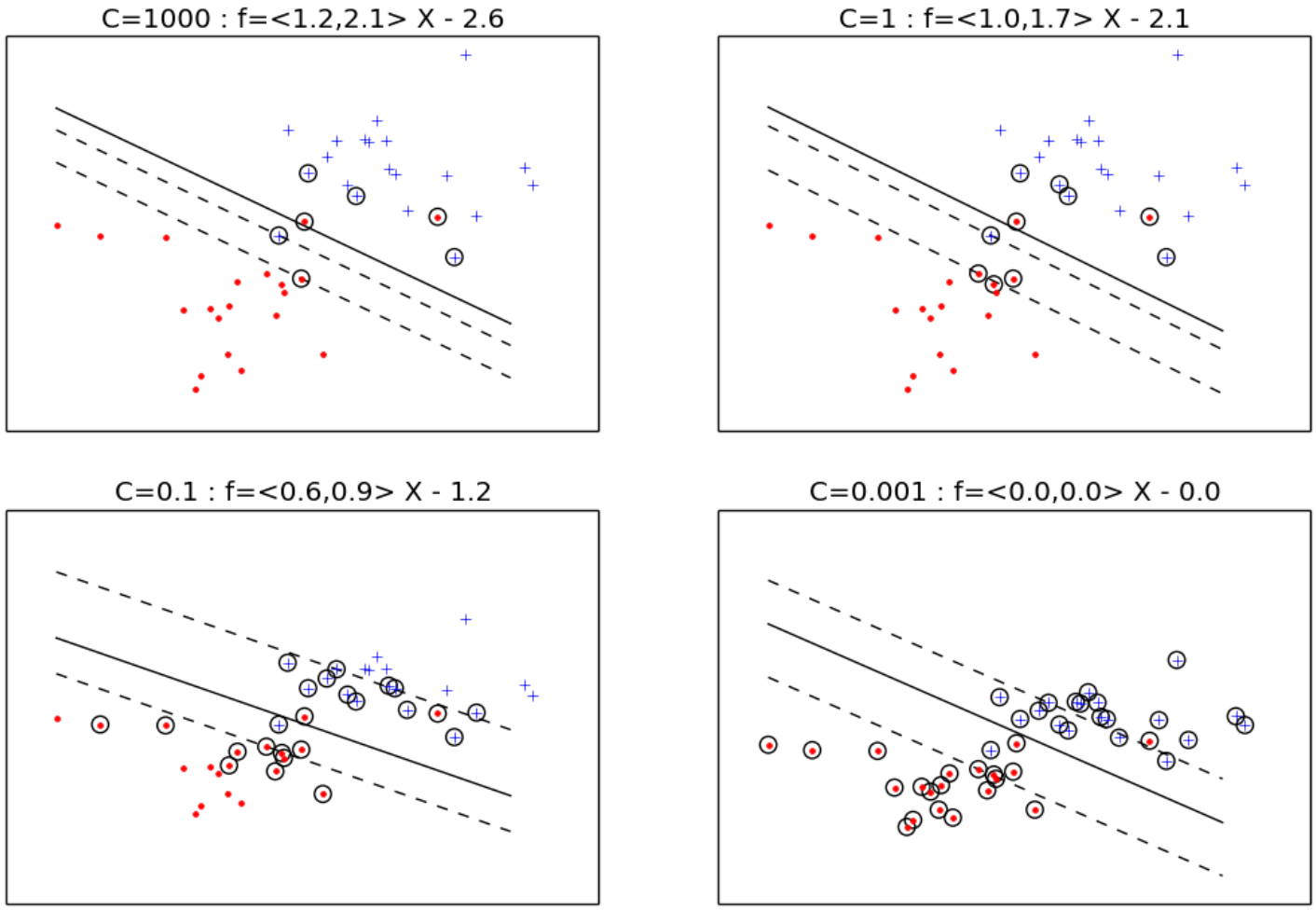
Minimize the training error (hinge loss)

Example Hinge Loss



CONTROLLING ERROR TOLERANCE

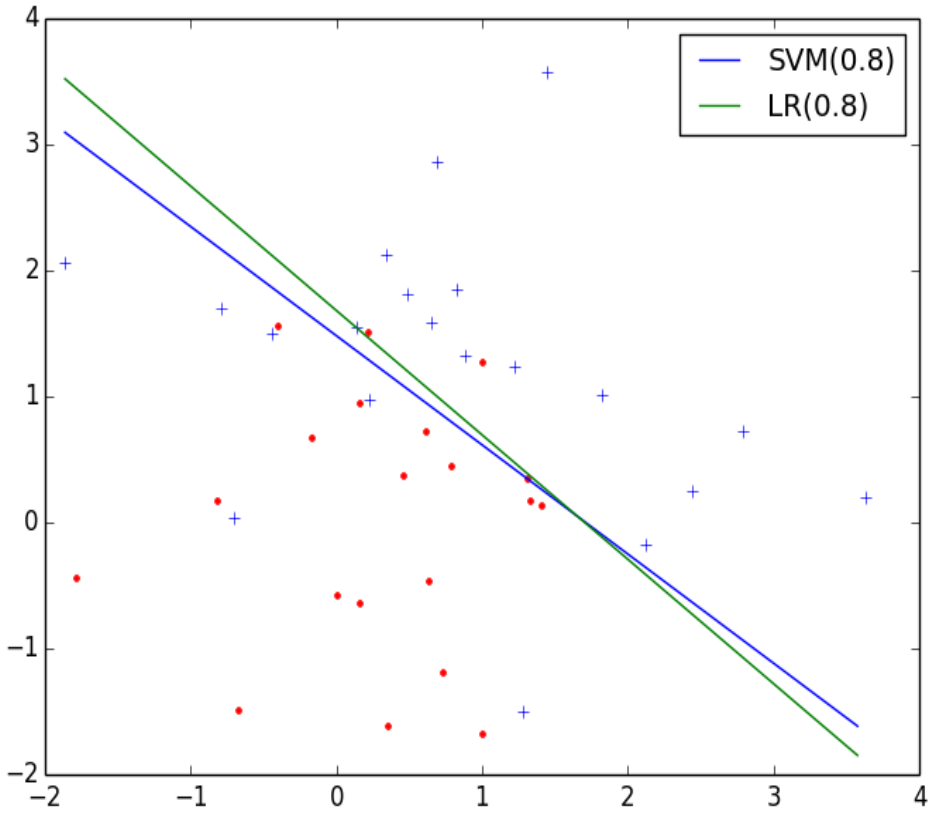
We control our error tolerance with the parameter C. As we decrease C, we shrink W (increase the margin) and also allow for more support vectors.



SVM VS. LOGREG

When dealing with linear models for classification, SVMs and Logistic Regression have very similar solutions. When to use which is a matter of needing to exploit properties beyond the decision boundary.

SVM vs. LR on Simulated Data



SVM

- Strong geometric interpretation
- Easily extendable to non-linear decision surfaces
- Strong learning guarantees

LR

- Designed for P(Y|X)
- Strong statistical properties
- Suited for ranking