Lecture 2: ML Basics + Spam Filtering

Siddharth Garg

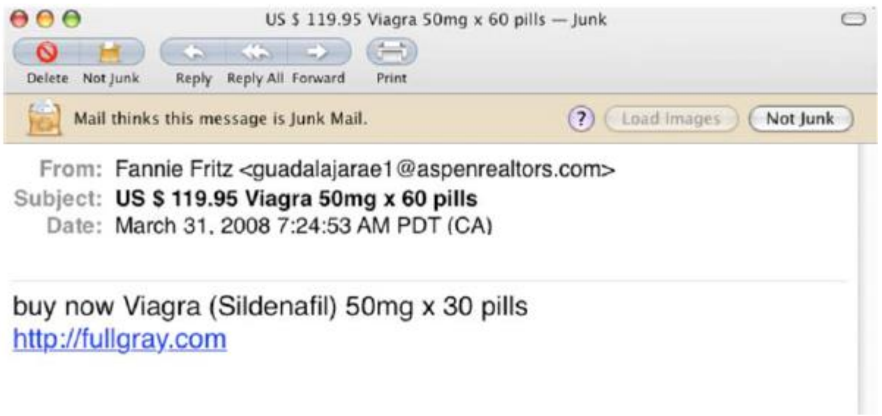
sg175@nyu.edu

Classification

Task (T):

- Emails $x \in$ all possible emails and y $\in \{spam, non\_spam\}$ find

$$f: x \rightarrow y$$



SPAM

Experience (E):

- A "training dataset" a emails marked as "spam" or "non_spam"

Performance (P):

- Spam detection accuracy

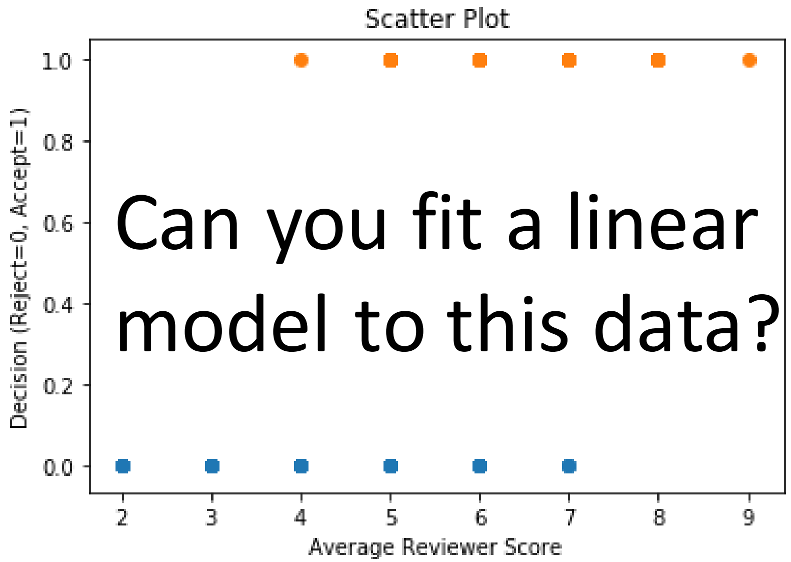
"*Supervised* Learning (Classification)"

Binary Classification

“Categorical variable”

Binary Classification Task (T):

- Simplest example where $x \in \Re$ and $y \in \{0,1\}$



Can you fit a linear model to this data?

- Dataset of ICLR’18 review scores vs. accept/reject decisions

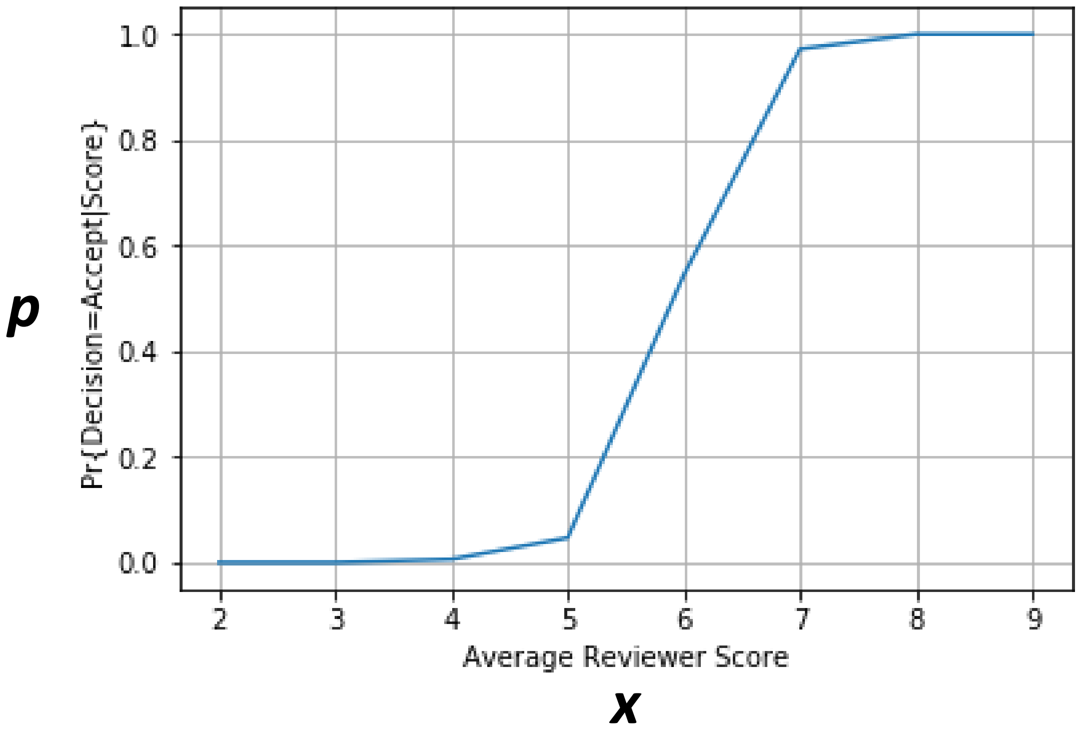
TL;DR	_bibtex	abstract	authorids	authors	conf_1	conf_2	conf_3	decision	review	review_1	review_2	review_3	title
None	@article{\nsharma2018hyperedge2vec:,\ntitle={H...	Data structured in form of overlapping or non-...	[sharm170@umn.edu, srjoty@ntu.edu.sg, himanshu...	[Ankit Sharma, Shafiq Joty, Himanshu Kharkwal,...	3.0	3.0	4.0	Reject	5.000000	5.0	5.0	5.0	Hyperedge2vec: Distributed Representations for...
Query-based black-box attacks on deep neural n...	@article{\nnitin2018exploring,\ntitle={Explori...	Existing black-box attacks on deep neural netw...	[abhagoji@princeton.edu, _w@eecs.berkeley.edu,...	[Arjun Nitin Bhagoji, Warren He, Bo Li, Dawn S...	4.0	3.0	4.0	Reject	6.000000	5.0	6.0	7.0	Exploring the Space of Black-box Attacks on De...
A theory and algorithmic framework for predict...	@article{\nd.2018learning,\ntitle={Learning We...	Predictive models that generalize well under d...	[fredrikj@mit.edu, kallus@cornell.edu, urish22...	[Fredrik D. Johansson, Nathan Kallus, Uri Shal...	3.0	3.0	4.0	Reject	6.666667	5.0	8.0	7.0	Learning Weighted Representations for Generali...
We prove that DNN is a recursively approximate...	@article{\nzheng2018understanding,\ntitle={Und...	Deep learning achieves remarkable generalizati...	[zhenggh@mail.ustc.edu.cn, jtsang@bjtu.edu.cn,...	[Guanhua Zheng, Jitao Sang, Changsheng Xu]	3.0	3.0	2.0	Reject	3.666667	2.0	3.0	6.0	Understanding Deep Learning Generalization by

Logistic Regression

Binary Classification Task (T):

Pr{Decision=Accept|Score}

- Instead, let's compute and plot $p = \Pr\{y = 1 \mid x\}$



- Idea: Linear regression to fit *p* as a function of *x*

$$p = \beta_1 x + \beta_0$$

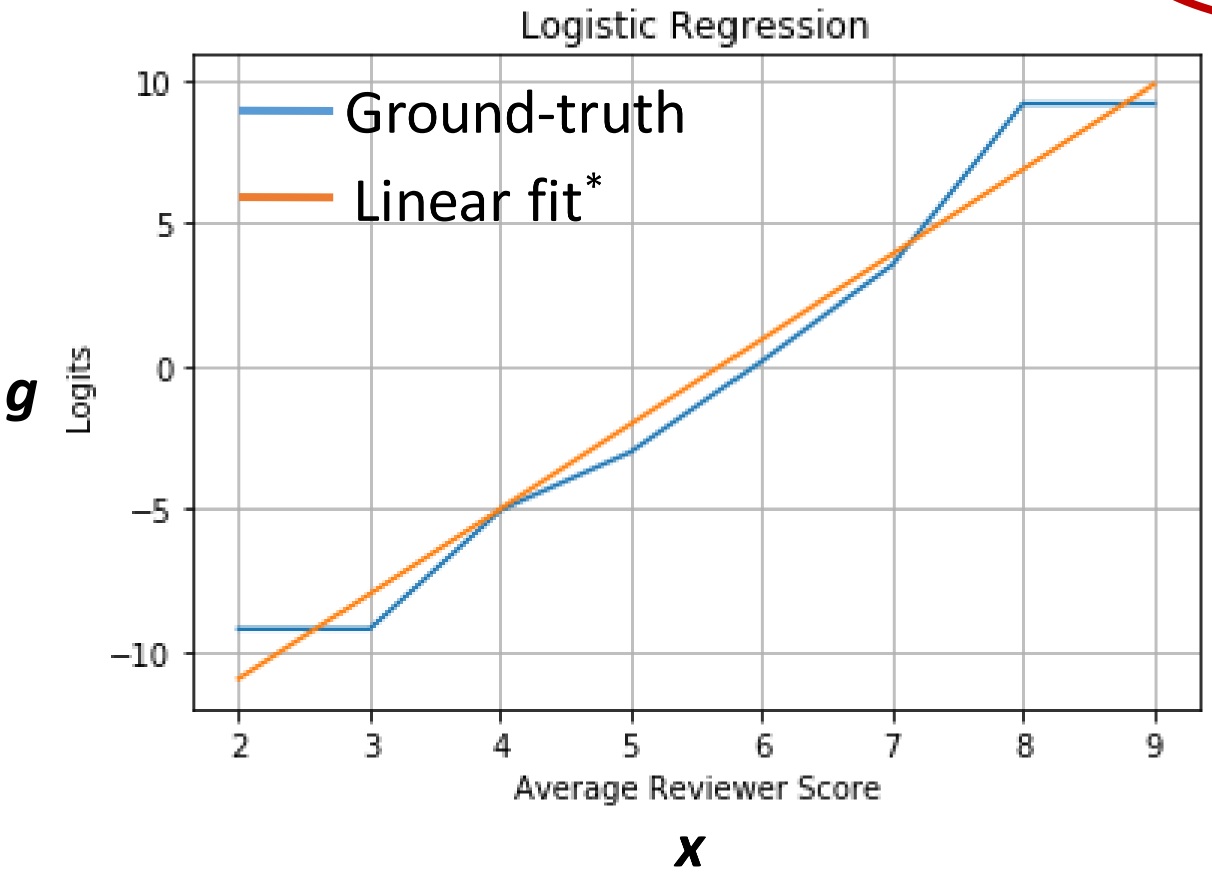
- Is this a good idea?
  - Probability *p* is always bounded between [0,1]

Logistic Regression

Binary Classification Task (T):

- Consider the following function: $g = \log(\frac{p}{1-p})$

"Logits" Function



- What is the range of g?

$$g \in [-\infty, \infty]$$

- Logistic Regression: fit logits function using a linear model!

$$g = \log(\frac{p}{1-p}) = \beta_1 x + \beta_0$$

Note: the linear fit is illustrative only. How to determine the best linear fit will be discussed next!

Logistic Regression

Pr{Decision=Accept|Score}

$$g = \log\left(\frac{p}{1-p}\right) = \beta_1 x + \beta_0 \Rightarrow p = \frac{1}{1+e^{-(\beta_1 x + \beta_0)}}$$

- What is Pr{Decision=Reject|Score}

$$1-p = \frac{e^{-(\beta_1 x + \beta_0)}}{1+e^{-(\beta_1 x + \beta_0)}}$$

How do we find the model parameters β_1 and β_0?

Model Estimation

- We will use an approach referred to as Maximum Likelihood Estimation (MLE)
  - Let's assume that the model(i.e., β_1 and β_0) is magically known. Consider the training dataset below. What is the likelihood that the dataset came from our model?

#	X	Y
1	$x_1 = 3$	$y_1 = 0$
2	$x_2 = 8$	$y_2 = 1$
..		
..		
N	$x_N = 6$	$y_N = 1$

$$Likelihood = \frac{e^{-(\beta_1 x_1 + \beta_0)}}{1 + e^{-(\beta_1 x_1 + \beta_0)}} * \frac{1}{1 + e^{-(\beta_1 x_2 + \beta_0)}} * \ldots \frac{1}{1 + e^{-(\beta_1 x_N + \beta_0)}}$$

Model Estimation

- We will use an approach referred to as Maximum Likelihood Estimation (MLE)
  - Let's assume that the model(i.e., β_1 and β_0) is magically known. Consider the training dataset below. What is the likelihood that the dataset came from our model?

#	X	Y
1	$x_1 = 3$	$y_1 = 0$
2	$x_2 = 8$	$y_2 = 1$
..		
..		
N	$x_N = 6$	$y_N = 1$

$$Likelihood = \frac{e^{-(3\beta_1+\beta_0)}}{1+e^{-(3\beta_1+\beta_0)}} * \frac{1}{1+e^{-(8\beta_1+\beta_0)}} * \ldots \frac{1}{1+e^{-(6\beta_1+\beta_0)}}$$

Model Estimation

- We will use an approach referred to as Maximum Likelihood Estimation (MLE)
  - Let's assume that the model(i.e., β_1 and β_0) is magically known. Consider the training dataset below. What is the likelihood that the dataset came from our model?

#	X	Y
1	$x_1 = 3$	$y_1 = 0$
2	$x_2 = 8$	$y_2 = 1$
..		
..		
N	$x_N = 6$	$y_N = 1$

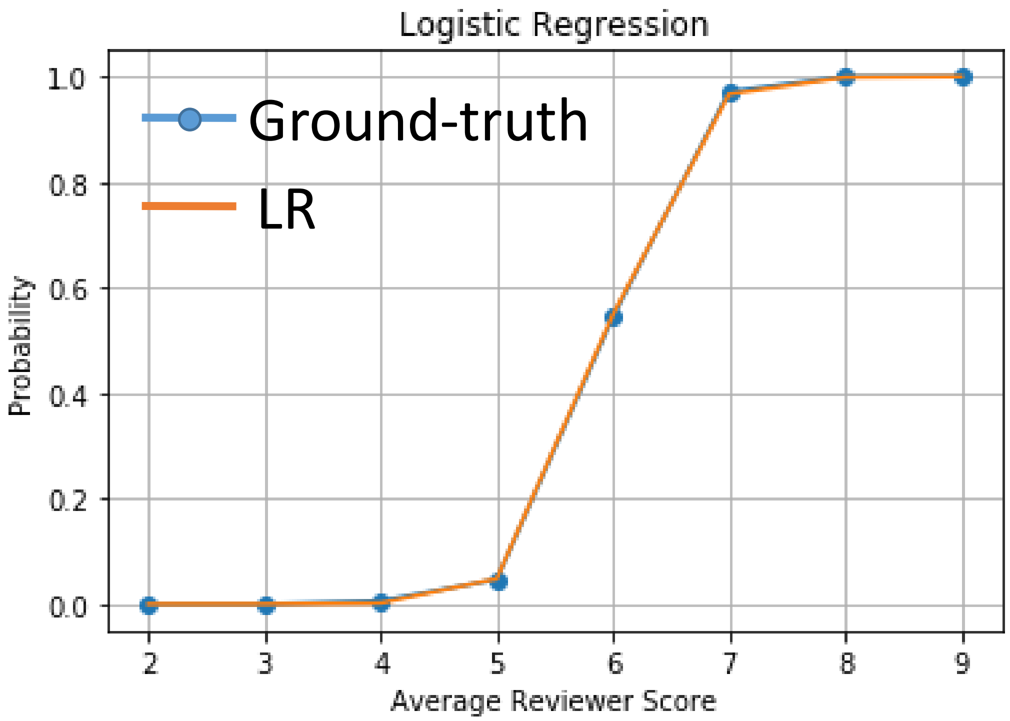
$$Log-Likelihood = \log(\frac{e^{-(3\beta_1+\beta_0)}}{1+e^{-(3\beta_1+\beta_0)}}) + \log(\frac{1}{1+e^{-(8\beta_1+\beta_0)}}) + \ldots \log(\frac{1}{1+e^{-(6\beta_1+\beta_0)}})$$

$g(\beta_1, \beta_0)$ Function of model parameters only

$\boxed{\text{Find } \beta_1 \text{ and } \beta_0 \text{ that maximize g (or minimize the "loss" } -\text{g)}}$

$$Loss(\beta_1, \beta_0) = -g(\beta_1, \beta_0)$$

We Won’t Worry About How




```python
from sklearn import linear_model

#Instantiate an LR object
logreg = sklearn.linear_model.LogisticRegression(C=1e5);

#Recall: your training data must have a column of ones for the constant term
xd = np.ones((numPapers,2));
xd[:,0] = np.append(rscores,ascores)

yd = np.append(rlabels,alabels);

logreg.fit(xd,yd);

#Plot Pr{Accept|Score}
rv = np.ones((len(revRange),2));
rv[:,0] = revRange;
prpredict=logreg.predict_proba(rv)
```

From regression to classification: if probability of Accept > 0.5, then output Accept.

Logistic Regression: Multi-Variate Case

UCI Spam Dataset:

https://archive.ics.uci.edu/ml/datasets/Spambase

Attribute Information:

The last column of 'spambase.data' denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. Most of the attributes indicate whether a particular word or character was frequently occuring in the e-mail. The run-length attributes (55-57) measure the length of sequences of consecutive capital letters. For the statistical measures of each attribute, see the end of this file. Here are the definitions of the attributes:

48 continuous real [0,100] attributes of type word_freq_WORD
= percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by
non-alphanumeric characters or end-of-string.

6 continuous real [0,100] attributes of type char_freq_CHAR]
= percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,...] attribute of type capital_run_length_average
= average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_longest
= length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_total
= sum of length of uninterrupted sequences of capital letters
= total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam
= denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

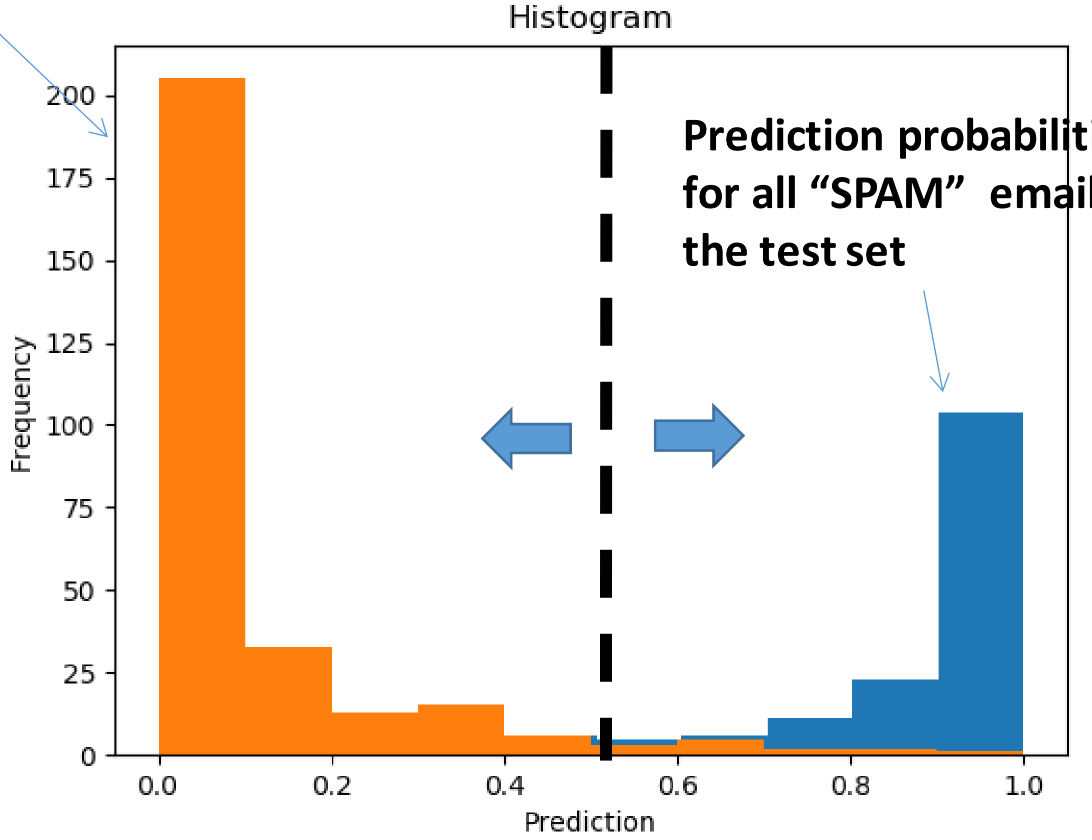
- **57 Real or integer valued features**
- **Binary output class**

$$p_{spam} = \frac{1}{1 + e^{-(\sum_{i=1}^{M} \beta_i x_i + \beta_0)}}$$

LR on Spam Database: Results

90% of samples used for training, remaining 10% used for test

Prediction probabilities for all "SPAM" emails in the test set



Prediction probabilities for all "SPAM" emails in the test set

```
#Instantiate an LR object
logreg = sklearn.linear_model.LogisticRegression(C=1e5);

#Recall: your training data must have a column of ones for the constant term
xd = np.ones((numPapers,2));
xd[:,0] = np.append(rscores,ascores)

yd = np.append(rlabels,alabels);

logreg.fit(xd,yd);

#Plot Pr{Accept|Score}
rv = np.ones((len(revRange),2));
rv[:,0] = revRange;
prpredict=logreg.predict_proba(rv)
```

Which emails are mis-predicted?

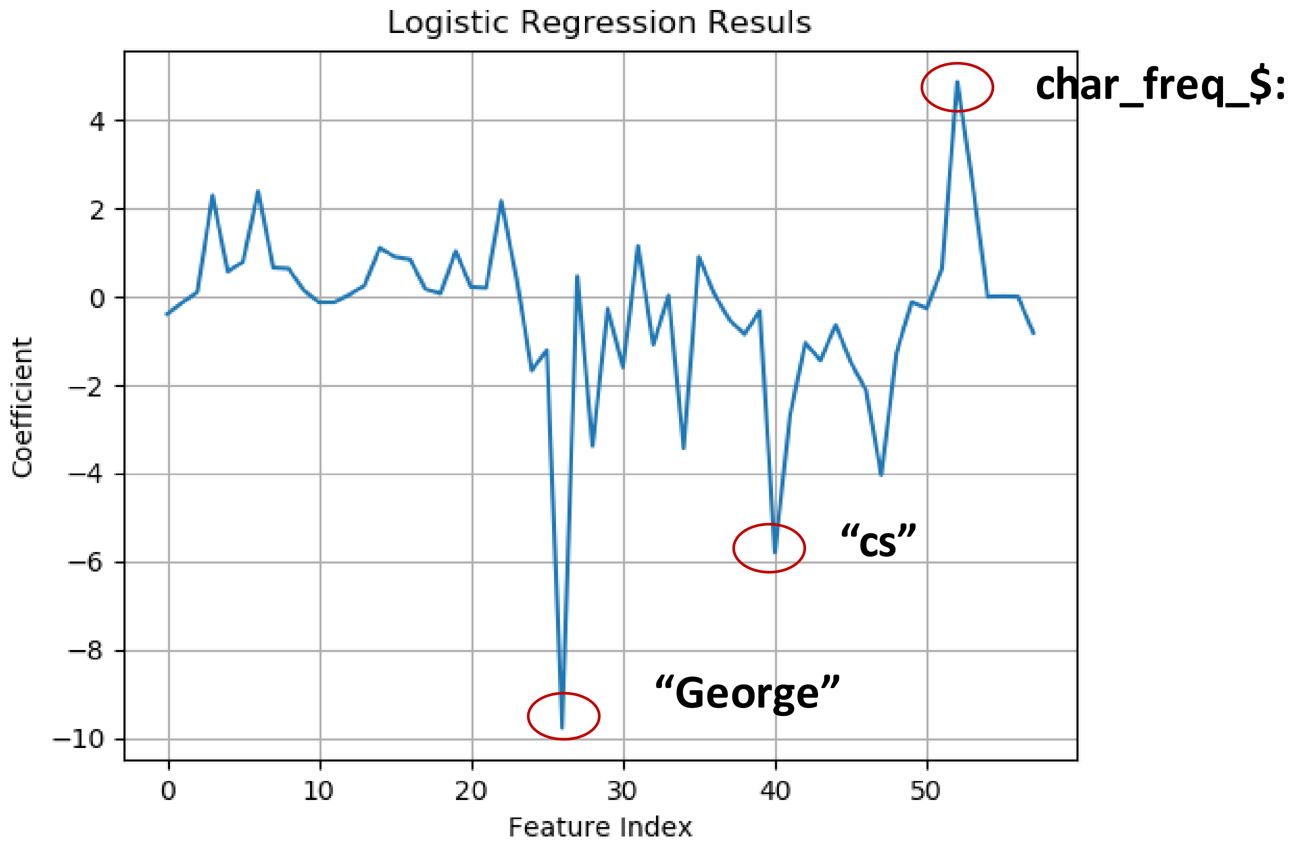
Accuracy on test set: ~92%

Which Features Matter?

Our Model:

$$p_{spam} = \frac{1}{1+e^{-(\sum_{i=1}^{M}\beta_i x_i + \beta_0)}}$$

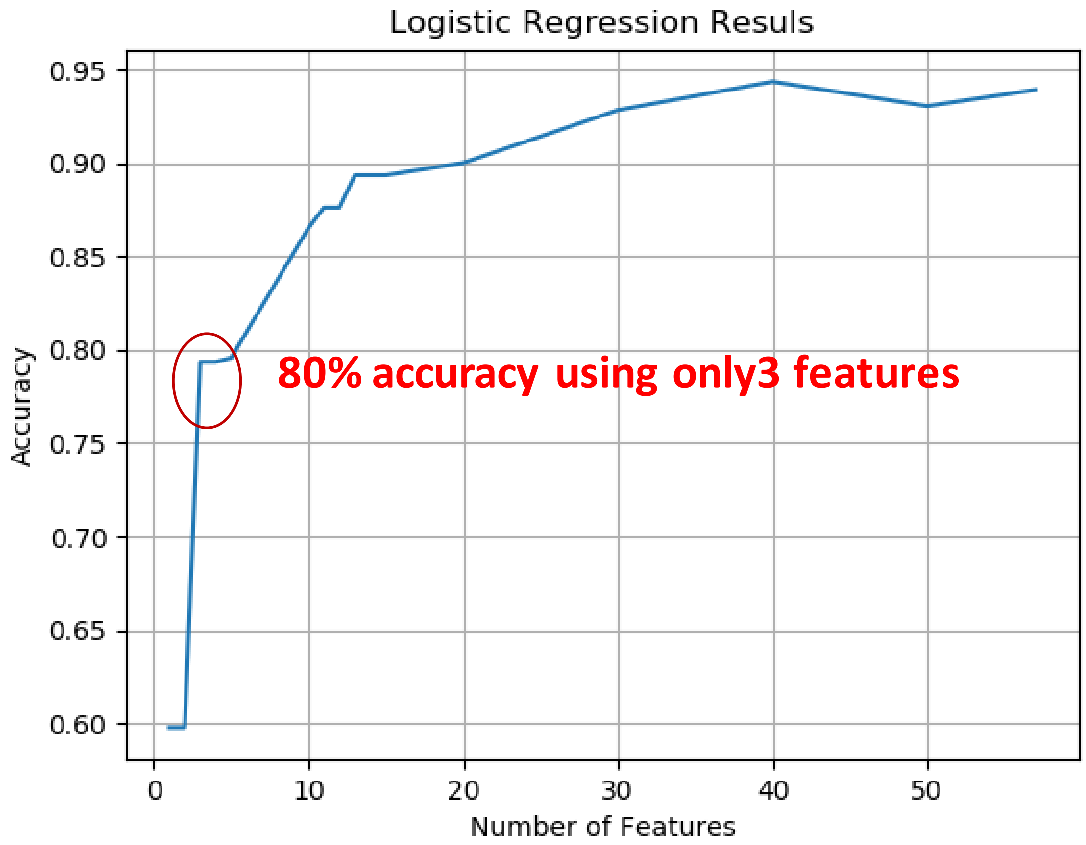
What does β_i=0 imply about feature *i*?



Reasonable hypothesis: features with larger absolute values of β matter more.

Feature Selection

Retrain and predict using only the top-k features



Can we explicitly train the parameters so as to prioritize a "sparser" model?

Why?

Low model complexity prevents overfitting!

Recall that during training we were seeking to minimize:

$$\hat{\beta} = \min_{\beta} Loss(\beta)$$

How should this objective function change?

Regularization

L_p Norm of a vector *x* $\|x\|_p = (\sum |x_i|^p)^{1/p}$

p	L_p Norm	Interpretation
0	$\|x\|_0 = (\sum \| x_i \|^0)^{1/0}$	Number of Non-zero Entries
1	$\|x\|_1 = (\sum \| x_i \|)$	Sum of absolute values
2	$\|x\|_2 = (\sum \| x_i \|^2)^{0.5}$	Root mean square
∞	$\|x\|_\infty = (\sum \| x_i \|^\infty)^0$	Max. value

"Regularized" loss

$$\hat{\beta} = \min_{\beta}\{Loss(\beta) + c\|\beta\|_0\}$$

c controls the relative importance of the regularization penalty

Regularization In Practice

L0 Regularization

$$\hat{\beta} = \min_{\beta}\{Loss(\beta) + c\|\beta\|_0\}$$

Hard "combinatorial" optimization problem!

Instead, the following regularization functions are commonly used:

L1 Regularization (LASSO)

$$\hat{\beta} = \min_{\beta}\{Loss(\beta) + c\|\beta\|_1\}$$

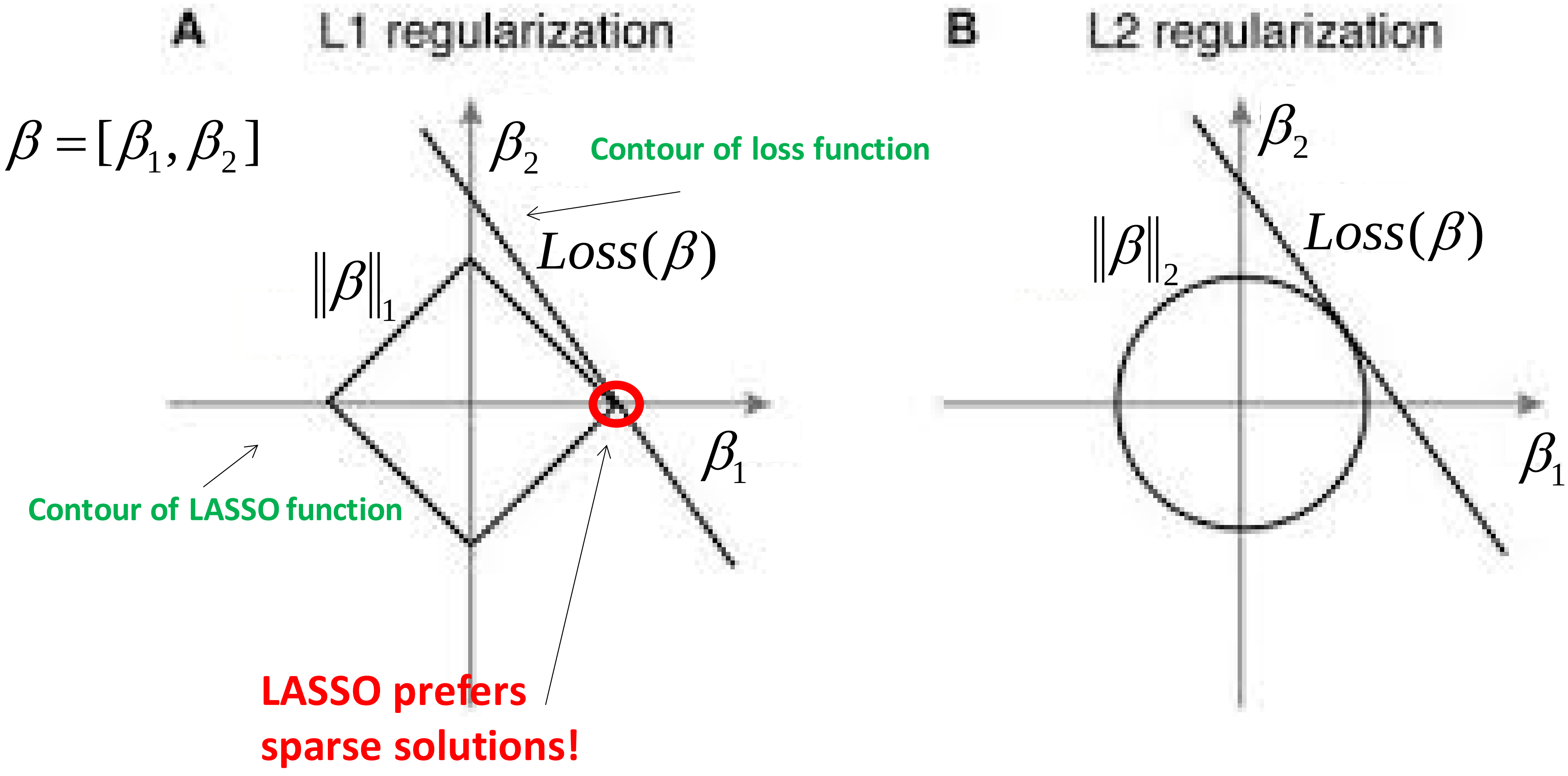
L2 Regularization (Ridge)

$$\hat{\beta} = \min_{\beta}\{Loss(\beta) + c\|\beta\|_2\}$$

We are penalizing "large" coefficients.

But why?

LASSO and Ridge Regularization



Regularization for Spam Classification

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross- entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag' and 'lbfgs' solvers. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty.

Read more in the User Guide.

Parameters: **penalty** : str, 'l1' or 'l2', default: 'l2'

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

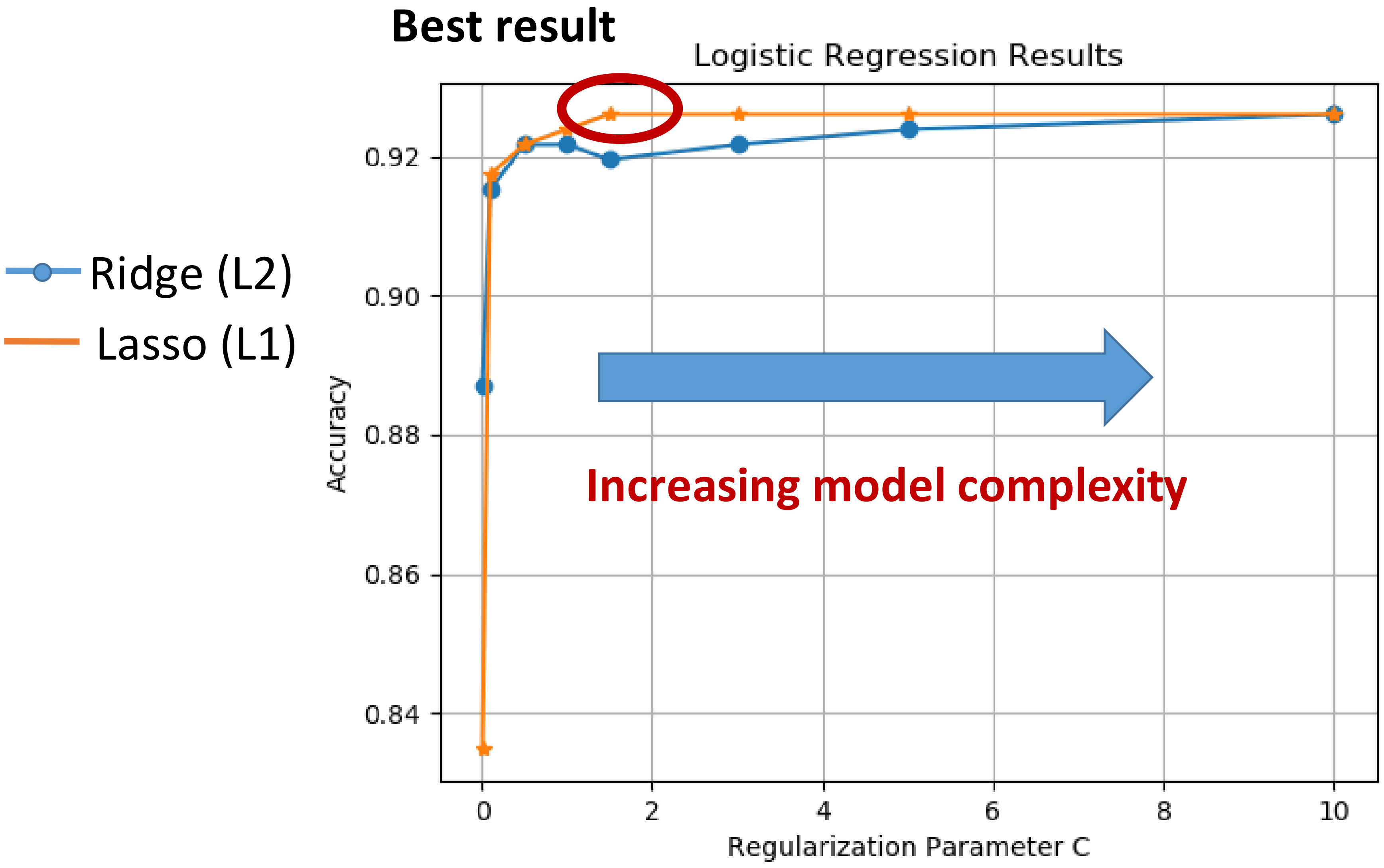
C : float, default: 1.0

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

Which regularization function to use?

How should we select *c*?

Impact of C


Best result
Logistic Regression Results
Ridge (L2)
Lasso (L1)
Accuracy
0.92
0.90
0.88
0.86
0.84
0
2
4
6
8
10
Regularization Parameter C
Increasing model complexity

Errors in Binary Classification

- Two types of errors:
  - Type I error (False positive / false alarm): Decide $\hat{y} = 1$ when $y = 0$
  - Type II error (False negative / missed detection): Decide $\hat{y} = 0$ when $y = 1$
- Implication of these errors may be different
  - Think of breast cancer diagnosis
- Accuracy of classifier can be measured by:
  - $TPR = P(\hat{y} = 1 | y = 1)$
  - $FPR = P(\hat{y} = 1 | y = 0)$

predicted→ real↓	*Class_pos*	*Class_neg*
Class_pos	TP	FN
Class_neg	FP	TN

$$\text{TPR (sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

[Remaining Slides from Prof. Rangan's Intro to ML Class]

Hard Decisions

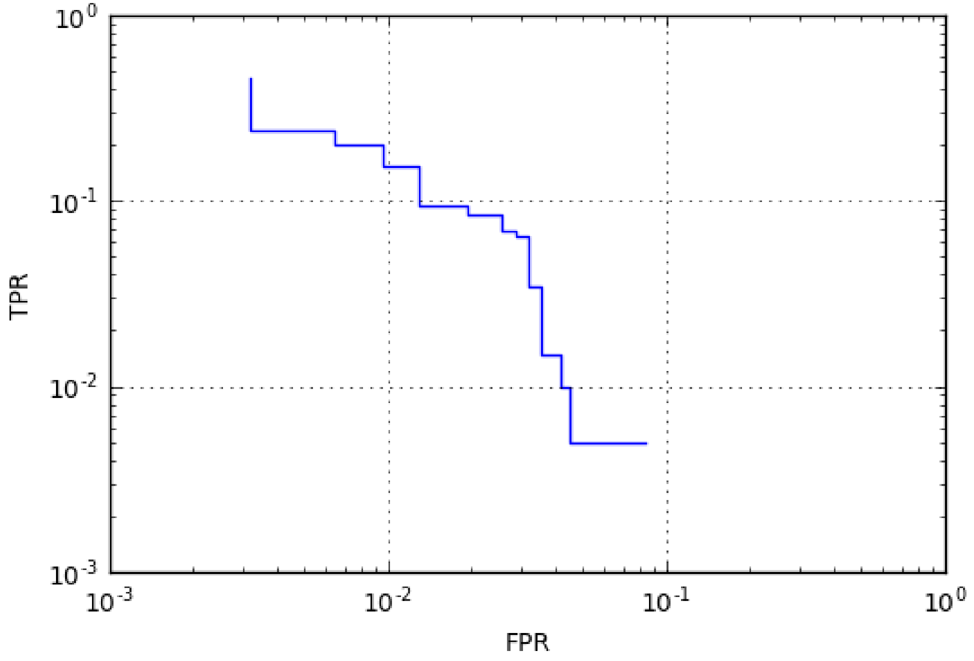
- Logistic classifier outputs a soft label: $P(y=1|x) \in [0,1]$
  - $P(y=1|x) \approx 1 \Rightarrow y=1$ more likely
  - $P(y=0|x) \approx 1 \Rightarrow y=0$ more likely
- Can obtain a hard label by thresholding:
  - Set $\hat{y}=1 \Leftrightarrow \quad P(y=1|x) > t$
  - t = Threshold
- How to set threshold?
  - Set $t = \frac{1}{2} \Rightarrow$ Minimizes overall error rate
  - Increasing $t \Rightarrow$ Decreases false positives
  - Decreasing $t \Rightarrow$ Decreases missed detections

ROC Curve

```
from sklearn import metrics
yprob = logreg.predict_log_proba(Xtr)
fpr, tpr, thresholds = metrics.roc_curve(ytr,yprob[:,1])

plt.loglog(fpr,1-tpr)
plt.grid()
plt.xlabel('FPR')
plt.ylabel('TPR')
```

- Varying threshold obtains a set of classifier
- Trades off FPR and TPR
- Can visualize with ROC curve
  - Receiver operating curve
  - Term from digital communications



Linear Classifiers

- LR based spam detector that we discussed is an example of a **Linear Classifier**

Recall that:

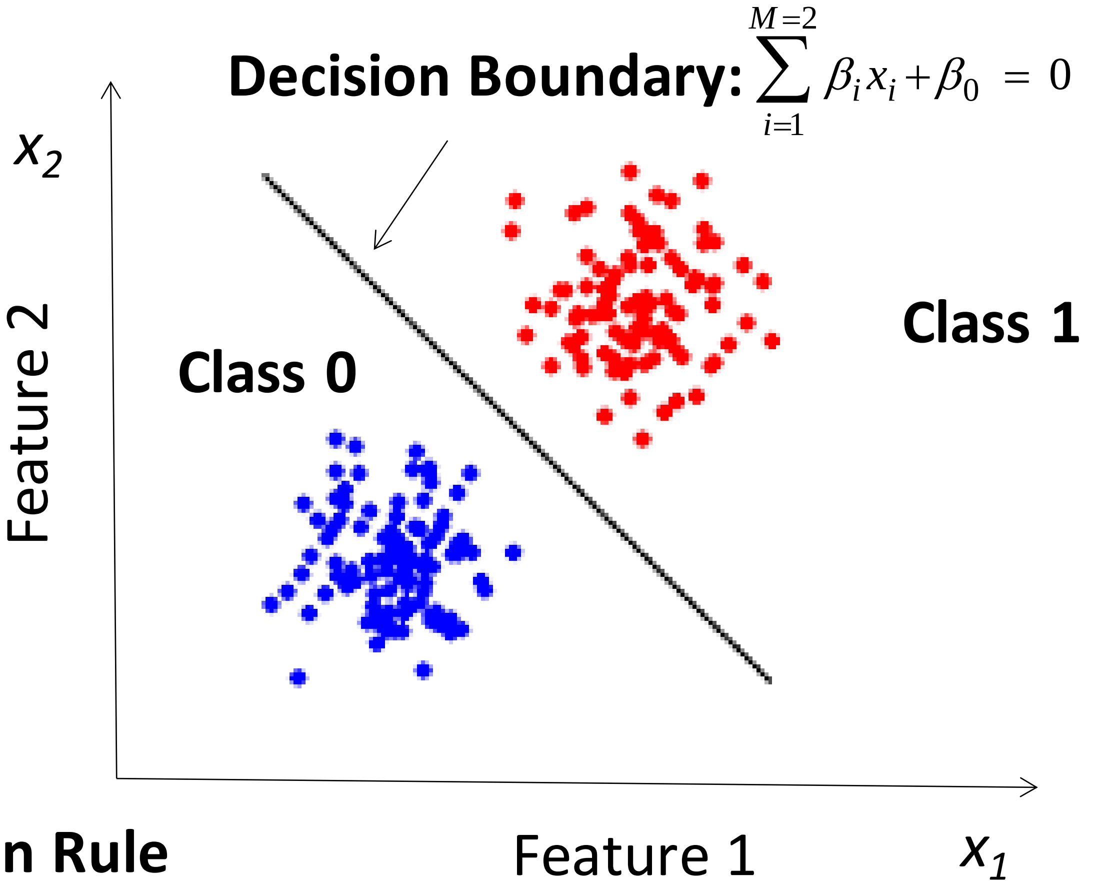
$$p_{Class1} = \frac{1}{1+e^{-(\sum_{i=1}^{M}\beta_i x_i + \beta_0)}}$$

$$x \in \text{Class 1 iff } p_{Class1} \geq 0.5$$

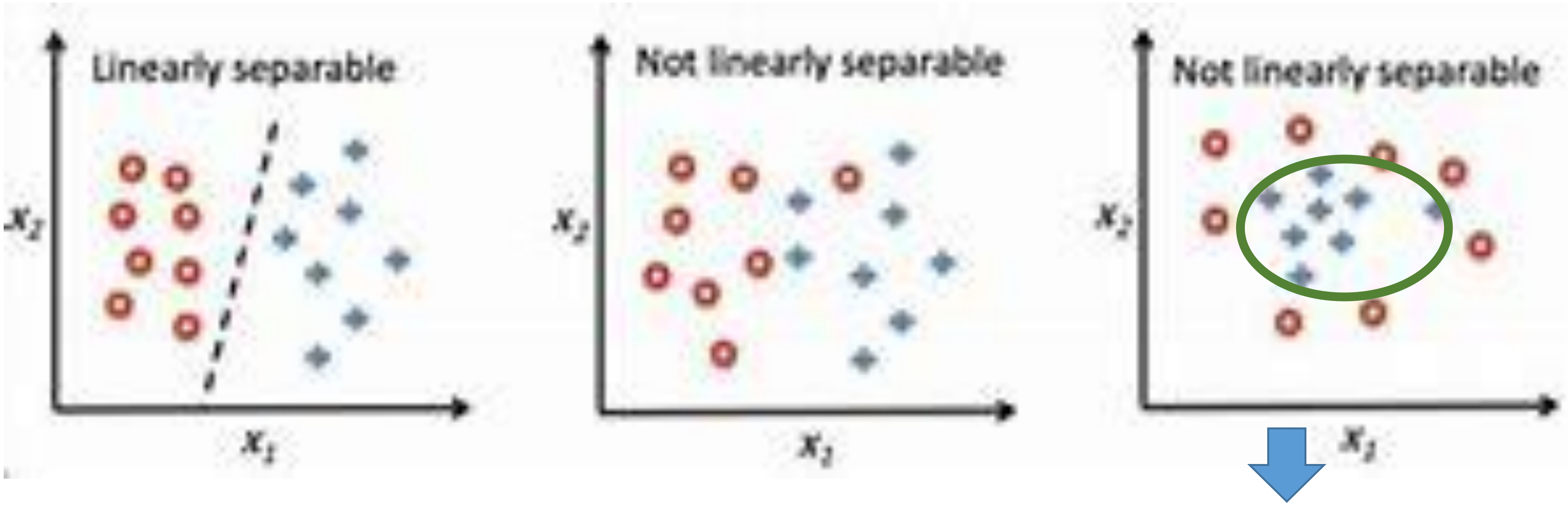
$$\Rightarrow e^{-(\sum_{i=1}^{M}\beta_i x_i + \beta_0)} \leq 1$$

$$\boxed{\Rightarrow (\sum_{i=1}^{M}\beta_i x_i + \beta_0) \geq 0}$$

Decision Rule



Linear Separability



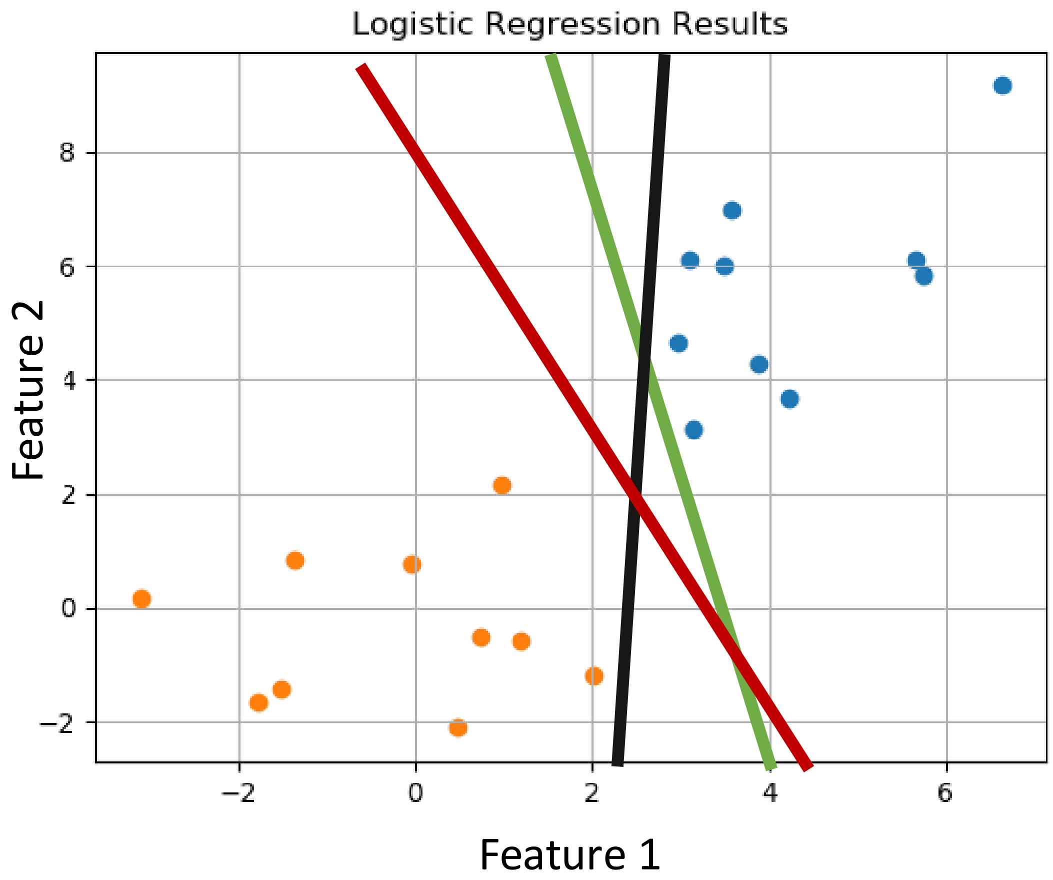
Is there a transformation of the features for which this data is linearly separable?

http://www.jeannicholashould.com/public/linearly_separable.jpg

Support Vector Machines (SVMs)

- Recall that LR attempts to minimize the log-likelihood of the training data under the LR model
  - LR has a probabilistic interpretation
- SVMs on the other hand have a geometric interpretation
  - Goal: maximize margin to closest training data sample

All 3 classifiers have 100% training accuracy



But which of the 3 classifiers would you pick?

SVMs: Maximize Margin

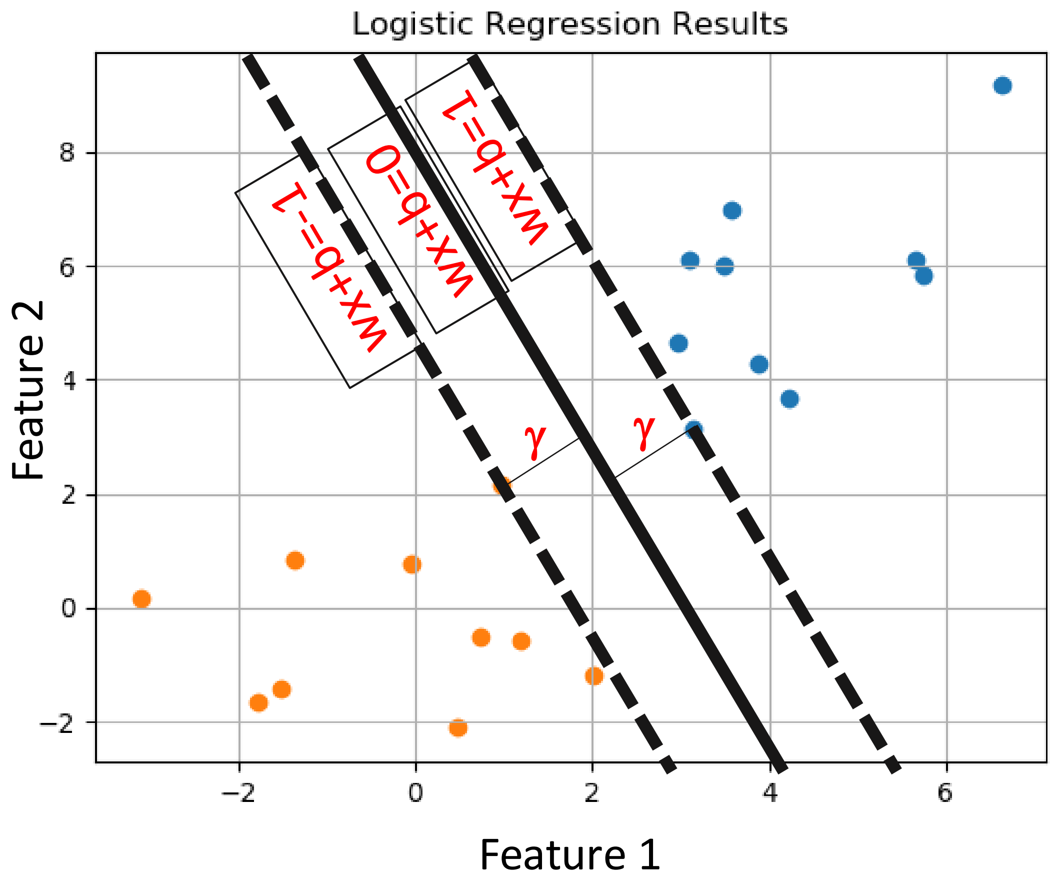
- Assume data is linearly seperable
- Goal: maximize margin to closest training data sample
  - Let the closest points on either side lie on :

$$wx + b = \pm 1$$

  - What is the value of the margin γ?

$$\lambda = \frac{1}{\|w\|_2}$$

Maximize



SVMs: Optimization

- Optimization problem

$$\max \frac{1}{\|w\|_2} \Rightarrow \min \|w\|_2$$

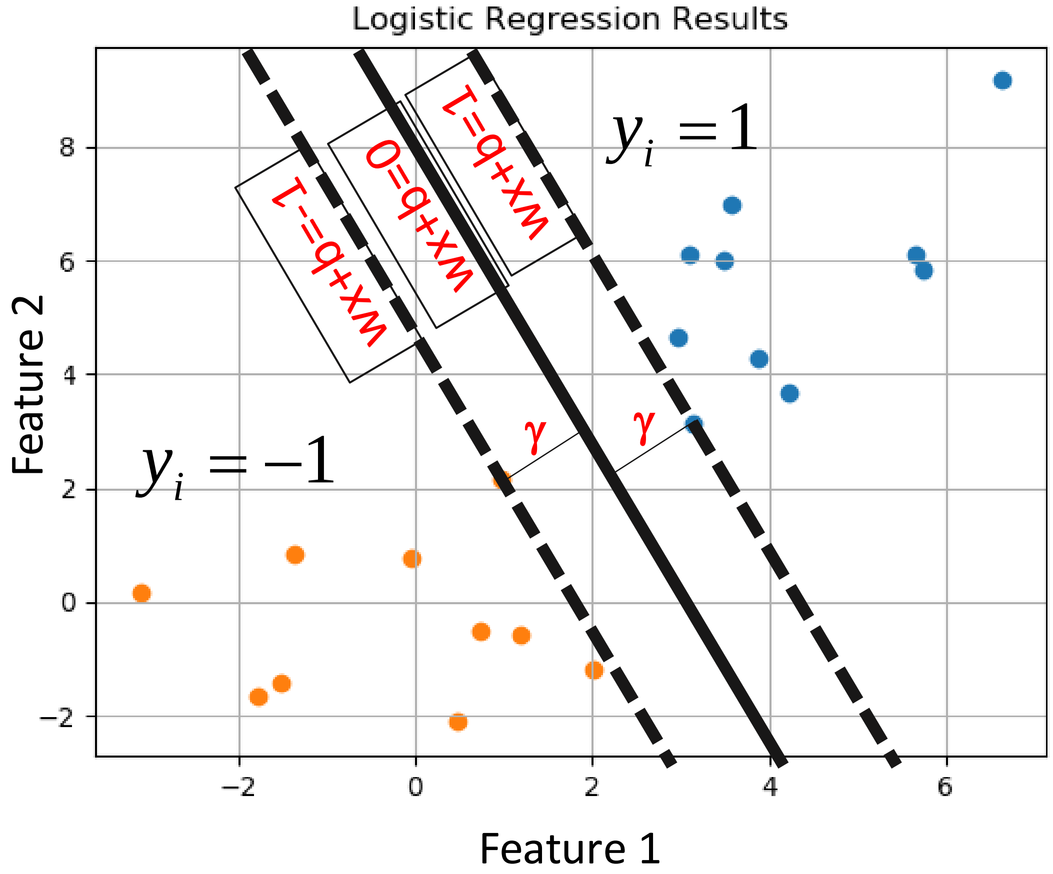
$$wx_i + b \geq 1 \qquad y_i = 1$$

$$wx_i + b \leq -1 \qquad y_i = -1$$

- Or equivalently

$$\min \|w\|_2$$

$$y_i(wx_i + b) \geq 1$$



SVMs: Non-linearly Separable

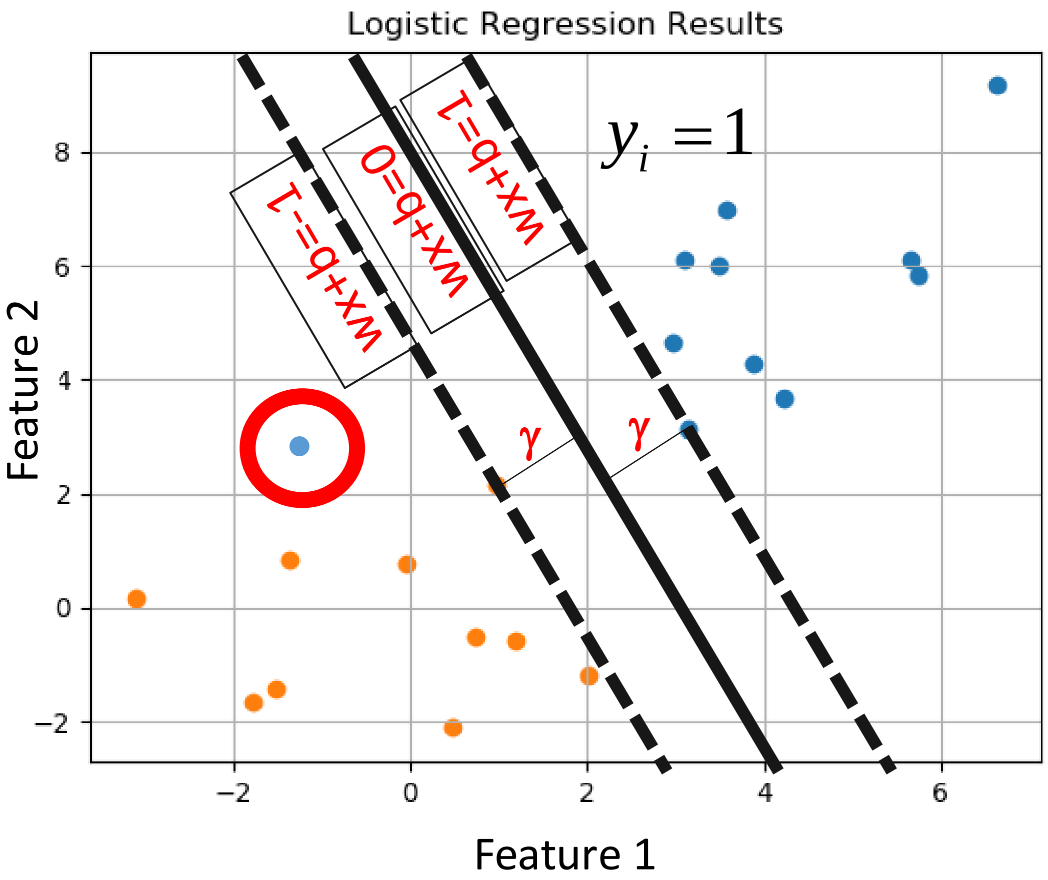
- Make the constraints "soft" by introducing an "margin" ε_i

Min $\sum_i \varepsilon_i + \alpha \|w\|_2$

$$y_i(wx_i + b) \geq 1 - \varepsilon_i$$

$$\varepsilon_i \geq 0$$

Points can be misclassified with "cost" ε_i



- Note that in any solution to this problem

$$\varepsilon_i = \max\{0, 1 - y_i(wx_i + b)\}$$

SVMs: Putting it All Together

- SVMs minimize the following objective

Regularization constant α

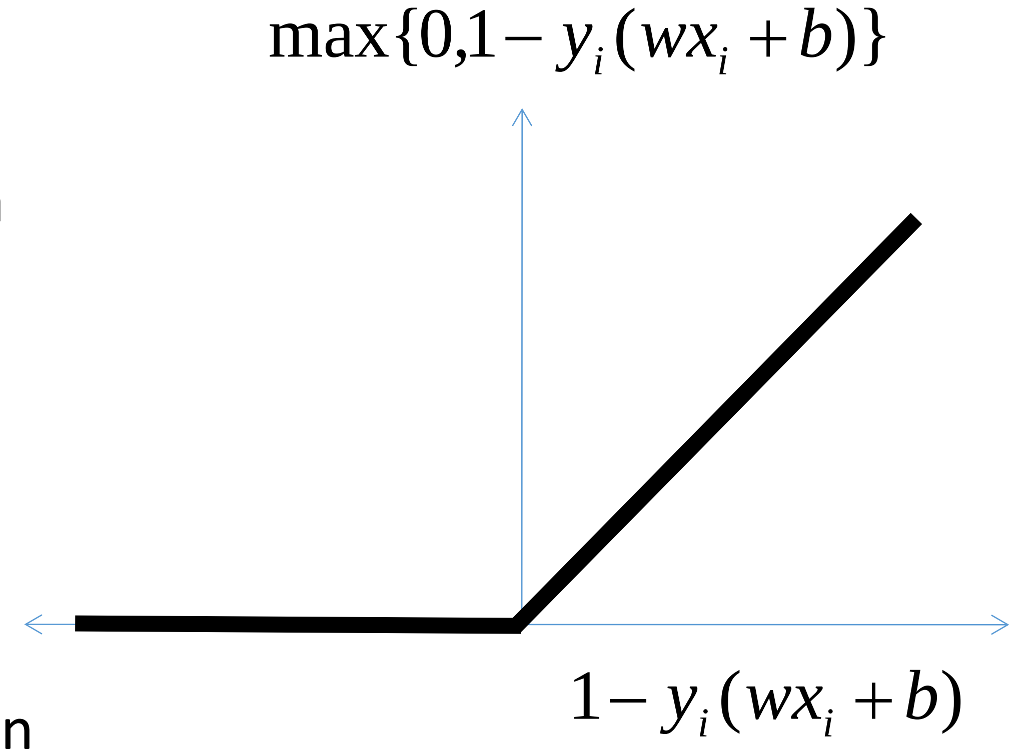
Increasing α results in greater margins at the expense of misclassifications on training data

$$\max\{0, 1 - y_i(wx_i + b)\} + \alpha \|w\|_2$$

"Hinge Loss"

Seeks to minimize mis-classfications

Tries to maximize margin (L_2 regularizer!)



Relation to Logistic Regression

- LR and SVM differ only on the loss function they use
  - “Log-loss” versus “hinge-loss”

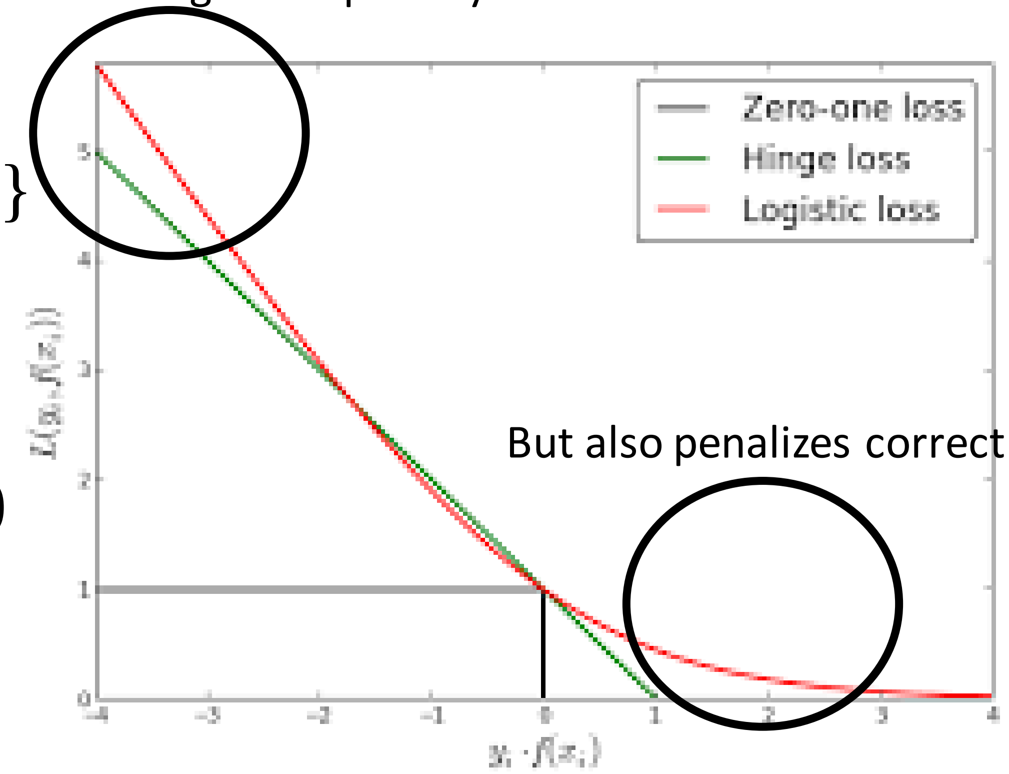
Hinge Loss:

$$\max\{0, 1 - y_i(wx_i + b)\} = \max\{0, 1 - q\}$$

Log Loss:

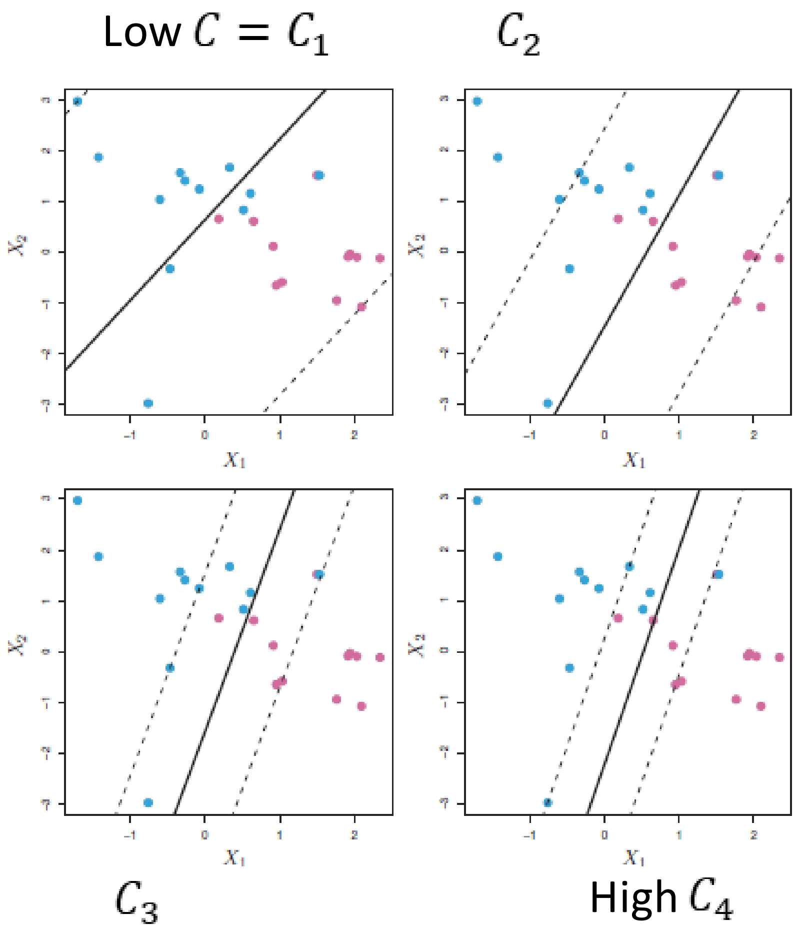
$$-\log_2\left(\frac{1}{1 + e^{-y_i(wx_i + b)}}\right) = \log_2(1 + e^{-q})$$

LR has greater penalty for mis-classification



But also penalizes correct classification!

Illustrating Effect of C

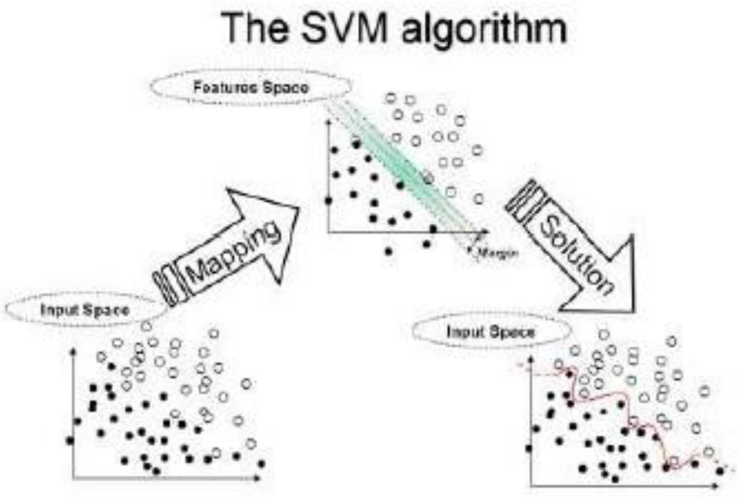


- Fig. 9.7 of ISL
  - Note: C has opposite meaning in ISL than python
  - Here, we use python meaning
- Low C:
  - Many violations of margin.
  - Many more SVs
  - Reduces variance by using more samples

Transform Problem

- Transform problem: replace $\boldsymbol{x}$ with $\phi(\boldsymbol{x})$
  - Enables more rich classifiers
  - Examples: polynomial classification $\phi(x) = [1, x, x^2, \ldots, x^{d-1}]$
- Tries to find separation in a feature space

From https://www.dtreg.com/solution/view/20



Kernel Trick

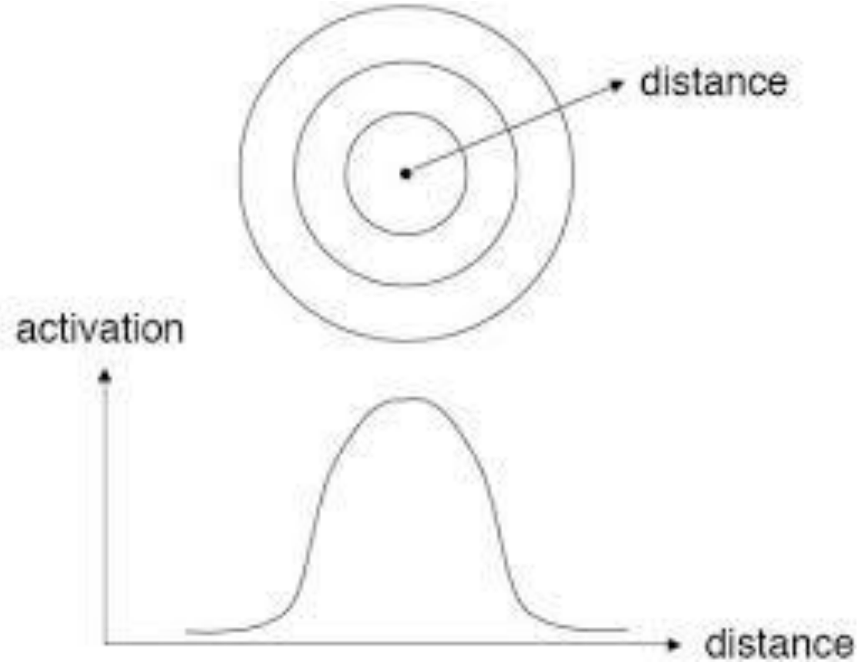
- Classifier is:
  - $z = b + \boldsymbol{w}^T \boldsymbol{x} = b + \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})$, $K(\boldsymbol{x}_i, \boldsymbol{x}) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x})$
  - $\hat{y} = \text{sign}(z) = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \end{cases}$

- Do not need to explicitly compute $\phi(\boldsymbol{x})$
- Can directly compute kernel $K(\boldsymbol{x}_i, \boldsymbol{x})$
  - Provided kernel corresponds to some $\phi(\boldsymbol{x})$

Understanding the Kernel

- Kernel function $K(\boldsymbol{x}_i, \boldsymbol{x})$:
  - measures "distance" between new sample $\boldsymbol{x}$ and training data $\boldsymbol{x}_i$
  - $K(\boldsymbol{x}_i, \boldsymbol{x})$ large $\Rightarrow$ $\boldsymbol{x}_i, \boldsymbol{x}$ close
  - $K(\boldsymbol{x}_i, \boldsymbol{x}) \approx 0 \Rightarrow \boldsymbol{x}_i, \boldsymbol{x}$ far

- Linear discriminant $z = b + \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})$
  - Weighs sample $\boldsymbol{x}_i$ that are close to $\boldsymbol{x}$

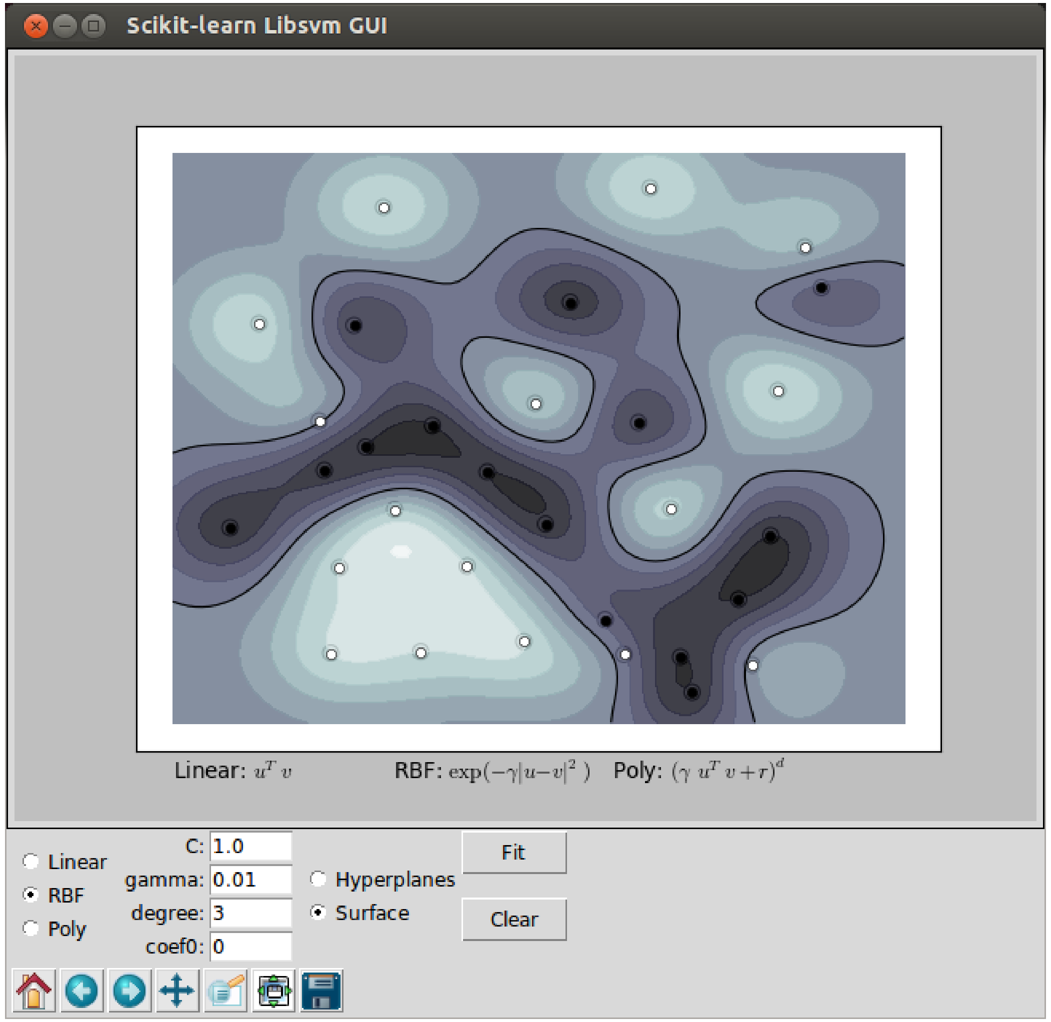


Possible Kernels

- Radial basis function:

$$K(x, x') = \exp[-\gamma \|x - x'\|^2]$$

  - $1/\gamma$ indicates width of kernel
- Polynomial kernel: $K(x, x') = |x^T x|^d$
  - Typically d=2



http://www.bogotobogo.com/python/scikit-learn/images/linearly_separable/svm_gui3.png

Spam Detection: Features

- Recall features used in the UCI Spam database

  48 continuous real [0,100] attributes of type word_freq_WORD

- Even easier way to encode features:
  - $x_i = 1$ if term i appears in a document; 0 otherwise
  - Boolean features

- Assume M Boolean features, x = $(x_1, x_2, \ldots, x_M)$
  - We want to map this M-dimensional Boolean input to a Boolean output y
  - *Thoughts?*
  - Instead of using LR or SVM we will start with an even simpler approach referred to as "Naiive Bayes"

Ref: Metsis, Vangelis, Ion Androutsopoulos, and Georgios Paliouras. "Spam filtering with naive bayes-which naive bayes?." In *CEAS*, vol. 17, pp. 28-69. 2006.

Naiive Bayes for Spam Filtering

- Assume M=1 Boolean feature, x = ($x_1, x_2, \ldots, x_M$)
- Each email is either {s=spam,l=legit}
- We begin by computing:

$$P\{spam \mid x\} = \frac{P\{x \mid spam\} * P\{spam\}}{P\{x\}}$$

Bayes Rule

$$\boxed{P\{A \cap B\} = P\{A \mid B\} * P\{B\}}$$

Ref: Metsis, Vangelis, Ion Androutsopoulos, and Georgios Paliouras. "Spam filtering with naive bayes-which naive bayes?." In *CEAS*, vol. 17, pp. 28-69. 2006.

Naiive Bayes for Spam Filtering

- We begin by computing:

$$P\{spam \mid x\} = \frac{P\{x \mid spam\} * P\{spam\}}{P\{x\}}$$

$$P\{x_1, x_2, .., x_M \mid spam\} = P\{x_1 \mid spam\} * P\{x_2 \mid spam\} * .. * P\{x_M \mid spam\}$$

Assuming that term occurrences are independent (given class)!

Is this a reasonable assumption?

Naiive Bayes for Spam Filtering

$$P\{x_1 \mid spam\} * P\{x_2 \mid spam\} * .. * P\{x_M \mid spam\}$$

How do we estimate this from the training dataset?

$P\{x_1 = 1 \mid spam\} =$ (#Spam emails that contain term 1)/(#spam emails)

What happens if term 1 never occurred in any spam email in the training dataset?

Laplacian Smoothing

$P\{x_1 = 1 | spam\} =$ ~~(#Spam emails that contain term 1)/(#spam emails)~~

= (#Spam emails that contain term 1+1)/(#spam emails+2)

Equivalent to assuming two addition spam emails in the training dataset, of which on contains all terms and the other is empty

$$P\{x_1 = 0 | spam\} = 1 - P\{x_1 = 1 | spam\}$$

= (#Spam emails that don't contain term 1+1)/(#spam emails+2)

Naiive Bayes for Spam Filtering

$$P\{x\} = P\{spam\} * P\{x \mid spam\} + P\{legit\} * P\{x \mid legit\}$$

$$P\{spam \mid x\} = \frac{P\{x \mid spam\} * P\{spam\}}{P\{x\}}$$ Vs. $$P\{legit \mid x\} = \frac{P\{x \mid legit\} * P\{legit\}}{P\{x\}}$$

Or:

$$P\{spam \mid x\} \geq threshold$$

References

- Cormack, Gordon V. "Email Spam Filtering: A Systematic Review." *Foundations and Trends® in Information Retrieval* 1.4 (2008): 335-455. https://www.ccs.neu.edu/home/vip/teach/IRcourse/IR_surveys/spam-filtering.pdf

- Metsis, Vangelis, Ion Androutsopoulos, and Georgios Paliouras. "Spam filtering with naive bayes-which naive bayes?." In *CEAS*, vol. 17, pp. 28-69. 2006.