

Online Advertising

[SECTIONS](#)[SEARCH](#)

U.S. INTERNATIONAL 中文

statueofliberty



The New York Times

Monday, January 12, 2015

[Today's Paper](#)[Video](#)

38°F

Hang Seng +0.51% ↑

[World](#) [U.S.](#) [Politics](#) [New York](#)[Business](#) [Opinion](#) [Technology](#) [Science](#) [Health](#) [Sports](#) [Arts](#) [Style](#) [Food](#) [Home](#) [Travel](#) [Magazine](#) [Real Estate](#) [ALL](#)**VOLVO**[EXPAND ▾](#)

Times Reporter Will Not Be Called to Testify in Leak Case

By MATT APPLIED 9:00 PM ET

The decision ends a seven-year legal fight over whether James Risen could be forced to name the sources of his reports on a botched C.I.A. operation.

[#39 Comments](#)

The Jiaozhou Bay Bridge, which cost \$2.3 billion, is the world's longest sea-crossing bridge.

Van Shantou/ChinaFotoPress via Associated Press[France Orders](#)

1 of 2 < >

The Opinion Pages

Choke First, Ask Questions Later

By THE EDITORIAL BOARD

A new report suggests that this disavowed tactic has never gone away and sometimes officers use it as a first, not last, resort.

- Editorial: United in Outrage
- Sheryl Sandberg and Adam Grant: Speaking While Female
- Taking Note: The Sony Hack and the Gender Pay Gap
- The Stone: Why Life Is Absurd

MENAGERIE

A Swarm in 'Dead City'

By GABRIELLE SELZ

At 14, I tried to run away. But millions of molting cicadas came between me and my freedom.



- Blow: Tamir Rice and the Value of Life
- Krugman: For the Love of Carbon
- Rosen for Debate: When Satire Cuts Both Ways
- Bruni, Douthat: Movies and Our Still-Wrenching History

Online Advertising is Big Business

Multiple billion dollar industry

\$43B in 2013 in USA, 17% increase over 2012

[PWC, Internet Advertising Bureau, April 2013]

Higher revenue in USA than cable TV and nearly
the same as broadcast TV

[PWC, Internet Advertising Bureau, Oct 2013]

Large source of revenue for Google and other
search engines



Canonical Scalable ML Problem

Problem is hard; we need all the data we can get!

- Success varies by type of online ad (banner, sponsor search, email, etc.) and by ad campaign, but can be less than 1% [Andrew Stern, iMedia Connection, 2010]

Lots of Data

- Lots of people use the internet
- Easy to gather labeled data



A great success story for scalable ML

The Players

Publishers: NYTimes, Google, ESPN

- Make money displaying ads on their sites

Advertisers: Marc Jacobs, Fossil, Macy's, Dr. Pepper

- Pay for their ads to be displayed on publisher sites
- They want to attract business

Matchmakers: Google, Microsoft, Yahoo

- Match publishers with advertisers
- In real-time (i.e., as a specific user visits a website)

Why Advertisers Pay?

Impressions

- Get message to target audience
- e.g., brand awareness campaign

Performance

- Get users to do something
- e.g., click on ad (pay-per-click) ←———— Most common
- e.g., buy something or join a mailing list

Efficient Matchmaking

Idea: Predict probability that user will click each ad and choose ads to maximize probability

- Estimate $\mathbb{P}(\text{click} \mid \text{predictive features})$
- Conditional probability: probability **given** predictive features

Predictive features

- Ad's historical performance
- Advertiser and ad content info
- Publisher info
- User info (e.g., search / click history)



Publishers Get Billions of Impressions Per Day

But, data is **high-dimensional, sparse, and skewed**

- Hundreds of millions of online users
- Millions of unique publisher pages to display ads
- Millions of unique ads to display
- Very few ads get clicked by users

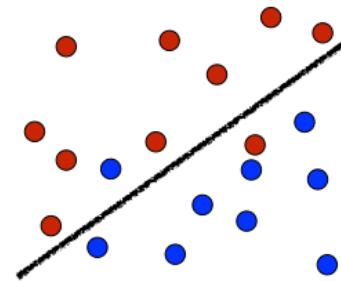
Massive datasets are crucial to tease out signal

Goal: Estimate $\mathbb{P}(\text{click} \mid \text{user, ad, publisher info})$

Given: Massive amounts of labeled data

Linear Classification and Logistic Regression

Classification

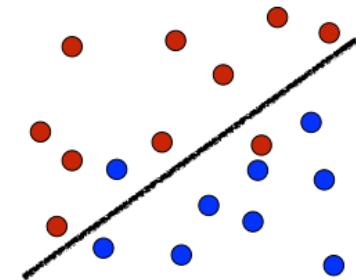


Goal: Learn a mapping from observations to discrete labels given a set of training examples (supervised learning)

Example: Spam Classification

- Observations are emails
- Labels are {spam, not-spam} (Binary Classification)
- Given a set of labeled emails, we want to predict whether a new email is spam or not-spam

Classification

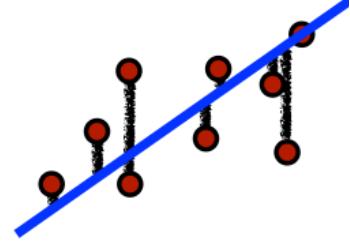


Goal: Learn a mapping from observations to discrete labels given a set of training examples (supervised learning)

Example: Click-through Rate Prediction

- Observations are user-ad-publisher triples
- Labels are {not-click, click} (Binary Classification)
- Given a set of labeled observations, we want to predict whether a new user-ad-publisher triple will result in a click

Reminder: Linear Regression



Example: Predicting shoe size from height, gender, and weight

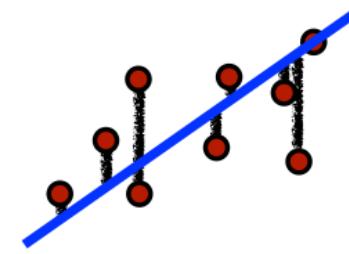
For each observation we have a feature vector, \mathbf{x} , and label, y

$$\mathbf{x}^\top = [x_1 \quad x_2 \quad x_3]$$

We assume a *linear* mapping between features and label:

$$y \approx w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Reminder: Linear Regression



Example: Predicting shoe size from height, gender, and weight

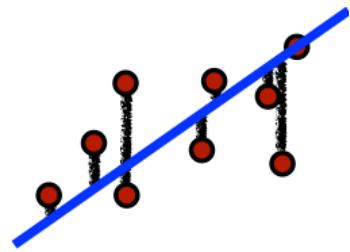
We can augment the feature vector to incorporate offset:

$$\mathbf{x}^\top = [1 \quad x_1 \quad x_2 \quad x_3]$$

We can then rewrite this linear mapping as scalar product:

$$y \approx \hat{y} = \sum_{i=0}^3 w_i x_i = \mathbf{w}^\top \mathbf{x}$$

Why a Linear Mapping?



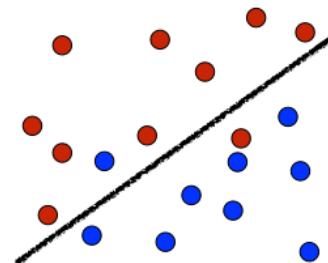
Simple

Often works well in practice

Can introduce complexity via feature extraction

Can we do something similar for classification?

Linear Regression \Rightarrow Linear Classifier



Example: Predicting rain from temperature, cloudiness, and humidity

Use the same feature representation: $\mathbf{x}^\top = [1 \quad x_1 \quad x_2 \quad x_3]$

How can we make class predictions?

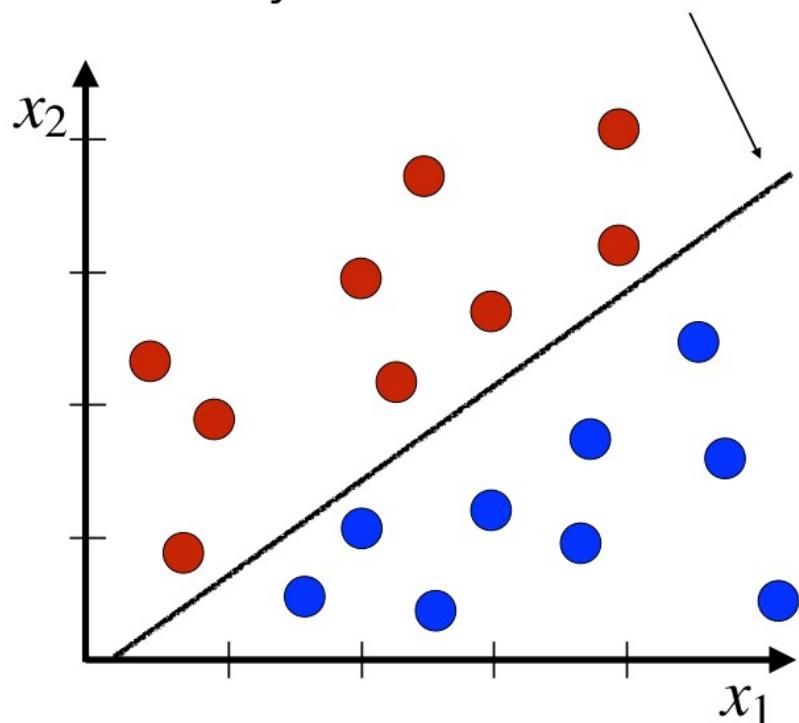
- {not-rain, rain}, {not-spam, spam}, {not-click, click}
- We can threshold by sign

$$\hat{y} = \sum_{i=0}^3 w_i x_i = \mathbf{w}^\top \mathbf{x} \implies \hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x})$$

Linear Classifier Decision Boundary

Decision
Boundary

$$3x_1 - 4x_2 - 1 = 0$$



Imagine $\mathbf{w}^\top = [-1 \quad 3 \quad -4]$

$$\mathbf{x}^\top = [1 \quad 2 \quad 3] : \mathbf{w}^\top \mathbf{x} = -7$$

$$\mathbf{x}^\top = [1 \quad 2 \quad 1] : \mathbf{w}^\top \mathbf{x} = 1$$

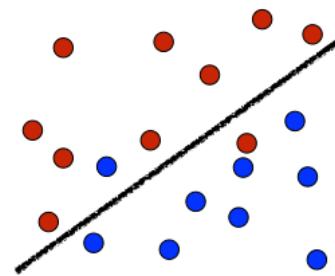
$$\mathbf{x}^\top = [1 \quad 5 \quad .5] : \mathbf{w}^\top \mathbf{x} = 12$$

$$\mathbf{x}^\top = [1 \quad 3 \quad 2.5] : \mathbf{w}^\top \mathbf{x} = -2$$

Let's interpret this rule: $\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x})$

- $\hat{y} = 1 : \mathbf{w}^\top \mathbf{x} > 0$
- $\hat{y} = -1 : \mathbf{w}^\top \mathbf{x} < 0$
- Decision boundary: $\mathbf{w}^\top \mathbf{x} = 0$

Evaluating Predictions



Regression: can measure ‘closeness’ between label and prediction

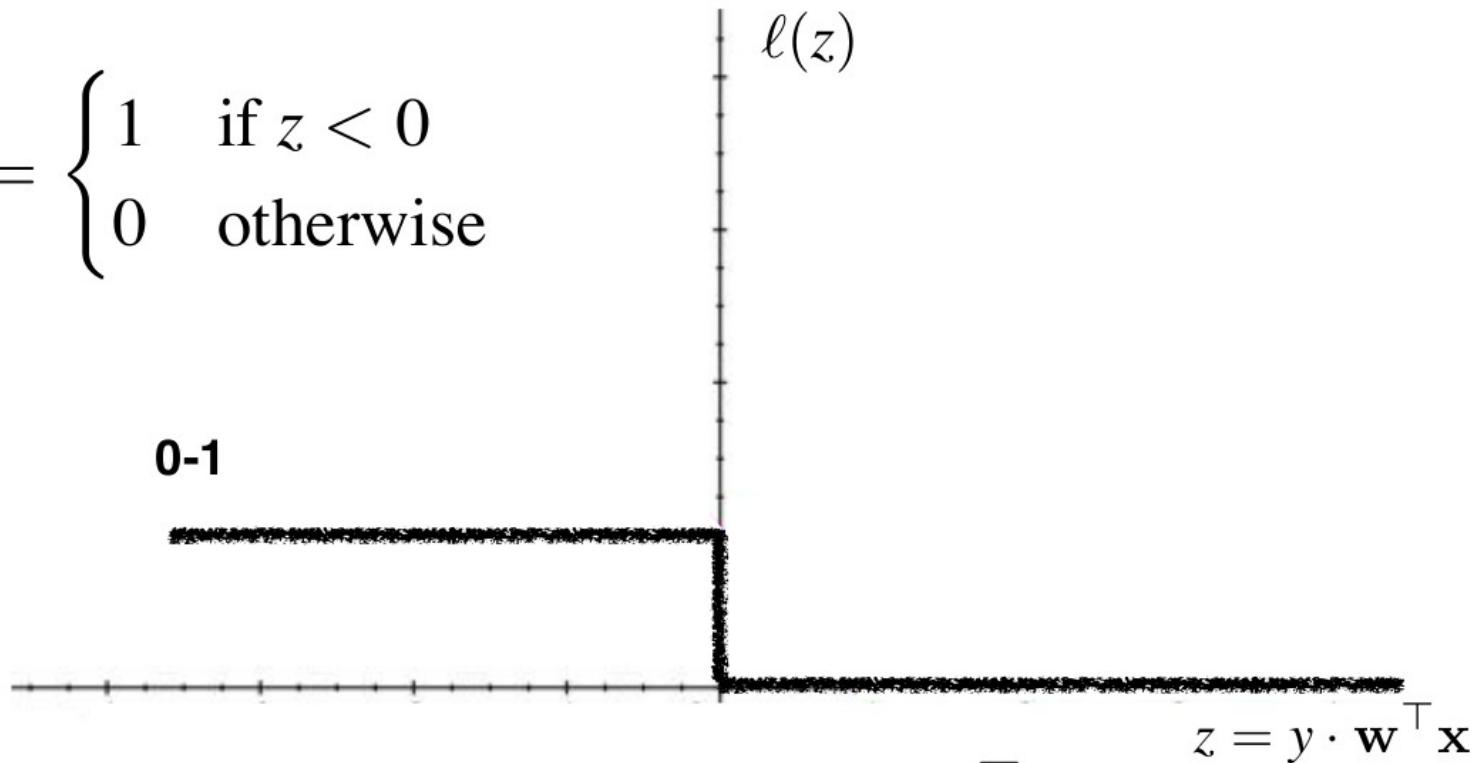
- Song year prediction: better to be off by a year than by 20 years
- Squared loss: $(y - \hat{y})^2$

Classification: Class predictions are discrete

- 0-1 loss: Penalty is 0 for correct prediction, and 1 otherwise

0/1 Loss Minimization

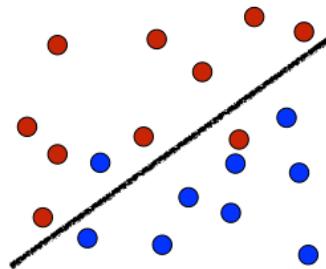
$$\ell_{0/1}(z) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{otherwise} \end{cases}$$



Let $y \in \{-1, 1\}$ and define $z = y \cdot \mathbf{w}^\top \mathbf{x}$

z is positive if y and $\mathbf{w}^\top \mathbf{x}$ have same sign, negative otherwise

How Can We Learn Model (\mathbf{w})?



Assume we have n training points, where $\mathbf{x}^{(i)}$ denotes the i th point

Recall two earlier points:

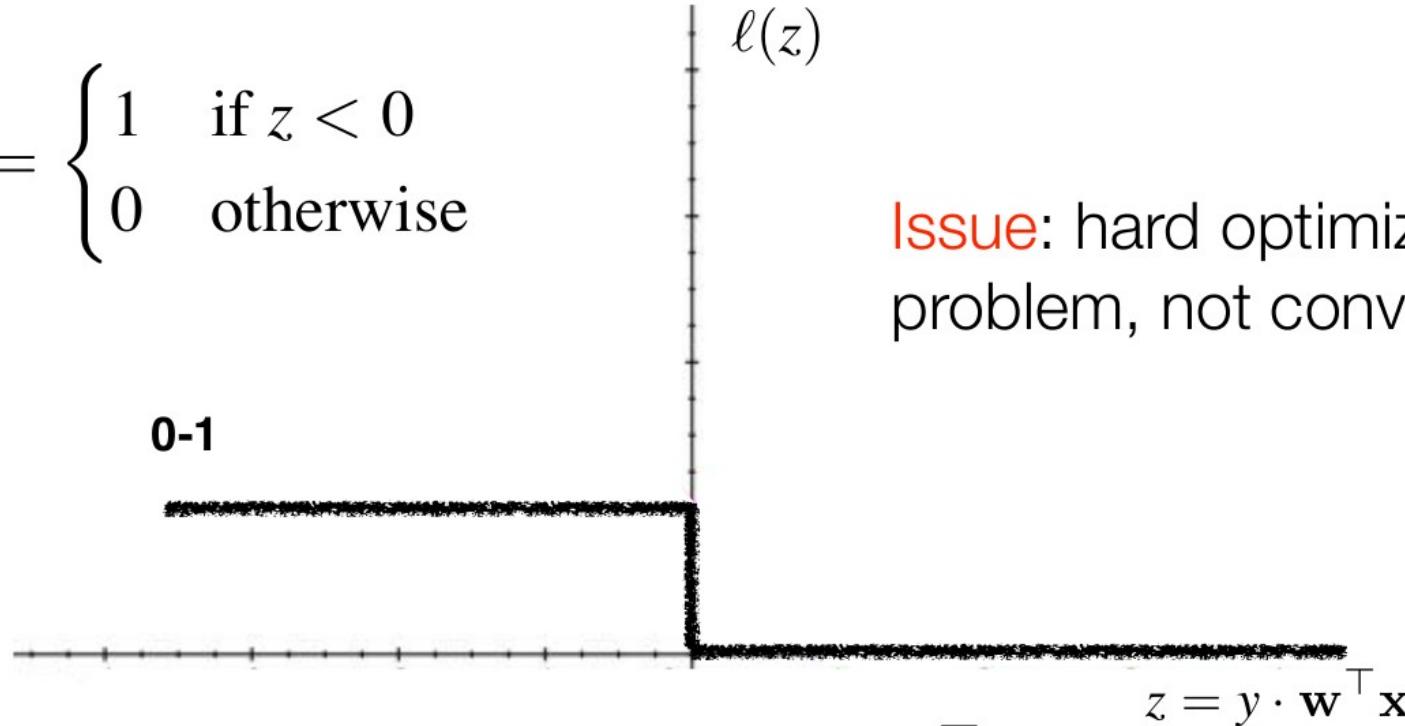
- *Linear assumption:* $\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x})$
- We use 0-1 loss: $\ell_{0/1}(z)$

Idea: Find \mathbf{w} that minimizes average 0-1 loss over training points:

$$\min_{\mathbf{w}} \sum_{i=1}^n \ell_{0/1}\left(y^{(i)} \cdot \mathbf{w}^\top \mathbf{x}^{(i)}\right)$$

0/1 Loss Minimization

$$\ell_{0/1}(z) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{otherwise} \end{cases}$$

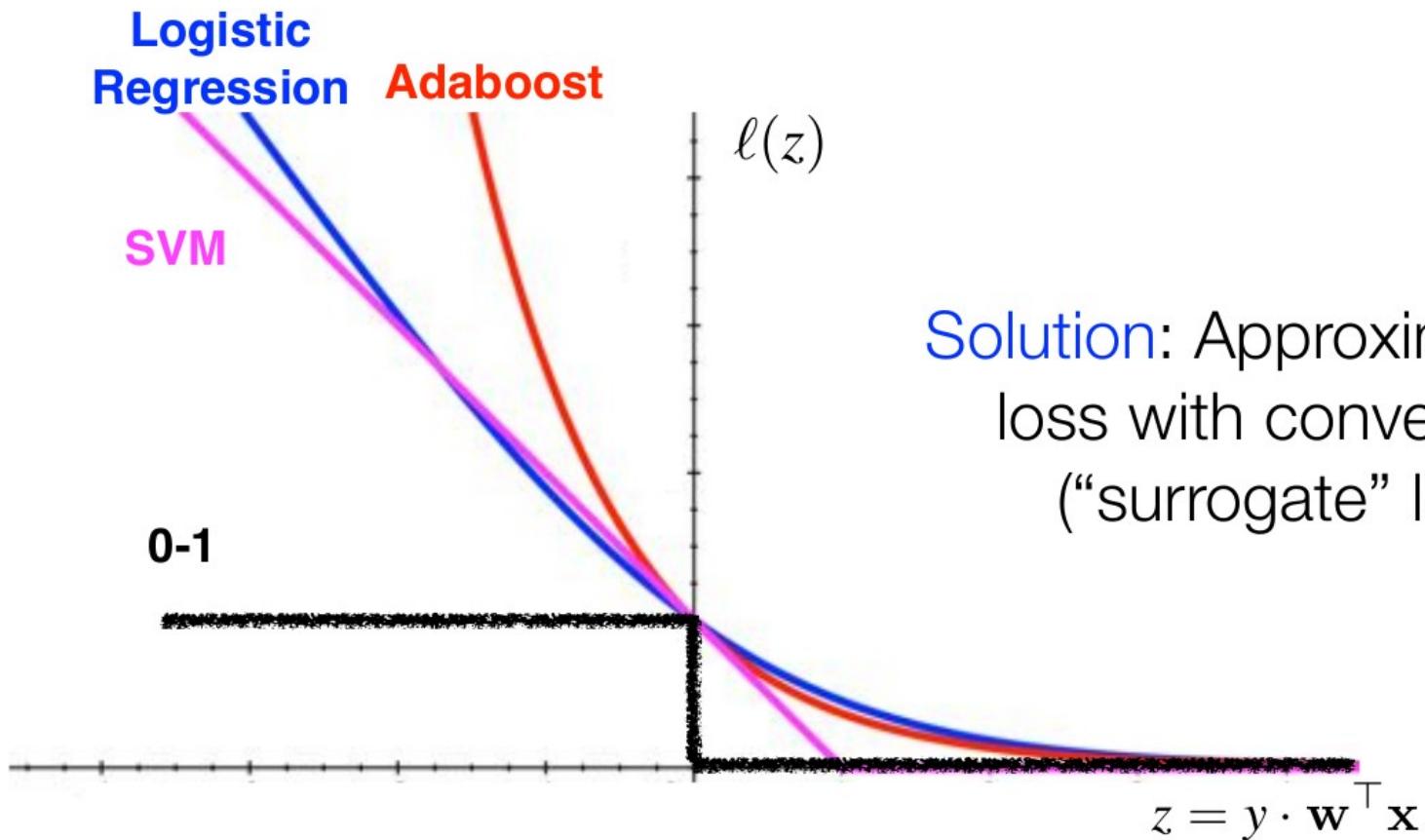


Issue: hard optimization problem, not convex!

Let $y \in \{-1, 1\}$ and define $z = y \cdot \mathbf{w}^\top \mathbf{x}$

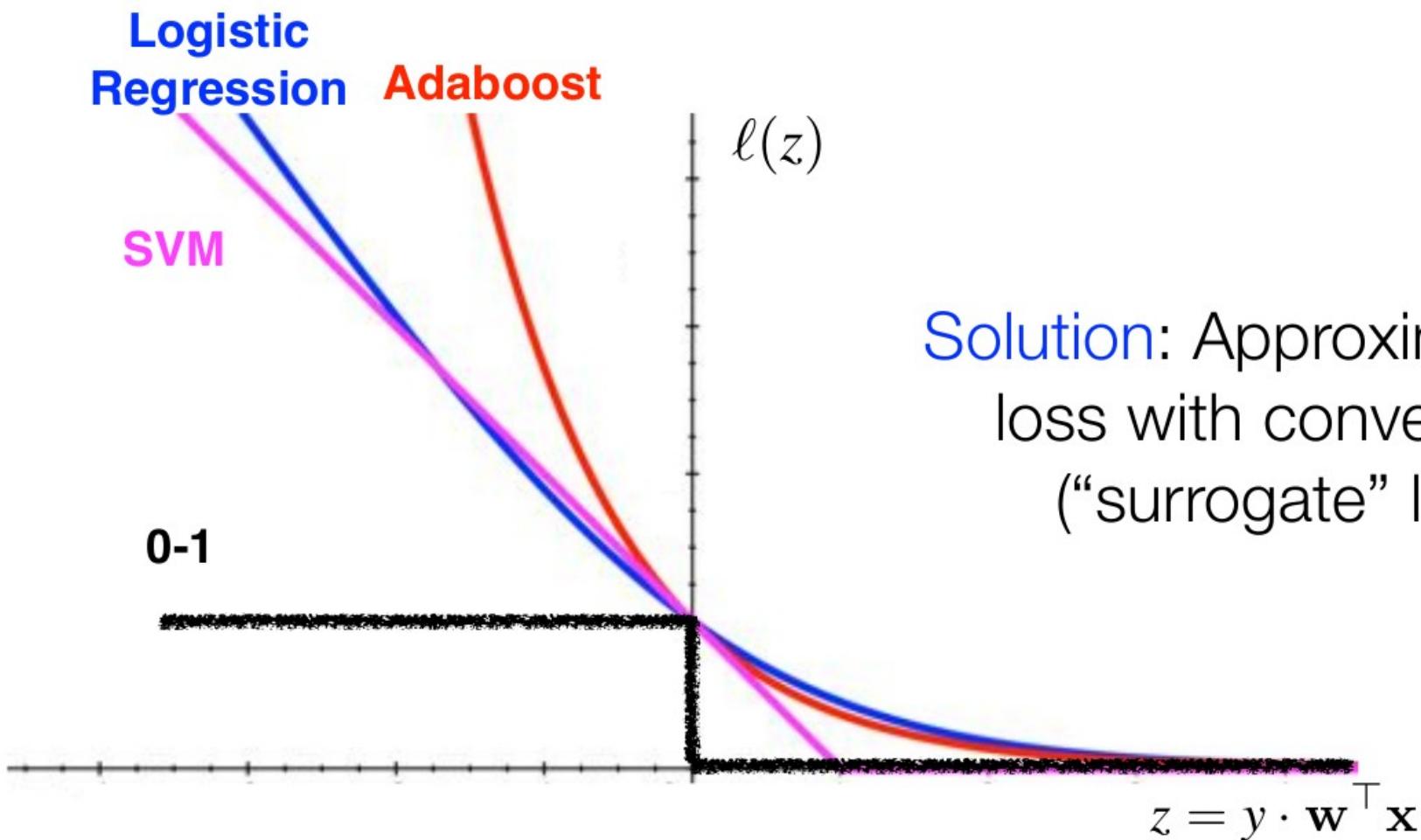
z is positive if y and $\mathbf{w}^\top \mathbf{x}$ have same sign, negative otherwise

Approximate 0/1 Loss



SVM (hinge), Logistic regression (logistic), Adaboost (exponential)

Approximate 0/1 Loss



Solution: Approximate 0/1 loss with convex loss (“surrogate” loss)

$$\text{Logistic loss (logloss): } \ell_{\log}(z) = \log(1 + e^{-z})$$

Logistic Regression Optimization

Logistic Regression: Learn mapping (\mathbf{w}) that minimizes logistic loss on training data

$$\min_{\mathbf{w}} \sum_{i=1}^n \ell_{log}\left(y^{(i)} \cdot \mathbf{w}^\top \mathbf{x}^{(i)}\right)$$

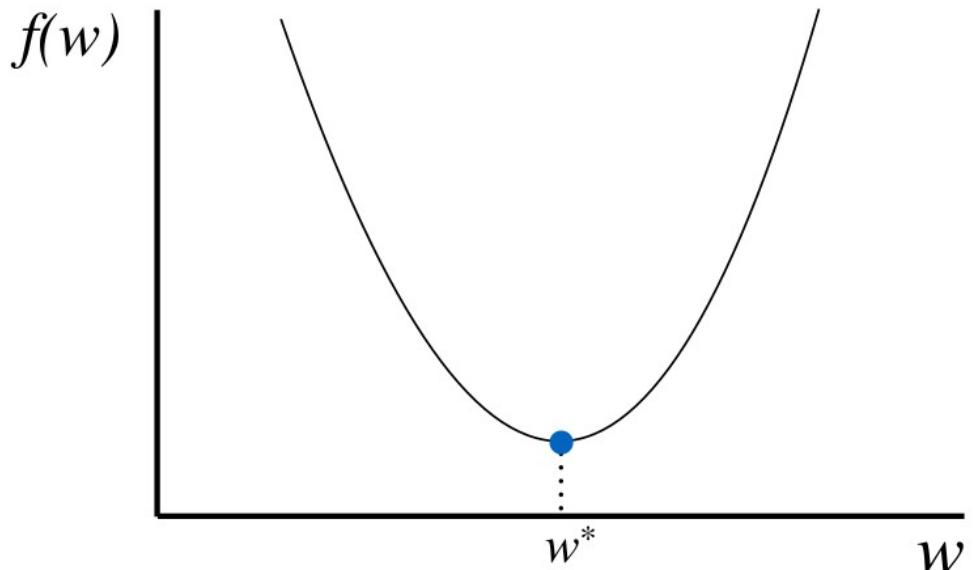
- Convex
- Closed form solution doesn't exist

Logistic Regression Optimization

Goal: Find \mathbf{w}^* that minimizes

$$f(\mathbf{w}) = \sum_{i=1}^n \ell_{log}\left(y^{(i)} \cdot \mathbf{w}^\top \mathbf{x}^{(i)}\right)$$

- Can solve via Gradient Descent



Update Rule: $\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha \nabla f(\mathbf{w})$

$$\sum_{j=1}^n \left[1 - \frac{1}{1 + \exp(-y^{(j)} \mathbf{w}_i^\top \mathbf{x}^{(j)})} \right] (-y^{(j)} \mathbf{x}^{(j)})$$

Step Size

Gradient

Logistic Regression Optimization

Regularized

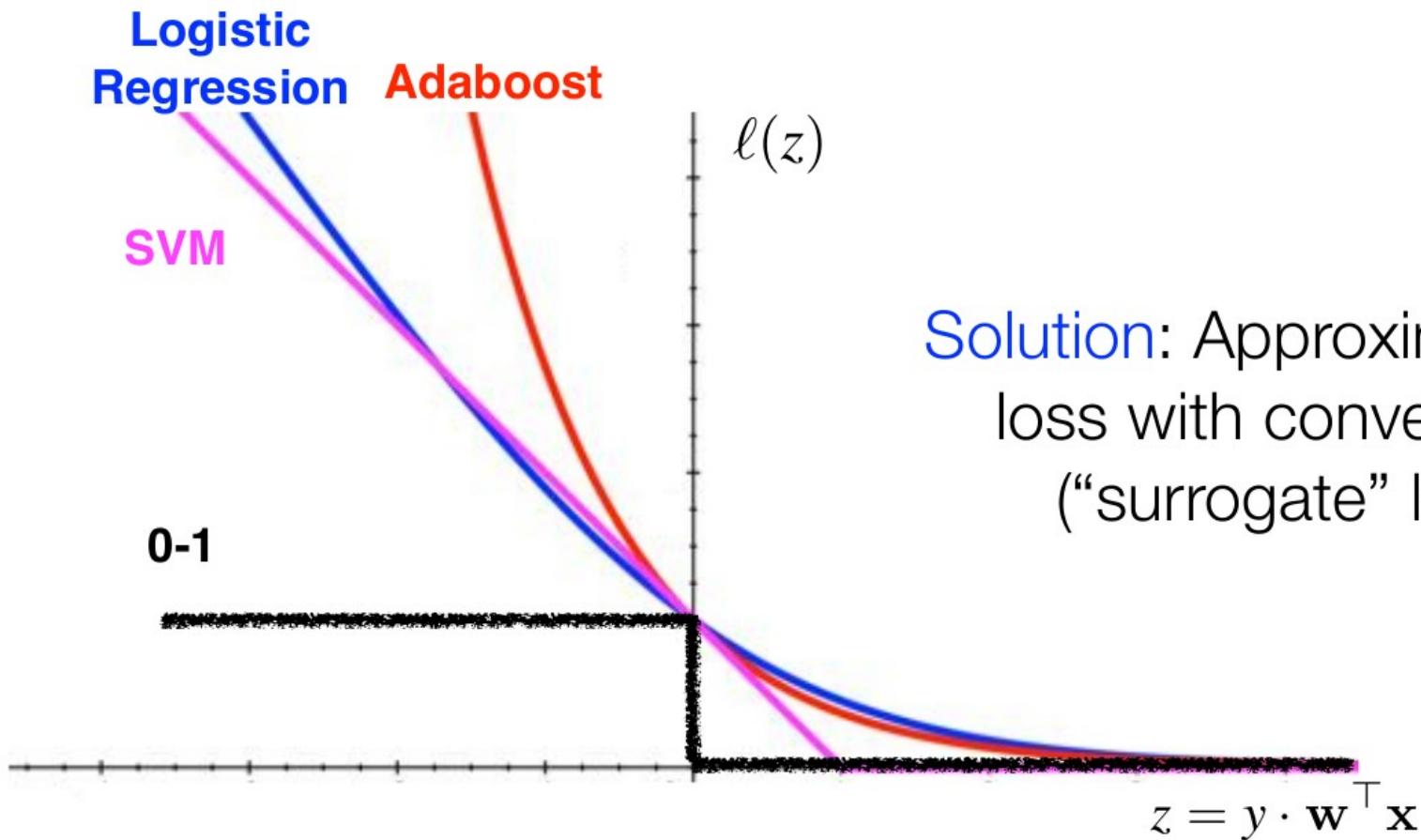
Logistic Regression: Learn mapping (\mathbf{w}) that minimizes logistic loss on training data with a regularization term

$$\min_{\mathbf{w}} \sum_{i=1}^n \frac{\text{Training LogLoss}}{\ell_{log}\left(y^{(i)} \cdot \mathbf{w}^\top \mathbf{x}^{(i)}\right)} + \frac{\text{Model Complexity}}{\lambda \|\mathbf{w}\|_2^2}$$

- Convex
- Closed form solution doesn't exist
- Can add regularization term (as in ridge regression)

Logistic Regression: Probabilistic Interpretation

Approximate 0/1 Loss



Solution: Approximate 0/1 loss with convex loss (“surrogate” loss)

SVM (hinge), Logistic regression (logistic), Adaboost (exponential)

Probabilistic Interpretation

Goal: Model conditional probability: $\mathbb{P}[y = 1 | \mathbf{x}]$

Example: Predict rain from temperature, cloudiness, humidity

- $\mathbb{P}[y = \text{rain} | t = 14^\circ\text{F}, c = \text{LOW}, h = 2\%] = .05$
- $\mathbb{P}[y = \text{rain} | t = 70^\circ\text{F}, c = \text{HIGH}, h = 95\%] = .9$

Example: Predict click from ad's historical performance, user's click frequency, and publisher page's relevance

- $\mathbb{P}[y = \text{click} | h = \text{GOOD}, f = \text{HIGH}, r = \text{HIGH}] = .1$
- $\mathbb{P}[y = \text{click} | h = \text{BAD}, f = \text{LOW}, r = \text{LOW}] = .001$

Probabilistic Interpretation

Goal: Model conditional probability: $\mathbb{P}[y = 1 | \mathbf{x}]$

First thought: $\mathbb{P}[y = 1 | \mathbf{x}] \cancel{=} \mathbf{w}^\top \mathbf{x}$

- Linear regression returns any real number, but probabilities range from 0 to 1!

How can we transform or ‘squash’ its output?

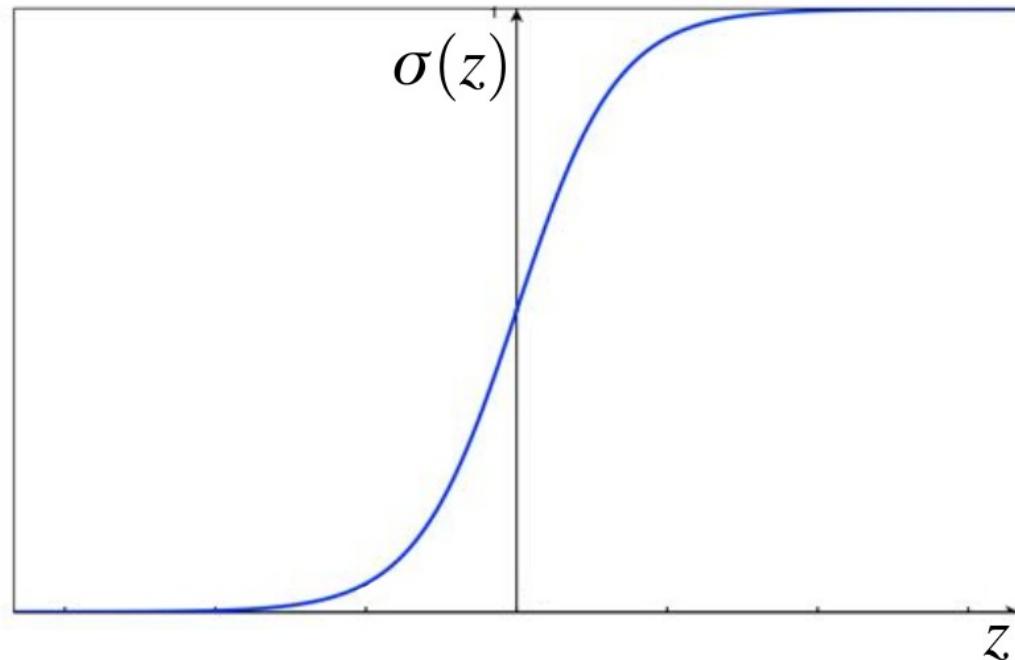
- Use logistic (or sigmoid) function:

$$\mathbb{P}[y = 1 | \mathbf{x}] = \sigma(\mathbf{w}^\top \mathbf{x})$$

Logistic Function

Maps real numbers to $[0, 1]$

- Large positive inputs $\Rightarrow 1$
- Large negative inputs $\Rightarrow 0$



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

Probabilistic Interpretation

Goal: Model conditional probability: $\mathbb{P}[y = 1 | \mathbf{x}]$

Logistic regression uses logistic function to model this conditional probability

- $\mathbb{P}[y = 1 | \mathbf{x}] = \sigma(\mathbf{w}^\top \mathbf{x})$
- $\mathbb{P}[y = 0 | \mathbf{x}] = 1 - \sigma(\mathbf{w}^\top \mathbf{x})$

For notational convenience we now define $y \in \{0, 1\}$

How Do We Use Probabilities?

To make class predictions, we need to convert probabilities to values in $\{0, 1\}$

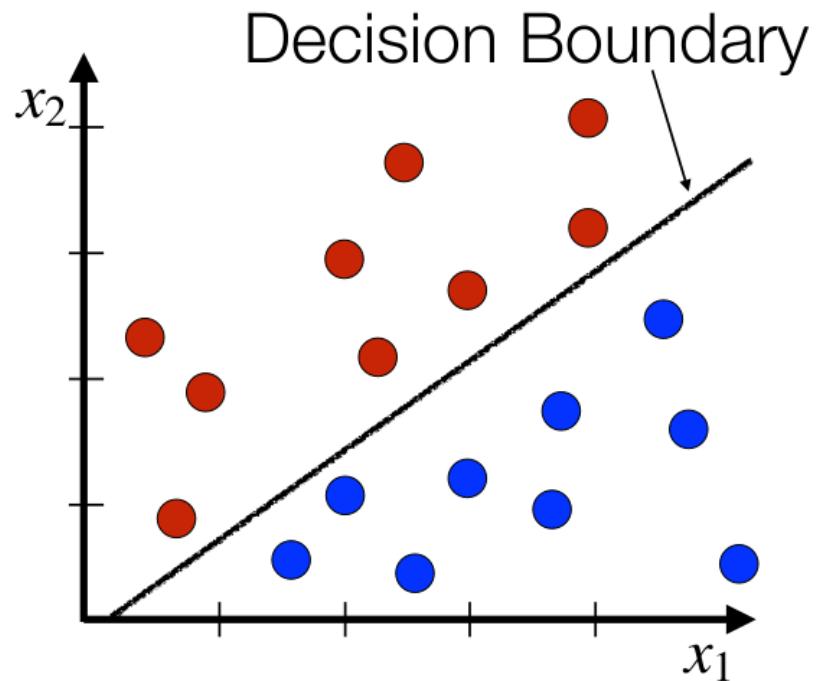
We can do this by setting a threshold on the probabilities

- Default threshold is 0.5
- $\mathbb{P}[y = 1 | \mathbf{x}] > 0.5 \implies \hat{y} = 1$

Example: Predict rain from temperature, cloudiness, humidity

- $\mathbb{P}[y = \text{rain} | t = 14^\circ\text{F}, c = \text{LOW}, h = 2\%] = .05 \quad \hat{y} = 0$
- $\mathbb{P}[y = \text{rain} | t = 70^\circ\text{F}, c = \text{HIGH}, h = 95\%] = .9 \quad \hat{y} = 1$

Connection with Decision Boundary?



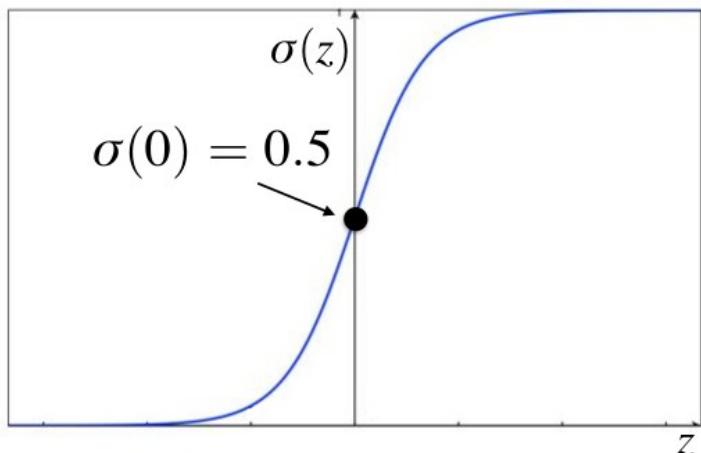
Threshold by sign to make class predictions: $\hat{y} = \text{sign}(w^\top x)$

- $\hat{y} = 1 : w^\top x > 0$
- $\hat{y} = 0 : w^\top x < 0$
- decision boundary: $w^\top x = 0$

How does this compare with thresholding probability?

- $\mathbb{P}[y = 1 | x] = \sigma(w^\top x) > 0.5 \implies \hat{y} = 1$

Connection with Decision Boundary?



$$\mathbf{w}^\top \mathbf{x} = 0 \iff \sigma(\mathbf{w}^\top \mathbf{x}) = 0.5$$

Threshold by sign to make class predictions: $\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x})$

- $\hat{y} = 1 : \mathbf{w}^\top \mathbf{x} > 0$
- $\hat{y} = 0 : \mathbf{w}^\top \mathbf{x} < 0$
- decision boundary: $\mathbf{w}^\top \mathbf{x} = 0$

How does this compare with thresholding probability?

- $\mathbb{P}[y = 1 | \mathbf{x}] = \sigma(\mathbf{w}^\top \mathbf{x}) > 0.5 \implies \hat{y} = 1$
- With threshold of 0.5, the decision boundaries are identical!

Using Probabilistic Predictions

How Do We Use Probabilities?

To make class predictions, we need to convert probabilities to values in $\{0, 1\}$

We can do this by setting a threshold on the probabilities

- Default threshold is 0.5
- $\mathbb{P}[y = 1 | \mathbf{x}] > 0.5 \implies \hat{y} = 1$

Example: Predict **rain** from **t**emperature, **c**loudiness, **h**umidity

- $\mathbb{P}[y = \text{rain} | t = 14^\circ\text{F}, c = \text{LOW}, h = 2\%] = .05 \quad \hat{y} = 0$
- $\mathbb{P}[y = \text{rain} | t = 70^\circ\text{F}, c = \text{HIGH}, h = 95\%] = .9 \quad \hat{y} = 1$

Setting different thresholds

In spam detection application, we model $\mathbb{P}[y = \text{spam} | \mathbf{x}]$

Two types of error

- Classify a not-spam email as spam (*false positive*, FP)
- Classify a spam email as not-spam (*false negative*, FN)

Can argue that false positives are more harmful than false negatives

- Worse to miss an important email than to have to delete spam

We can use a threshold greater than 0.5 to be more ‘conservative’

ROC Plots: Measuring Varying Thresholds

ROC plot displays FPR vs TPR

- Top left is perfect
- Dotted Line is random prediction (i.e., biased coin flips)

Can classify at various thresholds (T)

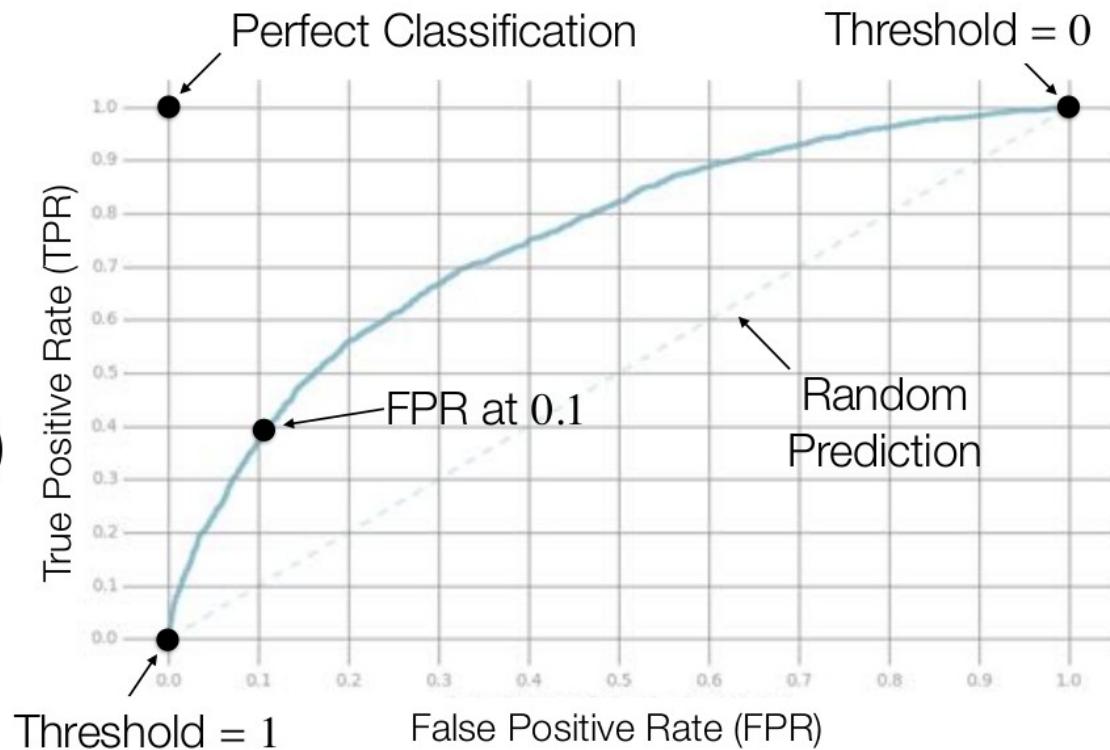
$T = 0$: Everything is spam

- $TPR = 1$, but $FPR = 1$

$T = 1$: Nothing is spam

- $FPR = 0$, but $TPR = 0$

We can tradeoff between TPR/FPR



FPR: % not-spam predicted as spam
TPR: % spam predicted as spam

Working Directly with Probabilities

Example: Predict **click** from ad's **historical** performance, user's click **frequency**, and publisher page's **relevance**

- $\mathbb{P}[y = \text{click} | h = \text{GOOD}, f = \text{HIGH}, r = \text{HIGH}] = .1 \quad \hat{y} = 0$
- $\mathbb{P}[y = \text{click} | h = \text{BAD}, f = \text{LOW}, r = \text{LOW}] = .001 \quad \hat{y} = 0$

Success can be less than 1% [Andrew Stern, iMedia Connection, 2010]

Probabilities provide more granular information

- Confidence of prediction
- Useful when combining predictions with other information

In such cases, we want to evaluate probabilities directly

- Logistic loss makes sense for evaluation!

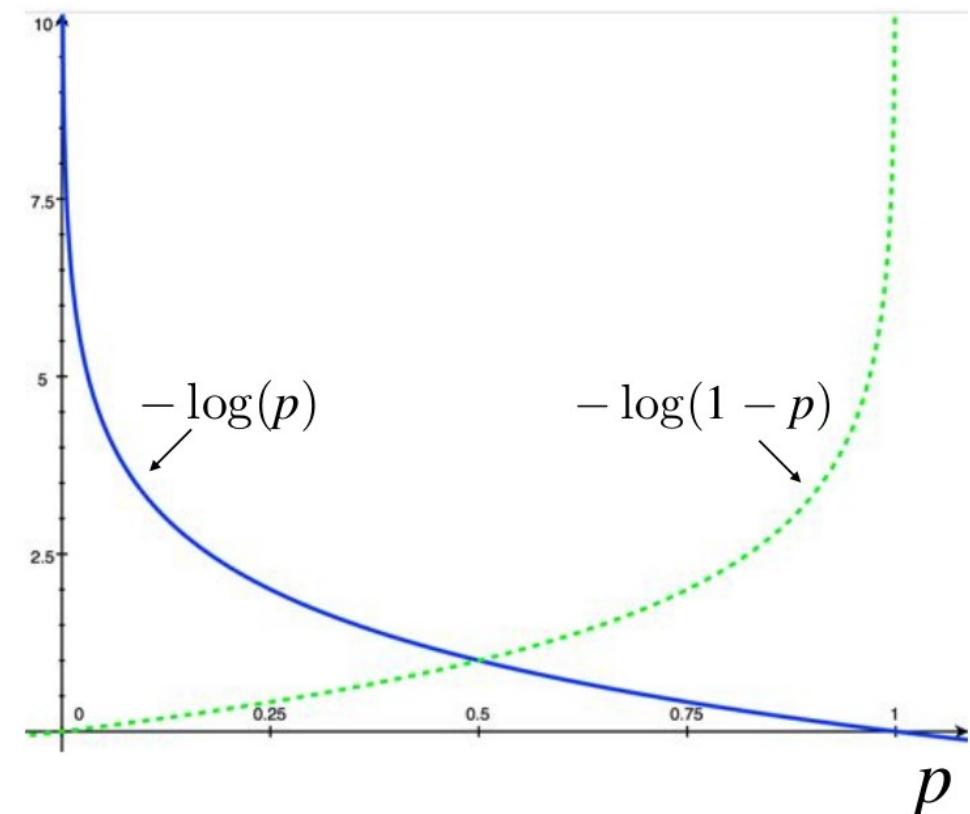
Logistic Loss

$$\ell_{log}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1 - p) & \text{if } y = 0 \end{cases}$$

When $y = 1$, we want $p = 1$

- No penalty at 1
- Increasing penalty away from 1

Similar logic when $y = 0$



Categorical Data and One-Hot-Encoding

Logistic Regression Optimization

Regularized

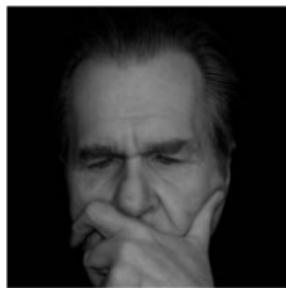
Logistic Regression: Learn mapping (\mathbf{w}) that minimizes logistic loss on training data with a regularization term

$$\min_{\mathbf{w}} \sum_{i=1}^n \frac{\text{Training LogLoss}}{\ell_{0/1}\left(y^{(i)} \cdot \mathbf{w}^\top \mathbf{x}^{(i)}\right)} + \frac{\text{Model Complexity}}{\lambda \|\mathbf{w}\|_2^2}$$

Data is assumed to be **numerical!**

Similar story for linear regression and many other methods

Raw Data is Sometimes Numeric



Images

	★	★★★★	
	★	★★★★	★★
	★★★★		★
	★		★★
		★★★★	★★
	★★★★	★★	

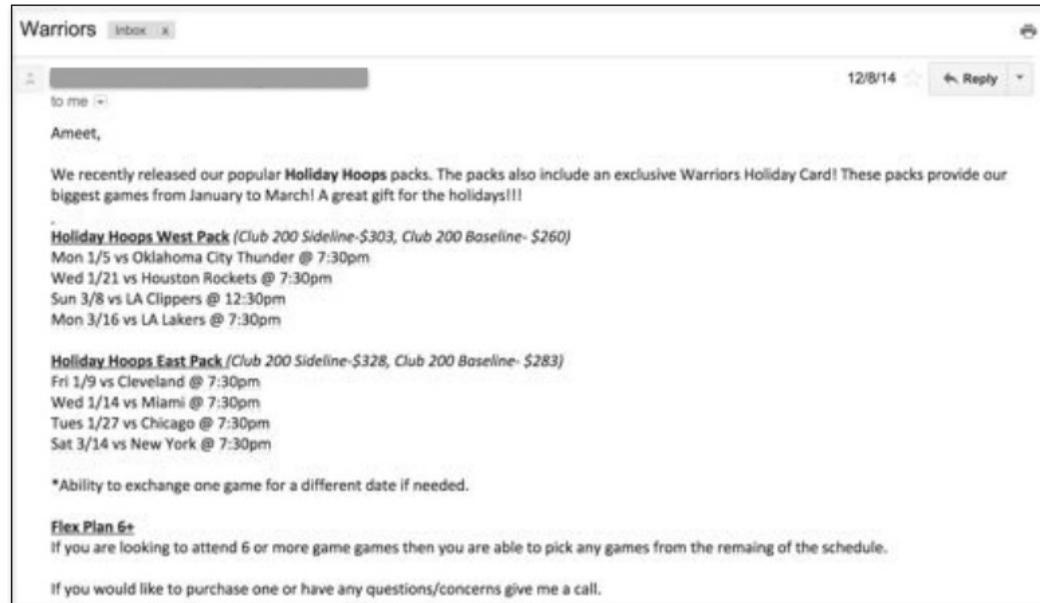
User Ratings

Raw Data is Often Non-Numeric

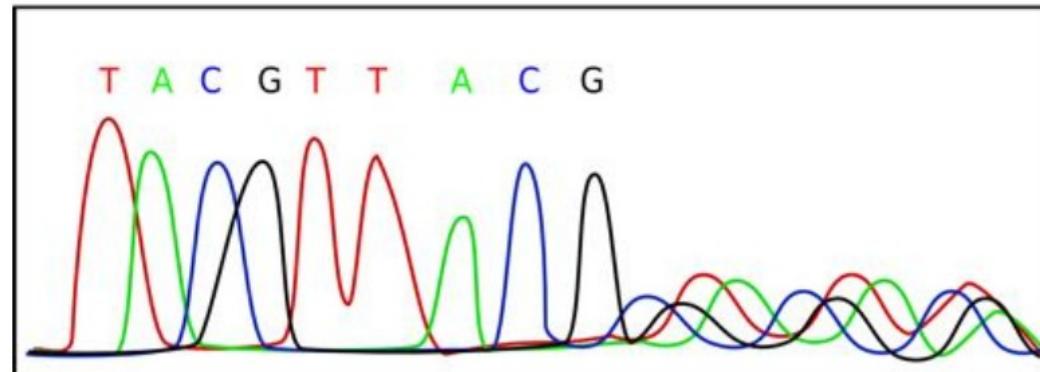
```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
3
4 <html xmlns="http://www.w3.org/1999/
xhtml">
5   <head>
6     <meta http-equiv="Content-
Type" content=
7       "text/html; charset=us-
ascii" />
8     <script type="text/
javascript">
9       function reDo() {top.
location.reload();}
10      if (navigator.appName ==
'Netscape') {top.onresize = reDo;}
11      dom=document.
getElementById;
12      </script>
13    </head>
14    <body>
15    </body>
16 </html>
```

Web hypertext

Email



Genomic
Data



Raw Data is Often Non-Numeric

Example: Click-through Rate Prediction

- User features: Gender, Nationality, Occupation, ...
- Advertiser / Publisher: Industry, Location, ...
- Ad / Publisher Site: Language, Text, Target Audience, ...

How to Handle Non-Numeric Features?

Option 1: Use methods that support these features

- Some methods, e.g., Decision Trees, Naive Bayes, naturally support non-numerical features
- However, this limits our options

Option 2: Convert these features to numeric features

- Allows us to use a wider range of learning methods
- How do we do this?

Types of Non-Numeric Features

Categorical Feature

- Has two or more categories
- No intrinsic ordering to the categories
- E.g., Gender, Country, Occupation, Language

Ordinal Feature

- Has two or more categories
- Intrinsic ordering, but no consistent spacing between categories, i.e., all we have is a relative ordering
- Often seen in survey questions, e.g., “Is your health poor, reasonable, good, excellent”

Non-Numeric \Rightarrow Numeric

One idea: Create single numerical feature to represent non-numeric one

Ordinal Features:

- Health categories = {'poor', 'reasonable', 'good', 'excellent'}
- 'poor' = 1, 'reasonable' = 2, 'good' = 3, 'excellent' = 4

We can use a single numerical feature that preserves this ordering ... but ordinal features only have an ordering and we introduce a degree of closeness that didn't previously exist

Non-Numeric \Rightarrow Numeric

One idea: Create single numerical feature to represent non-numeric one

Categorical Features:

- Country categories = {'ARG', 'FRA', 'USA'}
- 'ARG' = 1, 'FRA' = 2, 'USA' = 3
- Mapping implies FRA is between ARG and USA

Creating single numerical feature introduces relationships between categories that don't otherwise exist

Non-Numeric \Rightarrow Numeric

Another idea (One-Hot-Encoding): Create a ‘dummy’ feature for each category

Categorical Features:

- Country categories = {'ARG', 'FRA', 'USA'}
- We introduce one new dummy feature for each category
- 'ARG' \Rightarrow [1 0 0], 'FRA' \Rightarrow [0 1 0], 'USA' \Rightarrow [0 0 1]

Creating dummy features doesn't introduce spurious relationships

Computing and Storing OHE Features

Example: Categorical Animal Dataset

Features:

- Animal = {'bear', 'cat', 'mouse'}
- Color = {'black', 'tabby'}
- Diet (optional) = {'mouse', 'salmon'}

Datapoints:

- A1 = ['mouse', 'black', -]
- A2 = ['cat', 'tabby', 'mouse']
- A3 = ['bear', 'black', 'salmon']

How can we create OHE features?

Step 1: Create OHE Dictionary

Features:

- Animal = {'bear', 'cat', 'mouse'}
- Color = {'black', 'tabby'}
- Diet = {'mouse', 'salmon'}

7 dummy features in total

- 'mouse' category distinct for Animal and Diet features

OHE Dictionary:

Maps each category to dummy feature

- (Animal, 'bear') \Rightarrow 0
- (Animal, 'cat') \Rightarrow 1
- (Animal, 'mouse') \Rightarrow 2
- (Color, 'black') \Rightarrow 3
- ...

Step 2: Create Features with Dictionary

Datapoints:

- A1 = ['mouse', 'black', -]
- A2 = ['cat', 'tabby', 'mouse']
- A3 = ['bear', 'black', 'salmon']

OHE Features:

- Map non-numeric feature to it's binary dummy feature
- E.g., A1 = [0, 0, 1, 1, 0, 0, 0]



OHE Dictionary: Maps each category to dummy feature

- (Animal, 'bear') ⇒ 0
- (Animal, 'cat') ⇒ 1
- (Animal, 'mouse') ⇒ 2
- (Color, 'black') ⇒ 3
- ...

Step 2: Create Features with Dictionary

Datapoints:

- A1 = ['mouse', 'black', -]
- A2 = ['cat', 'tabby', 'mouse']
- A3 = ['bear', 'black', 'salmon']

OHE Features:

- Map non-numeric feature to its binary dummy feature
- E.g., A1 = [0, 0, 1, 1, 0, 0, 0]



OHE Dictionary: Maps each category to dummy feature

- (Animal, 'bear') ⇒ 0
- (Animal, 'cat') ⇒ 1
- (Animal, 'mouse') ⇒ 2
- (Color, 'black') ⇒ 3
- ...

OHE Features are Sparse

For a given categorical feature only a single OHE feature is non-zero — can we take advantage of this fact?

Dense representation: Store all numbers

- E.g., $A1 = [0, 0, 1, 1, 0, 0, 0]$

Sparse representation: Store indices / values for non-zeros

- Assume all other entries are zero
- E.g., $A1 = [(2,1), (3,1)]$

Sparse Representation

Example: Matrix with 10M observation and 1K features

- Assume 1% non-zeros

Dense representation: Store all numbers

- Store $10M \times 1K$ entries as doubles \Rightarrow 80GB storage

Sparse representation: Store indices / values for non-zeros

- Store value and location for non-zeros (2 doubles per entry)
- 50x savings in storage!
- We will also see computational saving for matrix operations

Feature Hashing

Non-Numeric \Rightarrow Numeric

One-Hot-Encoding: Create a ‘dummy’ feature for each category

Creating dummy features doesn’t introduce spurious relationships

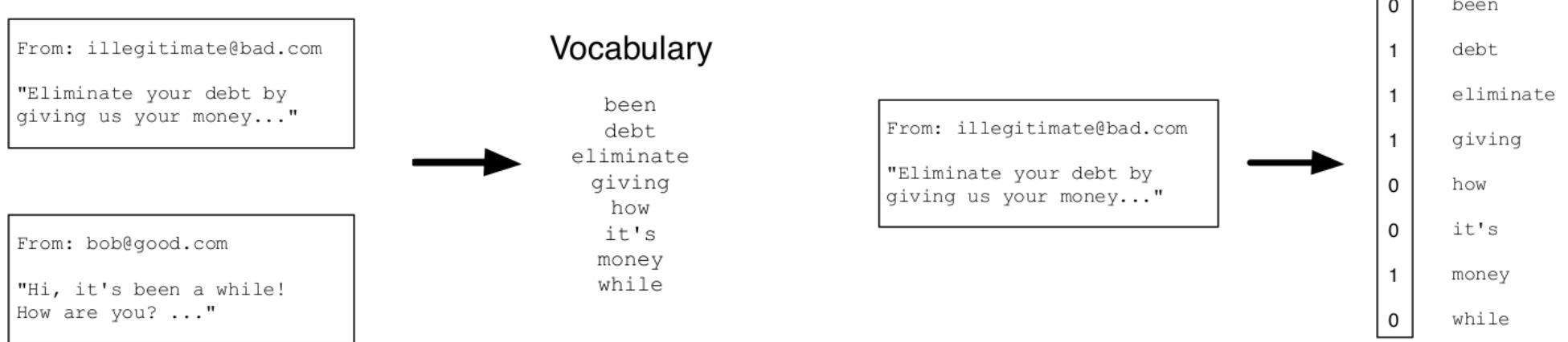
Dummy features can drastically increase dimensionality

- Number of dummy features equals number of categories!

Issue with CTR prediction data

- Includes many names (of products, advertisers, etc.)
- Text from advertisement, publisher site, etc.

“Bag of Words” Representation



Represent each document with a vocabulary of words

Over 1M words in English [Global Language Monitor, 2014]

We sometimes consider bigrams or adjacent words (similar idea to quadratic features)

High Dimensionality of OHE

Statistically: Inefficient learning

- We generally need bigger n when we have bigger d (though in distributed setting we often have very large n)
- We will have many non-predictive features

Computationally: Increased communication

- Linear models have parameter vectors of dimension d
- Gradient descent communicates the parameter vector to all workers at each iteration

How Can We Reduce Dimension?

One Option: Discard rare features

- Might throw out useful information (rare \neq uninformative)
- Must first compute OHE features, which is expensive

Another Option: Feature hashing



Can view as an unsupervised learning preprocessing step

- Use hashing principles to reduce feature dimension
- Obviates need to compute expensive OHE dictionary
- Preserves sparsity
- Theoretical underpinnings

High-Level Idea

Hash tables are an efficient data structure for data lookup, and hash functions also useful in cryptography

Hash Function: Maps an object to one of m buckets

- Should be efficient and distribute objects across buckets

In our setting, objects are feature categories

- We have fewer buckets than feature categories
- Different categories will ‘collide’, i.e., map to same bucket
- Bucket indices are hashed features

Feature Hashing Example

Datapoints: 7 feature categories

- A1 = ['mouse', 'black', -]
- A2 = ['cat', 'tabby', 'mouse']
- A3 = ['bear', 'black', 'salmon']

Hash Function: $m = 4$

- $H(\text{Animal}, \text{'mouse'}) = 3$
- $H(\text{Color}, \text{'black'}) = 2$

Hashed Features:

- A1 = [0 0 1 1]

Feature Hashing Example

Datapoints: 7 feature categories

- A1 = ['mouse', 'black', -]
- A2 = ['cat', 'tabby', 'mouse']
→
- A3 = ['bear', 'black', 'salmon']

Hashed Features:

- A1 = [0 0 1 1]
- A2 = [2 0 1 0]

Hash Function: $m = 4$

- $H(\text{Animal}, \text{'mouse'}) = 3$
- $H(\text{Color}, \text{'black'}) = 2$
- $H(\text{Animal}, \text{'cat'}) = 0$
- $H(\text{Color}, \text{'tabby'}) = 0$
- $H(\text{Diet}, \text{'mouse'}) = 2$

Feature Hashing Example

Datapoints: 7 feature categories

- A1 = ['mouse', 'black', -]
- A2 = ['cat', 'tabby', 'mouse']
- A3 = ['bear', 'black', 'salmon']



Hashed Features:

- A1 = [0 0 1 1]
- A2 = [2 0 1 0]
- A3 = [1 1 1 0]

Hash Function: $m = 4$

- $H(\text{Animal}, \text{'mouse'}) = 3$
- $H(\text{Color}, \text{'black'}) = 2$
- $H(\text{Animal}, \text{'cat'}) = 0$
- $H(\text{Color}, \text{'tabby'}) = 0$
- $H(\text{Diet}, \text{'mouse'}) = 2$
- $H(\text{Animal}, \text{'bear'}) = 0$
- $H(\text{Color}, \text{'black'}) = 2$
- $H(\text{Diet}, \text{'salmon'}) = 1$

Why Is This Reasonable?

Hash features have nice theoretical properties

- Good approximations of inner products of OHE features under certain conditions
- Many learning methods (including linear / logistic regression) can be viewed solely in terms of inner products

Good empirical performance

- Spam filtering and various other text classification tasks

Hashed features are a reasonable alternative for OHE features

Distributed Computation

```
trainHash = train.map(applyHashFunction)  
           .map(createSparseVector)
```

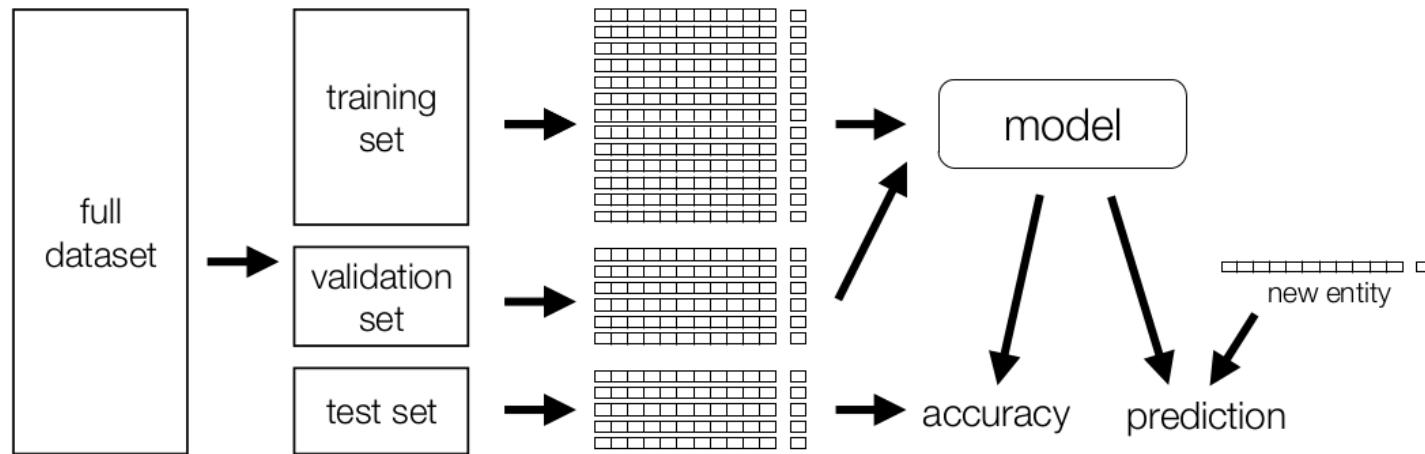
Step 1: Apply hash function on raw data

- Local computation and hash functions are usually fast
- No need to compute OHE features or communication

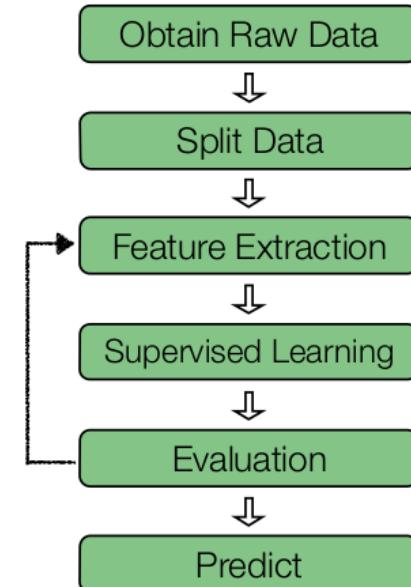
Step 2: Store hashed features in sparse representation

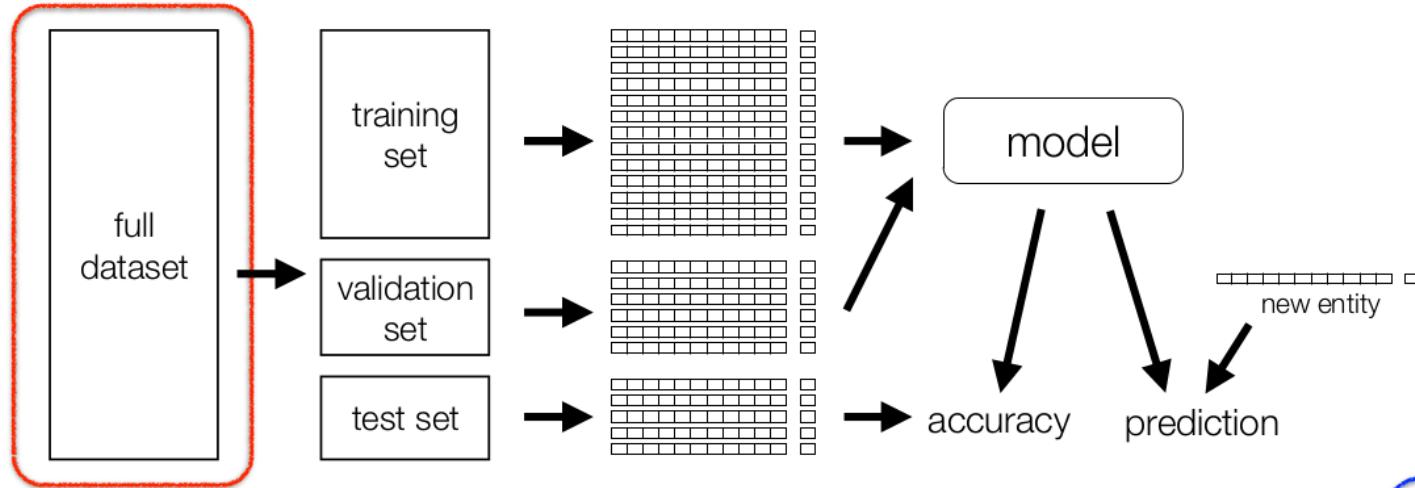
- Local computation

CTR Prediction Pipeline / Lab Preview



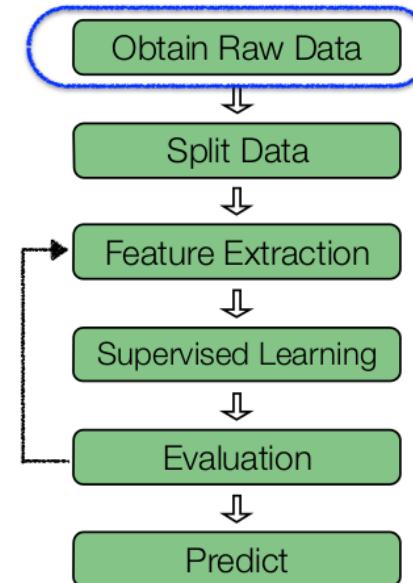
Goal: Estimate $\mathbb{P}(\text{click} \mid \text{user, ad, publisher info})$
Given: Massive amounts of labeled data

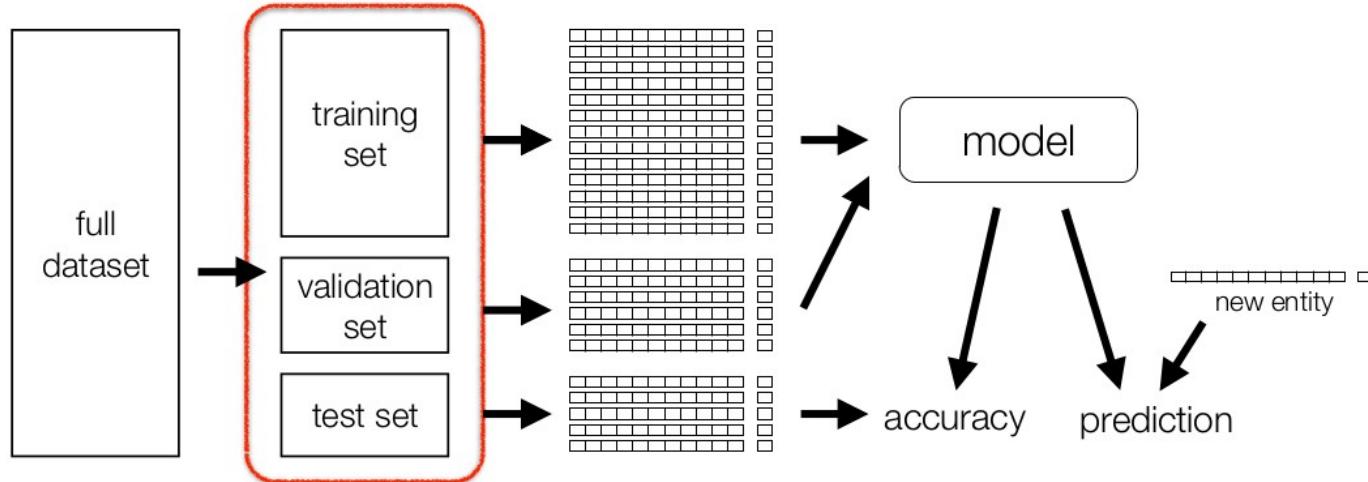




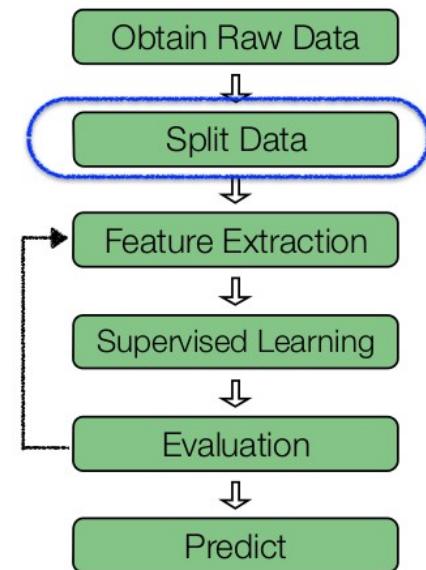
Raw Data: Criteo Dataset from Kaggle competition

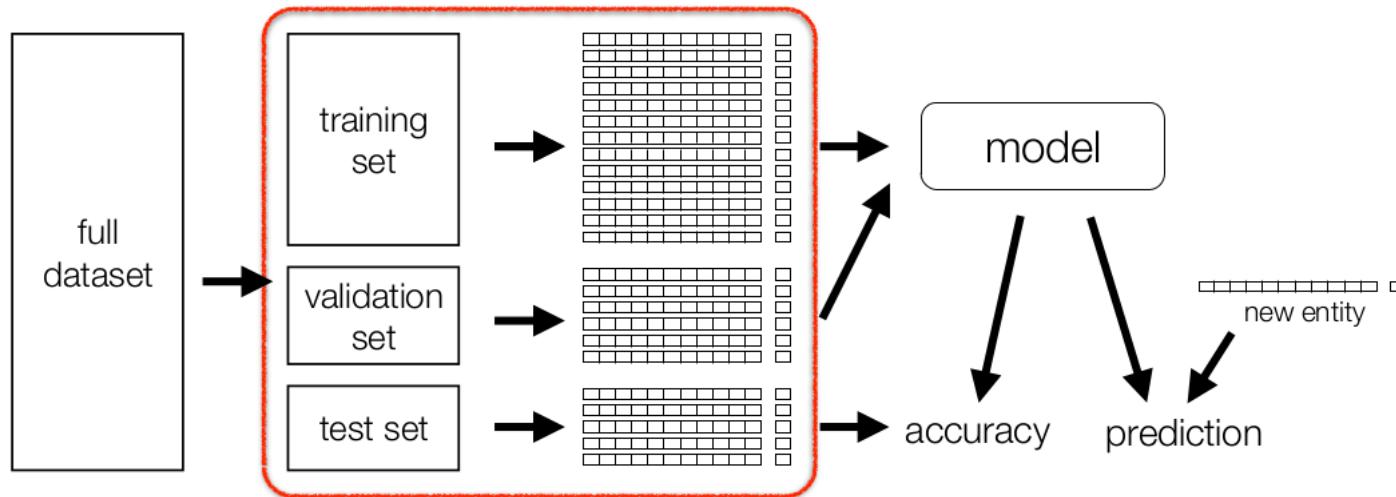
- We'll work with subsample of larger CTR dataset
- 39 masked user, ad and publisher features
- Full Kaggle dataset has 33M distinct categories (and this dataset is a small subset of Criteo's actual data)





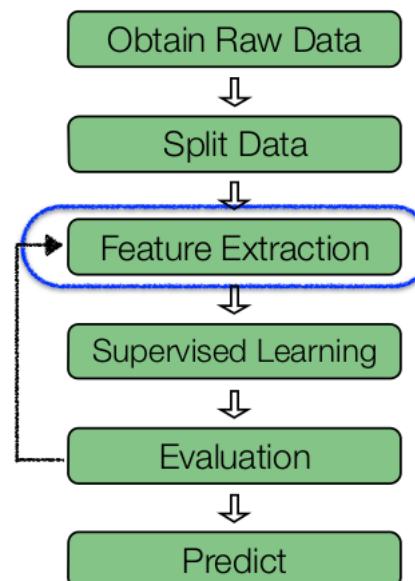
Split Data: Create training, validation, and test sets

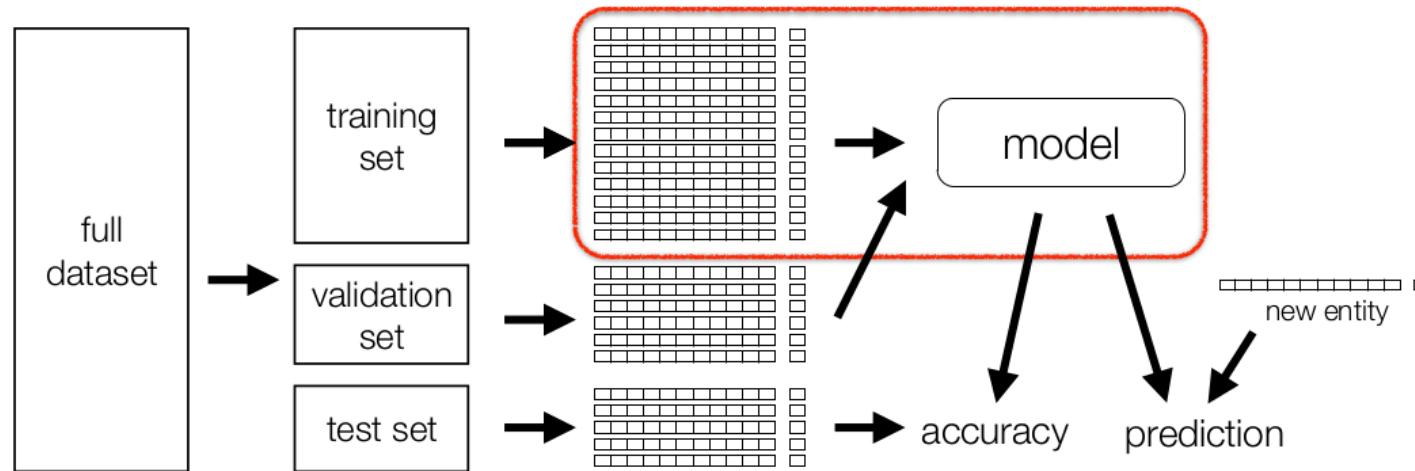




Feature Extraction: One-hot-encoding and feature hashing

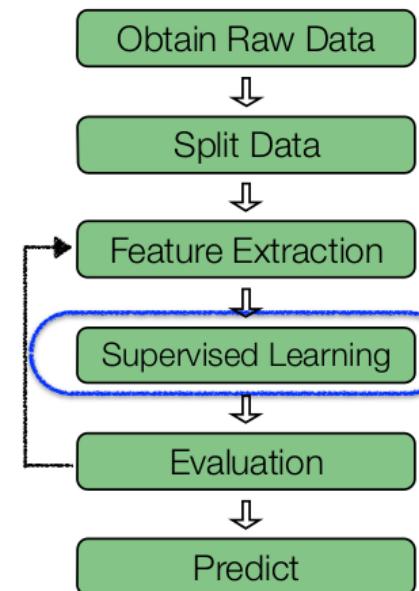
- We'll use a sparse data representation
- We'll visualize feature frequency
- Feature extraction is the main focus of this lab

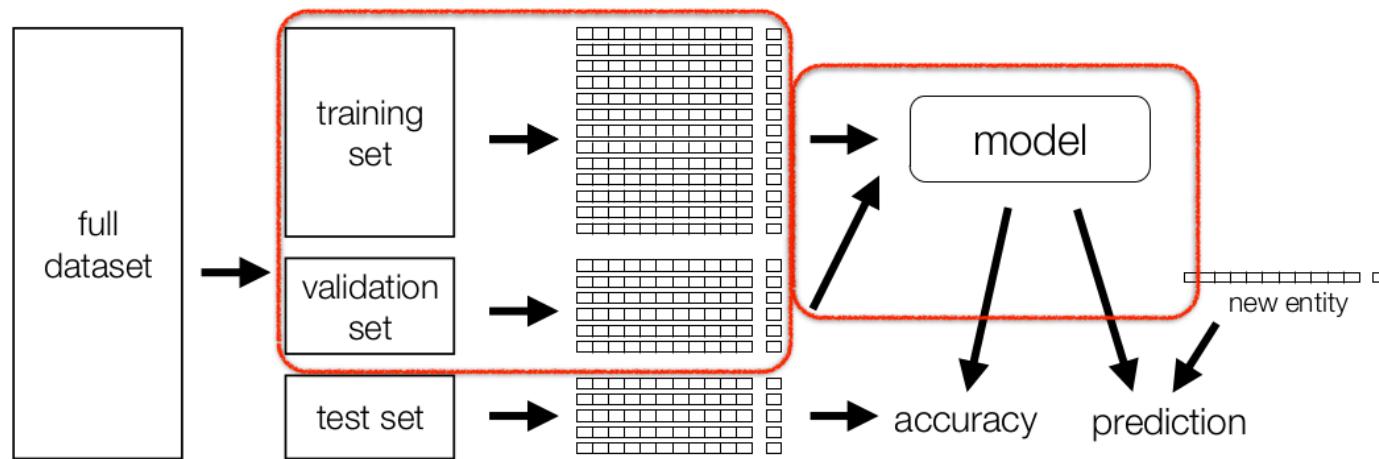




Supervised Learning: Logistic regression

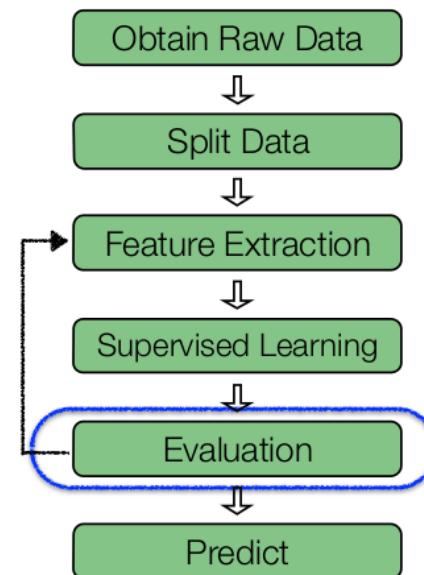
- Use MLlib implementation

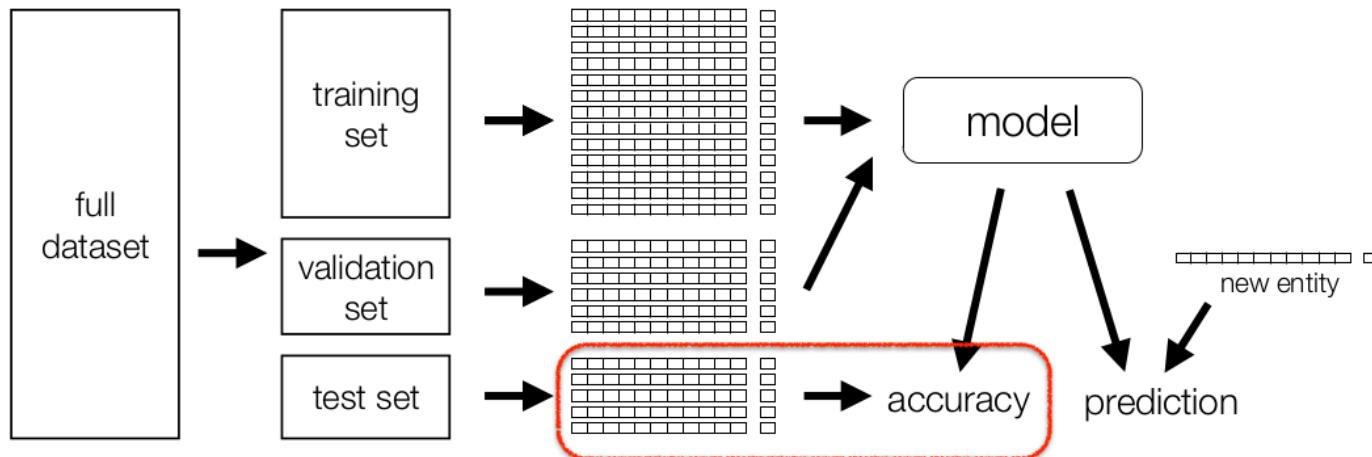




Evaluation (Part 1): Hyperparameter tuning

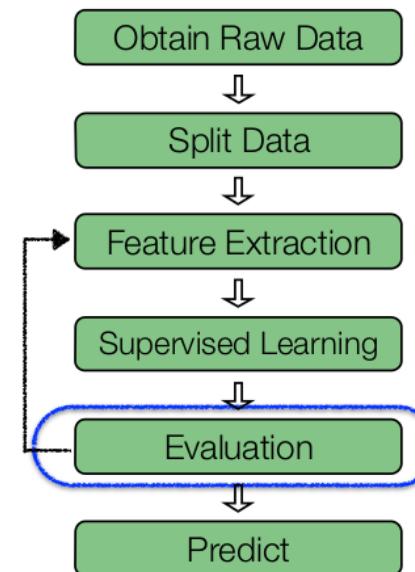
- Grid search to find good values for regularization
- Evaluate using logistic loss
- Visualize grid search
- Visualize predictions via ROC curve

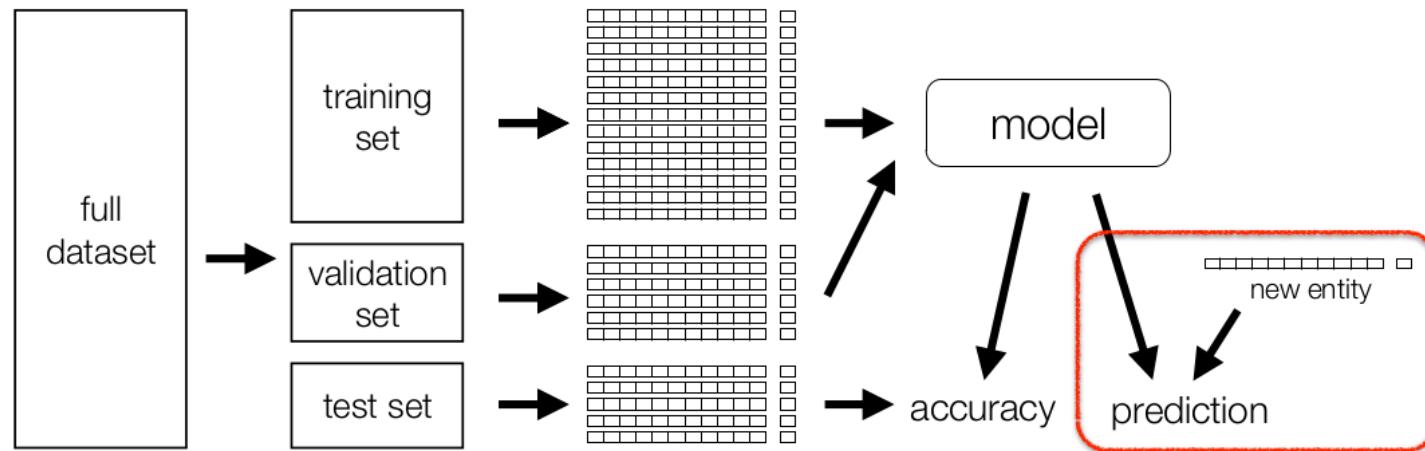




Evaluation (Part 2): Evaluate final model

- Evaluate using logistic loss
- Compare to baseline model (always predict value equal to the fraction of training points that correspond to click-through events)





Predict: Final model could be used to predict click-through rate for new user-ad tuple (we won't do this though)

