Machine Learning: Chenhao Tan

University of Colorado Boulder

LECTURE 8

Slides adapted from Jordan Boyd-Graber, Chris Ketelsen

Logistics

- HW2 available on Github, due in 11 days
- First social time at the end of this lecture

Learning objectives

- Understand some standard feature engineering techniques
- Understand probabilistic classification
- Understand logistic regression

Outline

Feature engineering techniques

Probabilistic classification

Logistic regression

- Logistic Regression Example

Outline

Feature engineering techniques

Probabilistic classification

Logistic regression

- Logistic Regression Example

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

Irrelevant Features

$$E[f \mid X] = E[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors?

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors? ☹

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors? ☹
- Perceptron?

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors? ☹
- Perceptron? ☺

Irrelevant Features

$$\mathrm{E}[f \mid X] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors? ☹
- Perceptron? ☺

What about *redundant* features ϕ_j and $\phi_{j'}$ such that $\phi_j \approx \phi_{j'}$?

Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of D, remove it.

Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of D, remove it.

Example: $\phi(x) = [\![$the word *the* occurs in document $x]\!]$

Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of D, remove it.

Example: $\phi(x) = [\![$the word *the* occurs in document $x]\!]$

Generalization: if a feature has variance (in D) **lower** than some threshhold value, remove it.

$$\text{sample\_mean}(\phi; D) = \frac{1}{N}\sum_{n=1}^{N}\phi(x_n) \qquad \text{(call it "}\bar{\phi}\text{")}$$

$$\text{sample\_variance}(\phi; D) = \frac{1}{N-1}\sum_{n=1}^{N}\left(\phi(x_n) - \bar{\phi}\right)^2 \qquad \text{(call it "Var}(\phi)\text{")}$$

Technique: Feature Normalization

Center a feature:

$$\phi(x) \rightarrow \phi(x) - \bar{\phi}$$

(This was a required step for principal components analysis!)

Scale a feature. Two choices:

$$\phi(x) \rightarrow \frac{\phi(x)}{\sqrt{\text{Var}(\phi)}} \qquad \text{"variance scaling"}$$

$$\phi(x) \rightarrow \frac{\phi(x)}{\max_n |\phi(x_n)|} \qquad \text{"absolute scaling"}$$

Technique: Feature Normalization

Center a feature:

$$\phi(x) \rightarrow \phi(x) - \bar{\phi}$$

(This was a required step for principal components analysis!)

Scale a feature. Two choices:

$$\phi(x) \rightarrow \frac{\phi(x)}{\sqrt{\mathrm{Var}(\phi)}} \qquad \text{“variance scaling”}$$

$$\phi(x) \rightarrow \frac{\phi(x)}{\max_n |\phi(x_n)|} \qquad \text{“absolute scaling”}$$

Remember that you’ll need to normalize test data before you test!

Technique: Example Normalization

We have been talking about normalizing columns.

We can also normalize rows. l_2 normalization is commonly used for bag of words.

$$\boldsymbol{x} = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2} = \frac{\boldsymbol{x}}{\sqrt{\sum_j \boldsymbol{x}[j]^2}}$$

Techniques: Creating New Features

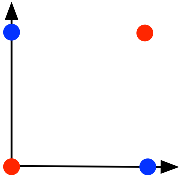
1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

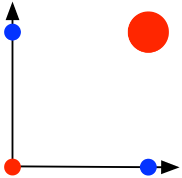


The classic "xor" problem: these points are *not* linearly separable.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

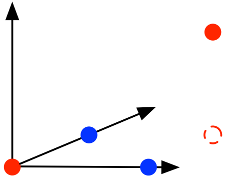


Define $\boldsymbol{x}[3] = \boldsymbol{x}[1] \wedge \boldsymbol{x}[2]$.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

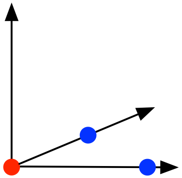


Rotating the view.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

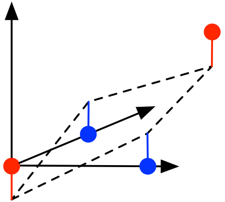
$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$



Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$



$$2 \cdot \boldsymbol{x}[1] + 2 \cdot \boldsymbol{x}[2] - 4 \cdot \boldsymbol{x}[3] - 1 = 0$$

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

Generalization: take the *product* of two features.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

   Generalization: take the *product* of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we'd like.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

   Generalization: take the *product* of two features.
2. Even more generally, we can create conjunctions (or products) using as many features as we'd like.
   This is one view of what decision trees are doing!
   - Every leaf's path (from root) is a conjunction feature.
   - Why not build decision trees, extract the features and toss them into the perceptron?

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

   Generalization: take the *product* of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we'd like.
   This is one view of what decision trees are doing!
   - Every leaf's path (from root) is a conjunction feature.
   - Why not build decision trees, extract the features and toss them into the perceptron?
3. Transformations on features can be useful. For example,

$$\phi(x) \rightarrow \text{sign}(\phi(x)) \cdot \log(1 + |\phi(x)|)$$

Techniques: Creating New Features

Remember that adding features does not always bring benefits.

You could be just bring irrelevant, redundant, or features that make linearly separable datasets not linearly separable.

A more realistic but easy example

Given the following data about the locations of two cities, predict whether it is possible to drive between these two cities.

City 1 lat.	City 1 long.	City 2 lat.	City 2 long.	drivable
123.24	46.71	121.33	47.34	Yes
123.24	56.91	121.33	55.23	Yes
123.24	46.71	121.33	55.34	No
123.24	46.71	130.99	47.34	No

Feature engineering

Features represent the food of machine learning.

Feature engineering

Features represent the food of machine learning.

Garbage in, garbage out.

Feature engineering

Features represent the food of machine learning.

Garbage in, garbage out.

- Pruning
- Normalization
- Creating new features

Feature engineering

Features represent the food of machine learning.

Garbage in, garbage out.

- Pruning
- Normalization
- Creating new features

In practice, feature engineering requires a deep understanding of the problem.

Outline

Recap

K-nearest neighbor

- Find $\mathcal{N}_K(\boldsymbol{x})$: the set of K training examples nearest to $\boldsymbol{x}$
- Predict $\hat{y}$ to be majority label in $\mathcal{N}_K(\boldsymbol{x})$
- Admits a probabilistic interpretation of class given data: $p(y = c \mid \boldsymbol{x})$

Recap

K-nearest neighbor

- Find $\mathcal{N}_K(\boldsymbol{x})$: the set of K training examples nearest to $\boldsymbol{x}$
- Predict $\hat{y}$ to be majority label in $\mathcal{N}_K(\boldsymbol{x})$
- Admits a probabilistic interpretation of class given data: $p(y = c \mid \boldsymbol{x})$

Perceptron

- Learn weights $\boldsymbol{w}$ and b via the perceptron algorithm
- Predict $\hat{y}$ via $\hat{y} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
- Has no probabilistic interpretation

Probabilistic Models

- hypothesis function $h : X \to Y$.

Probabilistic Models

- hypothesis function $h : X \to Y$.
  In this special case, we define h based on estimating a probabilistic model $P(X, Y)$.

Probabilistic Classification

Input: $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ training examples

$$y_i \in \{c_1, c_2, \ldots, c_J\}$$

Goal: $h : X \to Y$
For each class c_j, estimate

$$P(y = c_j \mid \boldsymbol{x}, S_{\text{train}})$$

Assign to $\boldsymbol{x}$ the class with the highest probability

$$\hat{y} = h(\boldsymbol{x}) = \arg\max_c P(y = c \mid \boldsymbol{x}, S_{\text{train}})$$

Outline

Feature engineering techniques

Probabilistic classification

Logistic regression

Logistic Regression Example

What are we talking about?

- Probabilistic classification: $p(y|x)$
- Classification uses: ad placement, spam detection
- Building block of other machine learning methods

Logistic Regression: Definition

- Weight vector β_i
- Observations X_i
- "Bias" β_0 (like intercept in linear regression)

$$P(Y = 0|X) = \frac{1}{1 + \exp\left[\beta_0 + \sum_i \beta_i X_i\right]} \tag{1}$$

$$P(Y = 1|X) = \frac{\exp\left[\beta_0 + \sum_i \beta_i X_i\right]}{1 + \exp\left[\beta_0 + \sum_i \beta_i X_i\right]} \tag{2}$$

Logistic Regression: Definition

- Weight vector β_i
- Observations X_i
- "Bias" β_0 (like intercept in linear regression)

$$P(Y=0|X) = \frac{1}{1+\exp\left[\beta_0 + \sum_i \beta_i X_i\right]} \tag{1}$$

$$P(Y=1|X) = \frac{\exp\left[\beta_0 + \sum_i \beta_i X_i\right]}{1+\exp\left[\beta_0 + \sum_i \beta_i X_i\right]} \tag{2}$$

$$\beta_0 + \sum_i \beta_i X_i = \log \frac{P(Y=1|X)}{P(Y=0|X)}$$

Logistic Regression: Definition

- Weight vector β_i
- Observations X_i
- "Bias" β_0 (like intercept in linear regression)

$$P(Y = 0|X) = \frac{1}{1 + \exp\left[\beta_0 + \sum_i \beta_i X_i\right]} \tag{1}$$

$$P(Y = 1|X) = \frac{\exp\left[\beta_0 + \sum_i \beta_i X_i\right]}{1 + \exp\left[\beta_0 + \sum_i \beta_i X_i\right]} \tag{2}$$

$$\beta_0 + \sum_i \beta_i X_i = \log \frac{P(Y = 1|X)}{P(Y = 0|X)}$$

What is the decision boundary?

Logistic Regression: Definition

- Weight vector β_i
- Observations X_i
- For shorthand, we'll say that

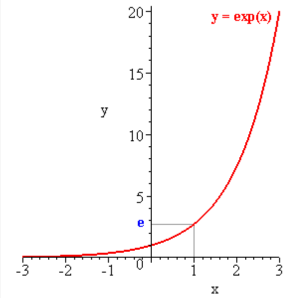
$$P(Y=1|X) = \sigma((\beta_0 + \sum_i \beta_i X_i)) \tag{3}$$

$$P(Y=0|X) = 1 - \sigma((\beta_0 + \sum_i \beta_i X_i)) \tag{4}$$

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

What's this "exp" doing?

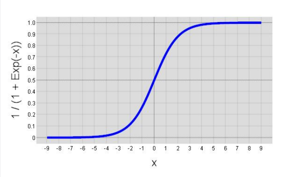
Exponential function



- $\exp[x]$ is shorthand for e^x
- e is a special number, about 2.71828
  - e^x is the limit of compound interest formula as compounds become infinitely small
  - It's the function whose derivative is itself
- The "logistic" function is $\sigma(z) = \frac{1}{1+e^{-z}}$
- Looks like an "S"
- Always between 0 and 1.

What's this "exp" doing?

Logistic function



- $\exp[x]$ is shorthand for e^x
- e is a special number, about 2.71828
  - e^x is the limit of compound interest formula as compounds become infinitely small
  - It's the function whose derivative is itself
- The "logistic" function is $\sigma(z) = \frac{1}{1+e^{-z}}$
- Looks like an "S"
- Always between 0 and 1.
  - Allows us to model probabilities
  - Different from **linear** regression

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- What does $Y = 1$ mean?

Example 1: Empty Document?

$X = \{\}$

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- $Y = 1$: spam

Example 1: Empty Document?

$X = \{\}$

- $P(Y = 0) = \frac{1}{1+\exp[0.1]} =$
- $P(Y = 1) = \frac{\exp[0.1]}{1+\exp[0.1]} =$

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- $Y = 1$: spam

Example 1: Empty Document?

$X = \{\}$

- $P(Y = 0) = \frac{1}{1+\exp[0.1]} = 0.48$
- $P(Y = 1) = \frac{\exp[0.1]}{1+\exp[0.1]} = 0.52$
- Bias β_0 encodes the prior probability of a class

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
“viagra”	β_1	2.0
“mother”	β_2	-1.0
“work”	β_3	-0.5
“nigeria”	β_4	3.0

- $Y = 1$: spam

Example 2

$X = \{\text{Mother}, \text{Nigeria}\}$

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- $Y = 1$: spam

Example 2

$X = \{\text{Mother}, \text{Nigeria}\}$

- $P(Y=0) = \frac{1}{1+\exp\left[0.1-1.0+3.0\right]} =$
- $P(Y=1) = \frac{\exp\left[0.1-1.0+3.0\right]}{1+\exp\left[0.1-1.0+3.0\right]} =$
- Include bias, and sum the other weights

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- $Y = 1$: spam

Example 2

$X = \{\text{Mother}, \text{Nigeria}\}$

- $P(Y=0) = \frac{1}{1+\exp\left[0.1-1.0+3.0\right]} = 0.11$
- $P(Y=1) = \frac{\exp\left[0.1-1.0+3.0\right]}{1+\exp\left[0.1-1.0+3.0\right]} = 0.89$
- Include bias, and sum the other weights

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
“viagra”	β_1	2.0
“mother”	β_2	-1.0
“work”	β_3	-0.5
“nigeria”	β_4	3.0

- $Y = 1$: spam

Example 3

$X = \{\text{Mother}, \text{Work}, \text{Viagra}, \text{Mother}\}$

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- $Y = 1$: spam

Example 3

$X = \{\text{Mother}, \text{Work}, \text{Viagra}, \text{Mother}\}$

- $P(Y = 0) = \frac{1}{1+\exp[0.1-1.0-0.5+2.0-1.0]} =$
- $P(Y = 1) = \frac{\exp[0.1-1.0-0.5+2.0-1.0]}{1+\exp[0.1-1.0-0.5+2.0-1.0]} =$
- Multiply feature presence by weight

Logistic Regression Example

feature	coefficient	weight
bias	β_0	0.1
"viagra"	β_1	2.0
"mother"	β_2	-1.0
"work"	β_3	-0.5
"nigeria"	β_4	3.0

- $Y = 1$: spam

Example 3

$X = \{$Mother, Work, Viagra, Mother$\}$

- $P(Y=0) = \frac{1}{1+\exp\left[0.1-1.0-0.5+2.0-1.0\right]} = 0.60$
- $P(Y=1) = \frac{\exp\left[0.1-1.0-0.5+2.0-1.0\right]}{1+\exp\left[0.1-1.0-0.5+2.0-1.0\right]} = 0.40$
- Multiply feature presence by weight

How is Logistic Regression Used?

- Given a set of weights $\vec{\beta}$, we know how to compute the conditional likelihood $P(y|\beta, x)$
- Find the set of weights $\vec{\beta}$ that maximize the conditional likelihood on training data (next week)
- **Intuition**: higher weights mean that this feature implies that this feature is a good feature for the positive class