

Text Classification  
+  
Machine Learning Review

CS 287

# Contents

## Text Classification

### Preliminaries: Machine Learning for NLP

- Features and Preprocessing

- Output

### Classification

- Linear Models

- Linear Model 1: Naive Bayes

- Linear Model 2: Multiclass Logistic Regression

- Linear Model 3: Multiclass Hinge-Loss

*Earn a Degree based on your Life Experience*

*Obtain a Bachelor's, Master's, MBA, or PhD based on your present knowledge and life experience.*

*No required tests, classes, or books. Confidentiality assured.*

*Join our fully recognized Degree Program.*

*Are you a truly qualified professional in your field but lack the appropriate, recognized documentation to achieve your goals?*

*Or are you venturing into a new field and need a boost to get your foot in the door so you can prove your capabilities?*

*Call us for information that can change your life and help you to achieve your goals!!!*

**CALL NOW TO RECEIVE YOUR DIPLOMA WITHIN 30  
DAYS**

# Sentiment

## Good Sentences

- ▶ A thoughtful, provocative, insistently humanizing film.
- ▶ Occasionally melodramatic, it's also extremely effective.
- ▶ Guaranteed to move anyone who ever shook, rattled, or rolled.

## Bad Sentences

- ▶ A sentimental mess that never rings true.
- ▶ This 100-minute movie only has about 25 minutes of decent material.
- ▶ Here, common sense flies out the window, along with the hail of bullets, none of which ever seem to hit Sascha.

# Multiclass Sentiment

- ▶ ★★★

I visited The Abbey on several occasions on a visit to Cambridge and found it to be a solid, reliable and friendly place for a meal.

- ▶ ★★

However, the food leaves something to be desired. A very obvious menu and average execution

- ▶ ★★★★★

Fun, friendly neighborhood bar. Good drinks, good food, not too pricey. Great atmosphere!

# Text Categorization

- ▶ Straightforward setup.
- ▶ Lots of practical applications:
  - ▶ Spam Filtering
  - ▶ Sentiment
  - ▶ Text Categorization
  - ▶ e-discovery
  - ▶ Twitter Mining
  - ▶ Author Identification
  - ▶ ...
- ▶ Introduces machine learning notation.

However, a relatively solved problem these days.

# Contents

## Text Classification

### Preliminaries: Machine Learning for NLP

- Features and Preprocessing

- Output

## Classification

- Linear Models

- Linear Model 1: Naive Bayes

- Linear Model 2: Multiclass Logistic Regression

- Linear Model 3: Multiclass Hinge-Loss

## Preliminary Notation

- ▶ **b, m**; bold letters for vectors.
- ▶ **B, M**; bold capital letters for matrices.
- ▶  $\mathcal{B}, \mathcal{M}$ ; script-case for sets.
- ▶  $B, M$ ; capital letters for random variables.
- ▶  $b_i, x_i$ ; lower case for scalars or indexing into vectors.
- ▶  $\delta(i)$ ; one-hot vector at position  $i$

$$\delta(2) = [0; 1; 0; \dots]$$

- ▶  $\mathbf{1}(x = y)$ ; indicator 1 if  $x = y$ , o.w. 0



# Text Classification

1. Extract pertinent information from the sentence.
2. Use this to construct an input representation.
3. Classify this vector into an output class.

## Input Representation:

- ▶ Conversion from text into a mathematical representation?
- ▶ Main focus of this class, representation of language
- ▶ Point in coming lectures: *sparse* vs. *dense* representations

# Text Classification

1. Extract pertinent information from the sentence.
2. Use this to construct an input representation.
3. Classify this vector into an output class.

## **Input Representation:**

- ▶ Conversion from text into a mathematical representation?
- ▶ Main focus of this class, representation of language
- ▶ Point in coming lectures: *sparse* vs. *dense* representations

# Sparse Features

- ▶  $\mathcal{F}$ ; a discrete set of features types.
- ▶  $f_1 \in \mathcal{F}, \dots, f_k \in \mathcal{F}$ ; active features for input.

For a given sentence, let  $f_1, \dots, f_k$  be the relevant features.

Typically  $k \ll |\mathcal{F}|$ .

- ▶ Sparse representation of the input defined as,

$$\mathbf{x} = \sum_{i=1}^k \delta(f_i)$$

- ▶  $\mathbf{x} \in \mathbb{R}^{1 \times d_{\text{in}}}$ ; input representation

# Features 1: Sparse Bag-of-Words Features

Representation is counts of input words,

- ▶  $\mathcal{F}$ ; the vocabulary of the language.
- ▶  $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

A sentimental mess

$$\mathbf{x} = v(\text{word:A}) + v(\text{word:sentimental}) + v(\text{word:mess})$$

$$\mathbf{x}^\top = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \text{word:A} \\ \vdots \\ \text{word:mess} \\ \text{word:sentimental} \end{matrix}$$

## Features 2: Sparse Word Properties

Representation can use specific aspects of text.

- ▶  $\mathcal{F}$ ; Spelling, all-capitals, trigger words, etc.
- ▶  $\mathbf{x} = \sum_i \delta(f_i)$

Example: Spam Email

Your diploma puts a UUNIVERSITY JOB PLACEMENT COUNSELOR  
at your disposal.

$$\mathbf{x} = v(\text{misspelling}) + v(\text{allcapital}) + v(\text{trigger:diploma}) + \dots$$

$$\mathbf{x}^\top = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \text{misspelling} \\ \vdots \\ \text{capital} \\ \text{word:diploma} \end{matrix}$$

# Text Classification: Output Representation

1. Extract pertinent information from the sentence.
2. Use this to construct an input representation.
3. Classify this vector into an output class.

## **Output Representation:**

- ▶ How do encode the output classes?
- ▶ We will use a one-hot output encoding.
- ▶ In future lectures, efficiency of output encoding.

# Output Class Notation

- ▶  $\mathcal{C} = \{1, \dots, d_{\text{out}}\}$ ; possible output classes
- ▶  $c \in \mathcal{C}$ ; always one true output class
- ▶  $\mathbf{y} = \delta(c) \in \mathbb{R}^{1 \times d_{\text{in}}}$ ; true one-hot output representation

# Output Form: Binary Classification

Examples: spam/not-spam, good review/bad review, relevant/irrelevant document, many others.

- ▶  $d_{\text{out}} = 2$ ; two possible classes
- ▶ In our notation,

$$\begin{array}{ll} \text{bad } c = 1 & \mathbf{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \text{ vs.} \\ \text{good } c = 2 & \mathbf{y} = \begin{bmatrix} 0 & 1 \end{bmatrix} \end{array}$$

- ▶ Can also use a single output *sign* representation with  $d_{\text{out}} = 1$



## Output Form: Multiclass Classification

Examples: Yelp stars, etc.

- ▶  $d_{\text{out}} = 5$ ; for examples
- ▶ In our notation, one star, two star...

$$\begin{aligned} \star \ c = 1 \quad \mathbf{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ vs.} \\ \star\star \ c = 2 \quad \mathbf{y} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \dots \end{aligned}$$

Examples: Word Prediction (Unit 3)

- ▶  $d_{\text{out}} > 100,000$ ;
- ▶ In our notation,  $\mathcal{C}$  is vocabulary and each  $c$  is a word.

$$\begin{aligned} \textit{the} \ c = 1 \quad \mathbf{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \text{ vs.} \\ \textit{dog} \ c = 2 \quad \mathbf{y} &= \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \end{bmatrix} \dots \end{aligned}$$

# Evaluation

- ▶ Consider evaluating accuracy on outputs  $\mathbf{y}_1, \dots, \mathbf{y}_n$ .
- ▶ Given a decisions  $\hat{c}_1 \dots \hat{c}_n$  we measure accuracy as,

$$\sum_{i=1}^n \frac{\mathbf{1}(\delta(\hat{c}_i) = \mathbf{y}_i)}{n}$$

- ▶ Simplest of several different metrics we will explore in the class.

# Contents

## Text Classification

### Preliminaries: Machine Learning for NLP

- Features and Preprocessing

- Output

## Classification

- Linear Models

- Linear Model 1: Naive Bayes

- Linear Model 2: Multiclass Logistic Regression

- Linear Model 3: Multiclass Hinge-Loss

# Supervised Machine Learning

Let,

- ▶  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$ ; training data
- ▶  $\mathbf{x}_i \in \mathbb{R}^{1 \times d_{\text{in}}}$ ; input representations
- ▶  $\mathbf{y}_i \in \mathbb{R}^{1 \times d_{\text{out}}}$ ; gold output representations (one-hot vectors)

Goal: Learn a classifier from input to output classes.

Note:

- ▶  $\mathbf{x}_i$  is an input vector  $x_{i,j}$  is element of the vector, or just  $x_j$  when there is a clear single input .
- ▶ Practically, store design matrix  $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{in}}}$  and output classes.

# Experimental Setup

- ▶ Data is split into three parts training, validation, and test.
- ▶ Experiments are all run on training and validation, test is final output.
- ▶ For assignments, full training and validation data, and only inputs for test.

For very small text classification data sets,

- ▶ Use K-fold cross-validation.
  1. Split into K folds (equal splits).
  2. For each fold, train on other K-1 folds, test on current fold.

# Linear Models for Classification

Linear model,

$$\hat{\mathbf{y}} = \mathbf{x}\mathbf{W} + \mathbf{b}$$

- ▶  $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ ,  $\mathbf{b} \in \mathbb{R}^{1 \times d_{\text{out}}}$ ; model parameters
- ▶ Note  $\hat{\mathbf{y}}$  is **not** one-hot, informally “score” vector.

Class decision,

$$\hat{c} = \arg \max_{i \in \mathcal{C}} \hat{y}_i$$

# Interpreting Linear Models

Parameters give scores to possible outputs,

- ▶  $W_{f,i}$  is the score for sparse feature  $f$  under class  $i$
- ▶  $b_i$  is a prior score for class  $i$
- ▶  $\hat{y}_i$  is the total score for class  $i$
- ▶  $\hat{c}$  is highest scoring class under the linear model.

Example:

- ▶ For single feature score,

$$[\beta_1, \beta_2] = \delta(\text{word:dreadful})\mathbf{W},$$

Expect  $\beta_2 > \beta_1$  (assuming 2 is class *good*).

# Probabilistic Linear Models

Can estimate a linear model probabilistically ,

- ▶ Let output be a random variable  $Y$ , with sample space  $\mathcal{C}$ .
- ▶ Representation be a random vector  $X$ .
- ▶ Interested in estimating parameters  $\theta$  of,

$$P(Y|X; \theta)$$

Informally we use  $p(\mathbf{y} = c|\mathbf{x})$  for  $P(Y = c|X = \mathbf{x})$ .



# Log-Likelihood as Loss

- ▶  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$ ; supervised data
- ▶ Select parameters to maximize likelihood of training data.

$$\mathcal{L}(\theta) = - \sum_{i=1}^n \log p(\mathbf{y}_i | \mathbf{x}_i; \theta)$$

For linear models  $\theta = (\mathbf{W}, \mathbf{b})$

- ▶ Do this by minimizing negative log-likelihood (NLL).

$$\arg \min_{\theta} \mathcal{L}(\theta)$$

# Naive Bayes 1: Probabilistic Factorization

Reminder, Bayes Rule

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{x})}$$

Can be instead written (with  $\propto$  as normalizing factor)

$$p(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$$

For NLL,  $p(\mathbf{x})$  doesn't matter, estimate  $p(\mathbf{x}|\mathbf{y})$  and  $p(\mathbf{y})$ .

For a sparse model, with observed classes we can write as,

$$p(x_{f_1} = 1, \dots, x_{f_k} = 1 | \mathbf{y} = c) p(\mathbf{y} = c)$$

## Naive Bayes 2: Independence Assumption

$$\begin{aligned} p(x_{f_1} = 1, \dots, x_{f_k} = 1 | \mathbf{y} = c) p(\mathbf{y} = c) &= \\ \prod_{i=1}^k p(x_{f_i} = 1 | x_{f_1} = 1, \dots, x_{f_{i-1}} = 1, \mathbf{y} = c) p(\mathbf{y} = c) &\approx \\ \prod_{i=1}^k p(x_{f_i} | \mathbf{y}) p(\mathbf{y}) \end{aligned}$$

First is by chain-rule, second is by assumption.

# Multinomial Model

Brief aside,

- ▶  $P(S; \theta)$ ; parameterized as a multinomial distribution.
- ▶ Minimizing NLL for multinomial for data has a closed-form.

$$P(S = s; \theta) = \theta_s = \sum_{i=1}^n \frac{\mathbf{1}(s_i = s)}{n}$$

- ▶ Exercise: Derive this by minimizing  $\mathcal{L}$ .

# Multinomial Naive Bayes

- ▶ Both  $p(\mathbf{y})$  and  $p(\mathbf{x}|\mathbf{y})$  are parameterized as multinomials.
- ▶ Fit first as,

$$p(\mathbf{y} = c) = \sum_{i=1}^n \frac{\mathbf{1}(\mathbf{y}_i = c)}{n}$$

- ▶ Fit second using count matrix  $\mathbf{F}$  ,
  - ▶ Let

$$F_{f,c} = \sum_{i=1}^n \mathbf{1}(\mathbf{y}_i = c) \mathbf{1}(x_{i,f} = 1) \text{ for all } c \in \mathcal{C}, f \in \mathcal{F}$$

- ▶ Then,

$$p(x_f = 1 | \mathbf{y} = c) = \frac{F_{f,c}}{\sum_{f' \in \mathcal{F}} F_{f',c}}$$

How does this become a linear classifier?

$$W_{f,c} = \log p(x_f = 1 | \mathbf{y} = c)$$

$$b_c = \log p(\mathbf{y} = c) \text{ for all } c \in \mathcal{Y}$$

# Multinomial Naive Bayes

- ▶ Both  $p(\mathbf{y})$  and  $p(\mathbf{x}|\mathbf{y})$  are parameterized as multinomials.
- ▶ Fit first as,

$$p(\mathbf{y} = c) = \sum_{i=1}^n \frac{\mathbf{1}(\mathbf{y}_i = c)}{n}$$

- ▶ Fit second using count matrix  $\mathbf{F}$  ,
  - ▶ Let

$$F_{f,c} = \sum_{i=1}^n \mathbf{1}(\mathbf{y}_i = c) \mathbf{1}(x_{i,f} = 1) \text{ for all } c \in \mathcal{C}, f \in \mathcal{F}$$

- ▶ Then,

$$p(x_f = 1 | \mathbf{y} = c) = \frac{F_{f,c}}{\sum_{f' \in \mathcal{F}} F_{f',c}}$$

How does this become a linear classifier?

$$W_{f,c} = \log p(x_f = 1 | \mathbf{y} = c)$$

$$b_c = \log p(\mathbf{y} = c) \text{ for all } c \in \mathcal{Y}$$

# Multinomial Naive Bayes

- ▶ Both  $p(\mathbf{y})$  and  $p(\mathbf{x}|\mathbf{y})$  are parameterized as multinomials.
- ▶ Fit first as,

$$p(\mathbf{y} = c) = \sum_{i=1}^n \frac{\mathbf{1}(\mathbf{y}_i = c)}{n}$$

- ▶ Fit second using count matrix  $\mathbf{F}$  ,
  - ▶ Let

$$F_{f,c} = \sum_{i=1}^n \mathbf{1}(\mathbf{y}_i = c) \mathbf{1}(x_{i,f} = 1) \text{ for all } c \in \mathcal{C}, f \in \mathcal{F}$$

- ▶ Then,

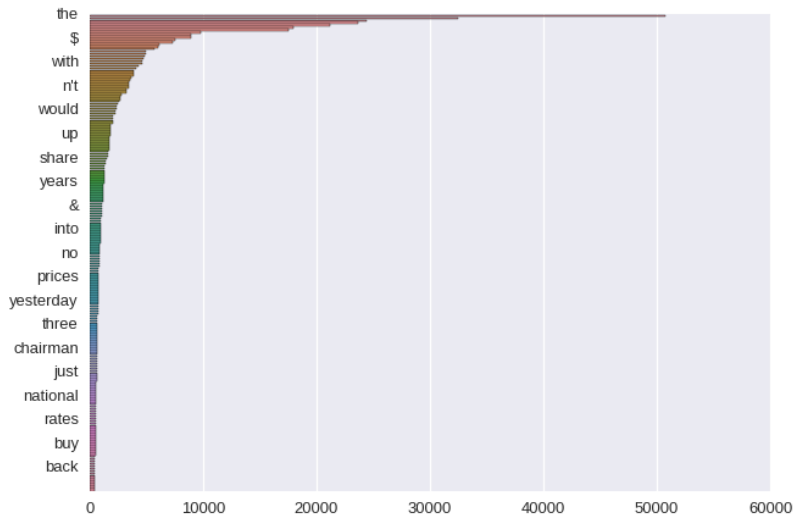
$$p(x_f = 1 | \mathbf{y} = c) = \frac{F_{f,c}}{\sum_{f' \in \mathcal{F}} F_{f',c}}$$

How does this become a linear classifier?

$$W_{f,c} = \log p(x_f = 1 | \mathbf{y} = c)$$

$$b_c = \log p(\mathbf{y} = c) \text{ for all } c \in \mathcal{Y}$$

## Digression: Zipf's Law





# Laplacian Smoothing

Method for handling the long tail of words by distributing mass,

- ▶ Add a value of  $\alpha$  to each element in the sample space before normalization.

$$\theta_s = \frac{\alpha + \sum_{i=1}^n \mathbf{1}(s_i = s)}{\alpha|\mathcal{S}| + n}$$

- ▶ (Similar to Dirichlet prior in a Bayesian interpretation.)

For naive Bayes:

$$\hat{\mathbf{F}} = \alpha + F$$

# Laplacian Smoothing

Method for handling the long tail of words by distributing mass,

- ▶ Add a value of  $\alpha$  to each element in the sample space before normalization.

$$\theta_s = \frac{\alpha + \sum_{i=1}^n \mathbf{1}(s_i = s)}{\alpha|\mathcal{S}| + n}$$

- ▶ (Similar to Dirichlet prior in a Bayesian interpretation.)

For naive Bayes:

$$\hat{\mathbf{F}} = \alpha + F$$

## Naive Bayes In Practice

- ▶ Very fast to train
- ▶ Relatively interpretable.
- ▶ Performs quite well on small datasets ?

Method	RT-s	MPQA	CR	Subj.
MNB-uni	77.9	85.3	79.8	<b>92.6</b>
MNB-bi	<b>79.0</b>	<b>86.3</b>	80.0	<b><u>93.6</u></b>
SVM-uni	76.2	86.1	79.0	90.8
SVM-bi	77.7	<b><u>86.7</u></b>	80.8	91.7
NBSVM-uni	<b>78.1</b>	85.3	80.5	92.4
NBSVM-bi	<b><u>79.4</u></b>	<b>86.3</b>	<b><u>81.8</u></b>	<b>93.2</b>
RAE	76.8	85.7	–	–
RAE-pretrain	77.7	<b>86.4</b>	–	–
Voting-w/Rev.	63.1	81.7	74.2	–

(RT-S [movie review], CR [customer reports], MPQA [opinion polarity], SUBJ [subjectivity])

# Multiclass Logistic Regression

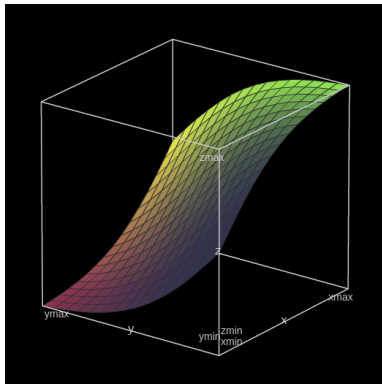
Alternative parametrization of probabilistic model.

Use a softmax to force a distribution,

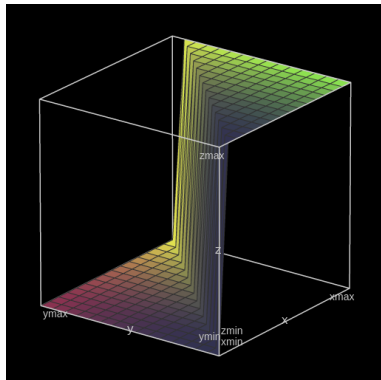
$$\text{softmax}(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\sum_{c \in \mathcal{C}} \exp(z_c)}$$

- ▶ Exercise: Confirm always gives a distribution.
- ▶ Denominator known as *partition* function (we'll see many times).

# Why is it called the softmax?



$$\text{softmax}([x \ y]) = \frac{\exp(x)}{\exp(x) + \exp(y)}$$



$$\text{arg max}([x \ y]) = \mathbf{1}(x > y)$$

## Multiclass logistic regression

$$\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$$

$$p(\mathbf{y} = c | \mathbf{x}; \theta) = \hat{y} = \text{softmax}(\mathbf{z}) = \frac{\exp(z_c)}{\sum_{c'} \exp(z_{c'})}$$

- $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ ,  $\mathbf{b} \in \mathbb{R}^{1 \times d_{\text{out}}}$ ; model parameters

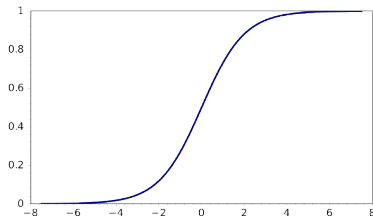
## Special Case: Logistic Regression

For binary classification:

$$\begin{aligned}\text{softmax}([z_1 \ z_2]) &= \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} \\ &= \frac{1}{1 + \exp(-(z_1 - z_2))} = \sigma(z_1 - z_2)\end{aligned}$$

Logistic sigmoid function:

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



# A Model with Many Names

- ▶ Multinomial Logistic Regression
- ▶ Log-Linear Model (particularly in NLP)
- ▶ Softmax Regression
- ▶ Max-Entropy (MaxEnt)



# Fitting Parameters

Recall probabilistic objective is:

$$\mathcal{L}(\theta) = - \sum_{i=1}^n \log p(\mathbf{y}_i | \mathbf{x}_i; \theta) = \sum_{i=1}^n L_{\text{cross-entropy}}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$

4 And the distribution is parameterized as a softmax,

$$\begin{aligned} L_{\text{cross-entropy}}(\mathbf{y}, \hat{\mathbf{y}}) &= -\log p(\mathbf{y} = c | \mathbf{x}; \theta) \\ &= \log \text{softmax}(\mathbf{z})_c \\ &= \hat{z}_c - \log \sum_{c' \in \mathcal{C}} \exp(z_{c'}) \end{aligned}$$

However, this is much harder to minimize, **no closed form**.

# Symbolic Gradients

- Partials of  $L(y, \hat{y})$

$$\frac{\partial L(y, \hat{y})}{\partial \hat{y}_j} = \frac{\mathbf{1}(y_j = 1)}{\hat{y}_j}$$

- Partials of  $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$

$$\frac{\partial \hat{y}_j}{\partial z_i} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i\hat{y}_j & i \neq j \end{cases}$$

- Partials of  $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

$$\frac{\partial z_i}{\partial b_{i'}} = \mathbf{1}(i = i') \quad \frac{\partial z_i}{\partial W_{f,i'}} = \mathbf{1}(i = i')$$

Homework: Compute these for yourself.

# Symbolic Gradients

- Partials of  $L(y, \hat{y})$

$$\frac{\partial L(y, \hat{y})}{\partial \hat{y}_j} = \frac{\mathbf{1}(y_j = 1)}{\hat{y}_j}$$

- Partials of  $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$

$$\frac{\partial \hat{y}_j}{\partial z_i} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i\hat{y}_j & i \neq j \end{cases}$$

- Partials of  $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

$$\frac{\partial z_i}{\partial b_{i'}} = \mathbf{1}(i = i') \quad \frac{\partial z_i}{\partial W_{f,i'}} = \mathbf{1}(i = i')$$

Homework: Compute these for yourself.

# Symbolic Gradients

- Partials of  $L(y, \hat{y})$

$$\frac{\partial L(y, \hat{y})}{\partial \hat{y}_j} = \frac{\mathbf{1}(y_j = 1)}{\hat{y}_j}$$

- Partials of  $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$

$$\frac{\partial \hat{y}_j}{\partial z_i} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i\hat{y}_j & i \neq j \end{cases}$$

- Partials of  $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

$$\frac{\partial z_i}{\partial b_{i'}} = \mathbf{1}(i = i') \quad \frac{\partial z_i}{\partial W_{f,i'}} = \mathbf{1}(i = i')$$

Homework: Compute these for yourself.

## Review: Chain Rule

Assume we have a function and a loss:

$$f : \mathbb{R}^m \rightarrow \mathbb{R}^n \quad L : \mathbb{R}^n \rightarrow \mathbb{R}$$

Then

$$\frac{\partial L(f(\mathbf{x}))}{\partial x_i} = \sum_{j=1}^n \frac{\partial f(\mathbf{x})_j}{\partial x_i} \frac{\partial L(f(\mathbf{x}))}{\partial f(\mathbf{x})_j}$$

For Softmax regression:

$$\frac{\partial L(y, \hat{y})}{\partial z_i} = \sum_j \frac{\partial \hat{y}_j}{\partial z_i} \frac{\mathbf{1}(y_j = 1)}{\hat{y}_j} = \begin{cases} 1 - \hat{y}_i & y_i = 1 \\ -\hat{y}_j & \text{ow.} \end{cases}$$

## Review: Chain Rule

Assume we have a function and a loss:

$$f : \mathbb{R}^m \rightarrow \mathbb{R}^n \quad L : \mathbb{R}^n \rightarrow \mathbb{R}$$

Then

$$\frac{\partial L(f(\mathbf{x}))}{\partial x_i} = \sum_{j=1}^n \frac{\partial f(\mathbf{x})_j}{\partial x_i} \frac{\partial L(f(\mathbf{x}))}{\partial f(\mathbf{x})_j}$$

For Softmax regression:

$$\frac{\partial L(y, \hat{y})}{\partial z_i} = \sum_j \frac{\partial \hat{y}_j}{\partial z_i} \frac{\mathbf{1}(y_j = 1)}{\hat{y}_j} = \begin{cases} 1 - \hat{y}_i & y_i = 1 \\ -\hat{y}_j & \text{ow.} \end{cases}$$

# Minimizing Gradients in Practice

Consider one example  $(\mathbf{x}, \mathbf{y})$ , we compute forward and then backward,

1. Compute scores  $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$
2. Compute softmax of scores,  $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$
3. Compute loss of scores,  $L(\mathbf{y}, \hat{\mathbf{y}})$
4. Compute gradient  $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j}$ .
5. Compute gradient  $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial z_i}$ .
6. Compute gradient of  $\mathbf{b}$  for all  $i' \in \mathcal{C}$  and  $\mathbf{W}$  for all  $i' \in \mathcal{C}, f \in \mathcal{F}$ ,

$$\frac{\partial L}{\partial b'_i} = \frac{\partial L}{\partial z'_i} \quad \frac{\partial L}{\partial W_{f,i'}} = \frac{\partial L}{\partial z'_i}$$

# Minimizing Gradients in Practice

Consider one example  $(\mathbf{x}, \mathbf{y})$ , we compute forward and then backward,

1. Compute scores  $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$
2. Compute softmax of scores,  $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$
3. Compute loss of scores,  $L(\mathbf{y}, \hat{\mathbf{y}})$
4. Compute gradient  $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j}$ .
5. Compute gradient  $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial z_i}$ .
6. Compute gradient of  $\mathbf{b}$  for all  $i' \in \mathcal{C}$  and  $\mathbf{W}$  for all  $i' \in \mathcal{C}, f \in \mathcal{F}$ ,

$$\frac{\partial L}{\partial b'_i} = \frac{\partial L}{\partial z'_i} \quad \frac{\partial L}{\partial W_{f,i'}} = \frac{\partial L}{\partial z'_i}$$



# Gradient-Based Optimization: SGD

**procedure** SGD

**while** training criterion is not met **do**

        Sample a training example  $\mathbf{x}_i, \mathbf{y}_i$

        Compute the loss  $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$

        Compute gradients  $\hat{\mathbf{g}}$  of  $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$  with respect to  $\theta$

$\theta \leftarrow \theta + \eta_k \hat{\mathbf{g}}$

**end while**

**return**  $\theta$

**end procedure**

## Gradient-Based Optimization: Minibatch SGD

**while** training criterion is not met **do**

Sample a minibatch of  $m$  examples  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)$

$\hat{\mathbf{g}} \leftarrow 0$

**for**  $i = 1$  to  $m$  **do**

Compute the loss  $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$

Compute gradients  $\mathbf{g}'$  of  $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$  with respect to  $\theta$

$\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} + \frac{1}{m} \mathbf{g}'$

**end for**

$\theta \leftarrow \theta + \eta_k \hat{\mathbf{g}}$

**end while**

**return**  $\theta$

## Softmax Notes: Regularization

$$\mathcal{L}(\theta) = - \sum_{i=1}^n L(\hat{\mathbf{y}}, \mathbf{y}) + ||\theta||_2^2$$

## Softmax Notes: Calculating Log-Sum-Exp

- ▶ Calculating  $\log \sum_{c' \in \mathcal{C}} \exp(\hat{y}_{c'})$  directly numerical issues.
- ▶ Instead  $\log \sum_{c' \in \mathcal{C}} \exp(\hat{y}_{c'} - M) + M$  where  $M = \max_{c' \in \mathcal{C}} \hat{y}_{c'}$

# Pros and Cons of Logistic Regression

- ▶ Less strong independence assumption.
- ▶ Can be very effective with good features.
- ▶ Still yields a probability distribution.
- ▶ Fitting parameters is more difficult.

Similar models make will be the main focus of this class.

## Other Loss Functions

What if we just try to directly find **W** and **b**?

$$\hat{\mathbf{y}} = \mathbf{x}\mathbf{W} + \mathbf{b}$$

- ▶ No longer a probabilistic interpretation.
- ▶ Just try to find parameters that fit training data.

# Hinge Loss

$$\mathcal{L}(\theta) = \sum_{i=1}^n L_{hinge}(\hat{\mathbf{y}}, \mathbf{y})$$

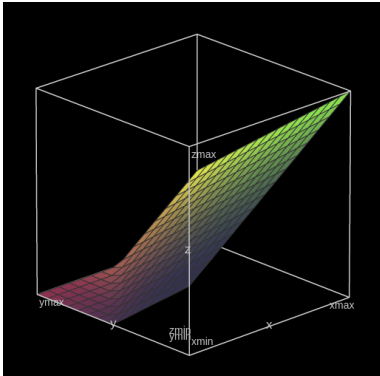
$$L(\hat{\mathbf{y}}, \mathbf{y}) = \max\{0, 1 - (\hat{y}_c + \hat{y}_{c'})\}$$

Where

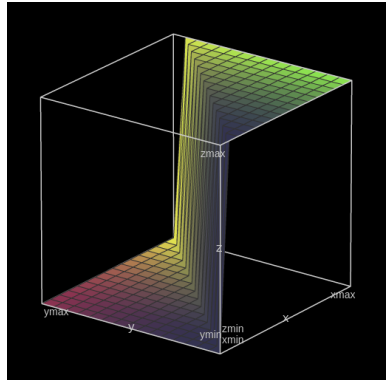
- ▶ Let  $c$  be defined as gold class  $y_{i,c} = 1$
- ▶ Let  $c'$  be defined as the highest scoring non-gold class

$$c' = \arg \max_{i \in \mathcal{C} \setminus \{c\}} \hat{y}_i$$

# Hinge Loss



$$\text{hinge}(\hat{\mathbf{y}}) = \mathbf{1}(\max\{0, 1 - (y - x)\})$$



$$\arg \max([x \ y]) = \mathbf{1}(x > y)$$



# Symbolic Gradients

- ▶ Let  $c$  be defined as gold class  $y_{i,c} = 1$
- ▶ Let  $c'$  be defined as the highest scoring non-gold class

$$c' = \arg \max_{i \in \mathcal{C} \setminus \{c\}} \hat{y}_i$$

Much simpler than logistic regression.

- ▶ Partial of  $L(y, \hat{y})$

$$\frac{\partial L(y, \hat{y})}{\partial \hat{y}_j} = \mathbf{1}(j = c) - \mathbf{1}(j = c')$$

## Notes: Hinge Loss: Regularization

- ▶ Many different names,
  - ▶ Margin Classifier
  - ▶ Multiclass Hinge
  - ▶ Linear SVM
- ▶ Important to use regularization.

$$\mathcal{L}(\theta) = - \sum_{i=1}^n L(\hat{\mathbf{y}}, \mathbf{y}) + \|\theta\|_2^2$$

- ▶ Can be much more efficient to train than LR. (No partition).

## Results: Longer Reviews

<b>Our results</b>	RT-2k	IMDB	Subj.
MNB-uni	83.45	83.55	<b>92.58</b>
MNB-bi	85.85	86.59	<b><u>93.56</u></b>
SVM-uni	86.25	86.95	90.84
SVM-bi	87.40	<b>89.16</b>	91.74
NBSVM-uni	87.80	88.29	92.40
NBSVM-bi	<b>89.45</b>	<b><u>91.22</u></b>	<b>93.18</b>
BoW (bnc)	85.45	87.8	87.77
BoW (b $\Delta t'$ c)	85.8	88.23	85.65
LDA	66.7	67.42	66.65
Full+BoW	87.85	88.33	88.45
Full+Unlab'd+BoW	<b>88.9</b>	88.89	88.13

IMDB (longer movie review), Subj (longer subjectivity)

- NBSVM is hinge-loss interpolated with Naive Bayes.