Natural Language Processing (CSE 517 & 447): Multinomial Logistic Regression

Noah Smith

© 2022

University of Washington
nasmith@cs.washington.edu

Winter 2022

Readings: Eisenstein (2019) 2 and Appendix B

Motivation

- Dominant perspective in NLP in the 1990s–today: supervised machine learning
  - This lecture's model is a direct ancestor of today's popular methods.
- Engineering approach: feature design
- Relevance today: interpretable and efficient classification

Classification in NLP

We approach many problems in NLP by treating them as problems of classification.

- Input might be a document, a paragraph, a sentence, a word
- Output is a label from a finite set of classes or labels, denoted $\mathcal{L}$, defined by your application or theory

Notation: $\text{classify} : \mathcal{V}^* \rightarrow \mathcal{L}$ is a classifier, e.g., the one you build. It is deterministic and typically constructed from data and machine learning.

Text (Document) Classification Examples

- Library-like subjects (e.g., the Dewey decimal system)
- News stories: politics vs. sports vs. business vs. technology ...
- Reviews of films, restaurants, products: postive vs. negative
- Author attributes: identity, political stance, gender, age, ...
- Email, arXiv submissions, etc.: spam vs. not
- What is the reading level of a piece of text?
- How influential will a scientific paper be?
- Will a piece of proposed legislation pass?
- What dialect is a text written in?
- Does the text contain content that will likely offend people?

Notation

$\mathcal{V}$ is the set of words in the language we're working with.

$\boldsymbol{X}$ is a random variable for texts (inputs); in a given instance it takes a value from $\mathcal{V}^*$ (sequences of words).

Y is a random variable for labels (outputs); in a given instance it takes a value from $\mathcal{L}$.

$p(\boldsymbol{X}, Y)$ is the "true" distribution of labeled texts; $p(Y)$ is the distribution of labels. **Normally, we do not know these distributions except by looking at data.**

Evaluating a Classifier

Accuracy:

$$\begin{aligned}
\mathrm{A}(\text{classify}) &= p(\text{classify}(\boldsymbol{X}) = Y) \\
&= \sum_{\boldsymbol{x} \in \mathcal{V}^*, \ell \in \mathcal{L}} p(\boldsymbol{X} = \boldsymbol{x}, Y = \ell) \cdot \begin{cases} 1 & \text{if } \text{classify}(\boldsymbol{x}) = \ell \\ 0 & \text{otherwise} \end{cases} \\
&= \sum_{\boldsymbol{x} \in \mathcal{V}^*, \ell \in \mathcal{L}} p(\boldsymbol{X} = \boldsymbol{x}, Y = \ell) \cdot \mathbf{1}\left\{\text{classify}(\boldsymbol{x}) = \ell\right\}
\end{aligned}$$

where p is the *true* distribution over data. Error is $1 - \mathrm{A}$.

Evaluating a Classifier

Accuracy:

$$\begin{aligned}
\mathrm{A}(\text{classify}) &= p(\text{classify}(\boldsymbol{X}) = Y) \\
&= \sum_{\boldsymbol{x} \in \mathcal{V}^*, \ell \in \mathcal{L}} p(\boldsymbol{X} = \boldsymbol{x}, Y = \ell) \cdot \begin{cases} 1 & \text{if classify}(\boldsymbol{x}) = \ell \\ 0 & \text{otherwise} \end{cases} \\
&= \sum_{\boldsymbol{x} \in \mathcal{V}^*, \ell \in \mathcal{L}} p(\boldsymbol{X} = \boldsymbol{x}, Y = \ell) \cdot \mathbf{1}\left\{\text{classify}(\boldsymbol{x}) = \ell\right\}
\end{aligned}$$

where p is the *true* distribution over data. Error is $1 - \mathrm{A}$.

This is *estimated* using a test dataset $\langle \bar{\boldsymbol{x}}_1, \bar{y}_1 \rangle, \ldots \langle \bar{\boldsymbol{x}}_m, \bar{y}_m \rangle$:

$$\hat{\mathrm{A}}(\text{classify}) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\left\{\text{classify}(\bar{\boldsymbol{x}}_i) = \bar{y}_i\right\}$$

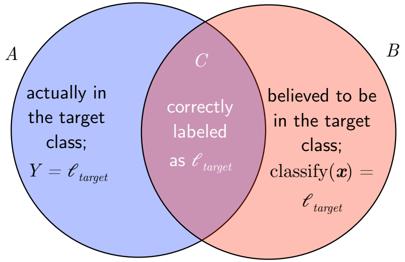
Some Issues with Test-Set Accuracy

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing “not spam.”

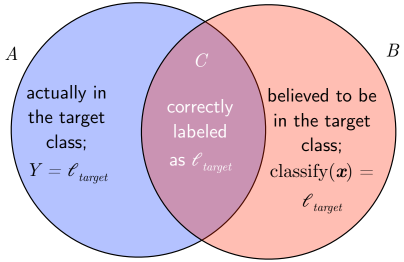
Evaluation in the "Needle in a Haystack" Case

Suppose one label $\ell_{target} \in \mathcal{L}$ is a "target."
Precision and **recall** encode the goals of returning a "pure" set of targeted instances and capturing *all* of them.


A
actually in the target class; $Y = \ell_{target}$
C
correctly labeled as ℓ_{target}
B
believed to be in the target class; $\text{classify}(\boldsymbol{x}) = \ell_{target}$

Evaluation in the "Needle in a Haystack" Case

Suppose one label $\ell_{target} \in \mathcal{L}$ is a "target."
Precision and **recall** encode the goals of returning a "pure" set of targeted instances and capturing *all* of them.



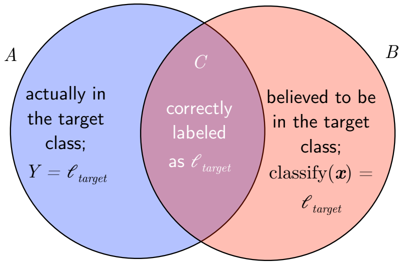
A
actually in the target class; $Y = \ell_{target}$
C
correctly labeled as ℓ_{target}
B
believed to be in the target class; $\text{classify}(\boldsymbol{x}) = \ell_{target}$

$$\hat{\mathrm{P}}(\text{classify}) = \frac{|C|}{|B|} = \frac{|A \cap B|}{|B|}$$

$$\hat{\mathrm{R}}(\text{classify}) = \frac{|C|}{|A|} = \frac{|A \cap B|}{|A|}$$

$$\hat{F}_1(\text{classify}) = 2 \cdot \frac{\hat{\mathrm{P}} \cdot \hat{\mathrm{R}}}{\hat{\mathrm{P}} + \hat{\mathrm{R}}}$$

Another View: Contingency Table


A
actually in the target class; $Y = \ell_{target}$
C
correctly labeled as ℓ_{target}
B
believed to be in the target class; $\text{classify}(\boldsymbol{x}) = \ell_{target}$

	$Y = \ell_{target}$	$Y \neq \ell_{target}$	
$\text{classify}(\boldsymbol{X}) = \ell_{target}$	$\lvert C\rvert$ (true positives)	$\lvert B \setminus C\rvert$ (false positives)	$\lvert B\rvert$
$\text{classify}(\boldsymbol{X}) \neq \ell_{target}$	$\lvert A \setminus C\rvert$ (false negatives)	(true negatives)	
	$\lvert A\rvert$		

Generalization of Precision and Recall

Macroaveraged precision and recall: let each class be the "target" and report the average $\hat{\mathrm{P}}$ and $\hat{\mathrm{R}}$ across all classes.

Microaveraged precision and recall: pool all one-vs.-rest decisions into a single contingency table, calculate $\hat{\mathrm{P}}$ and $\hat{\mathrm{R}}$ from that.

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing “not spam.”
  - Solution: report precision and recall

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing "not spam."
  - Solution: report precision and recall
- Relative importance of classes or cost of error types.

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing "not spam."
  - Solution: report precision and recall
- Relative importance of classes or cost of error types.
  - Solution: report precision and recall for each class, or categorize different error types.

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing "not spam."
  - Solution: report precision and recall
- Relative importance of classes or cost of error types.
  - Solution: report precision and recall for each class, or categorize different error types.
- Variance due to the test data.

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\text{A}} \approx 0.99$ by always guessing "not spam."
  - Solution: report precision and recall
- Relative importance of classes or cost of error types.
  - Solution: report precision and recall for each class, or categorize different error types.
- Variance due to the test data.
  - Solution: repeat entire experiment with shuffled data, multiple times, and report mean and standard deviation.

Some Issues with Test-Set Accuracy

- Class imbalance: if $p(Y = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing "not spam."
  - Solution: report precision and recall
- Relative importance of classes or cost of error types.
  - Solution: report precision and recall for each class, or categorize different error types.
- Variance due to the test data.
  - Solution: repeat entire experiment with shuffled data, multiple times, and report mean and standard deviation.
  - Test data is not representative of real data.

(Some additional topics in the “extras” section at the end of this file: cross-validation and statistical significance.)

Building a Text Classifier: Standard Line of Attack

1. Human experts label some data, or nature provides labeled data.
2. Feed the data to a supervised machine learning algorithm that constructs an automatic classifier $\mathrm{classify} : \mathcal{V}^* \to \mathcal{L}$
3. Apply $\mathrm{classify}$ to as much data as you want!

Note: we assume the texts are segmented into symbols from $\mathcal{V}$, even the new ones.

Features of a Text

Running example:

$\boxed{\boldsymbol{x} = \text{“The vodka was great, but don’t touch the hamburgers.”}}$

A different representation of the text sequences: features.

Features of a Text

Running example:

$\boxed{\boldsymbol{x} = \text{“The vodka was great, but don’t touch the hamburgers.”}}$

A different representation of the text sequences: features.

- Often, these are term (word or word sequence) frequencies.
  E.g., $\phi^{\mathit{freq.}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\mathit{freq.}}_{\text{the}}(\boldsymbol{x}) = 2$, $\phi^{\mathit{freq.}}_{\text{delicious}}(\boldsymbol{x}) = 0$,
  $\phi^{\mathit{freq.}}_{\text{don’t touch}}(\boldsymbol{x}) = 1$.

Features of a Text

Running example:

$\boxed{\boldsymbol{x} = \text{“The vodka was great, but don’t touch the hamburgers.”}}$

A different representation of the text sequences: features.

- Often, these are term (word or word sequence) frequencies. E.g., $\phi^{\mathit{freq.}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\mathit{freq.}}_{\text{the}}(\boldsymbol{x}) = 2$, $\phi^{\mathit{freq.}}_{\text{delicious}}(\boldsymbol{x}) = 0$, $\phi^{\mathit{freq.}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.

“Bag of words” model: one based on word frequency features alone.

Features of a Text

Running example:

$\boxed{\boldsymbol{x} = \text{“The vodka was great, but don’t touch the hamburgers.”}}$

A different representation of the text sequences: features.

- Often, these are term (word or word sequence) frequencies.
  E.g., $\phi^{\textit{freq.}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\textit{freq.}}_{\text{the}}(\boldsymbol{x}) = 2$, $\phi^{\textit{freq.}}_{\text{delicious}}(\boldsymbol{x}) = 0$,
  $\phi^{\textit{freq.}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.

Features of a Text

Running example:

$\boxed{\boldsymbol{x} = \text{“The vodka was great, but don't touch the hamburgers.”}}$

A different representation of the text sequences: features.

- Often, these are term (word or word sequence) frequencies. E.g., $\phi^{\textit{freq.}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\textit{freq.}}_{\text{the}}(\boldsymbol{x}) = 2$, $\phi^{\textit{freq.}}_{\text{delicious}}(\boldsymbol{x}) = 0$, $\phi^{\textit{freq.}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.
- Can also be binary word “presence” features. E.g., $\phi^{\textit{presence}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\textit{presence}}_{\text{the}}(\boldsymbol{x}) = 1$, $\phi^{\textit{presence}}_{\text{delicious}}(\boldsymbol{x}) = 0$, $\phi^{\textit{presence}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.

Features of a Text

Running example:

$\boxed{\boldsymbol{x} = \text{“The vodka was great, but don't touch the hamburgers.”}}$

A different representation of the text sequences: features.

- Often, these are term (word or word sequence) frequencies. E.g., $\phi^{\mathit{freq.}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\mathit{freq.}}_{\text{the}}(\boldsymbol{x}) = 2$, $\phi^{\mathit{freq.}}_{\text{delicious}}(\boldsymbol{x}) = 0$, $\phi^{\mathit{freq.}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.
- Can also be binary word “presence” features. E.g., $\phi^{\mathit{presence}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\mathit{presence}}_{\text{the}}(\boldsymbol{x}) = 1$, $\phi^{\mathit{presence}}_{\text{delicious}}(\boldsymbol{x}) = 0$, $\phi^{\mathit{presence}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.
- Transformations on word frequencies: logarithm, idf weighting

$$\forall v \in \mathcal{V}, \mathrm{idf}(v) = \log \frac{n}{|i : \mathrm{count}_{\boldsymbol{x}_i}(v) > 0|}$$

$$\phi^{\mathit{tfidf}}_{v}(\boldsymbol{x}) = \phi^{\mathit{freq.}}_{v}(\boldsymbol{x}) \cdot \mathrm{idf}(v)$$

Features of a Text

Running example:

$\boldsymbol{x} =$ "The vodka was great, but don't touch the hamburgers."

A different representation of the text sequences: features.

- Often, these are term (word or word sequence) frequencies.
  E.g., $\phi^{\textit{freq.}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\textit{freq.}}_{\text{the}}(\boldsymbol{x}) = 2$, $\phi^{\textit{freq.}}_{\text{delicious}}(\boldsymbol{x}) = 0$,
  $\phi^{\textit{freq.}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.
- Can also be binary word "presence" features.
  E.g., $\phi^{\textit{presence}}_{\text{hamburgers}}(\boldsymbol{x}) = 1$, $\phi^{\textit{presence}}_{\text{the}}(\boldsymbol{x}) = 1$, $\phi^{\textit{presence}}_{\text{delicious}}(\boldsymbol{x}) = 0$,
  $\phi^{\textit{presence}}_{\text{don't touch}}(\boldsymbol{x}) = 1$.
- Transformations on word frequencies: logarithm, idf weighting

$$\forall v \in \mathcal{V}, \mathrm{idf}(v) = \log \frac{n}{|i : \mathrm{count}_{\boldsymbol{x}_i}(v) > 0|}$$

$$\phi^{\textit{tfidf}}_{v}(\boldsymbol{x}) = \phi^{\textit{freq.}}_{v}(\boldsymbol{x}) \cdot \mathrm{idf}(v)$$

- "Bias" feature, $\phi^{\textit{bias}}$ which takes a constant value of 1.

Reflection

Given what you already know about words, can you think of features that might generalize better than the ones just discussed?

Features are Extremely Important!

The features fully determine what a learned model “sees” about an example.

We often stack the features into a **feature vector**: $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^d$, which “embeds” the input $\boldsymbol{x}$ in d-dimensional space

Aperitif: (Binary) Logistic Regression

A logistic regression model is defined by:

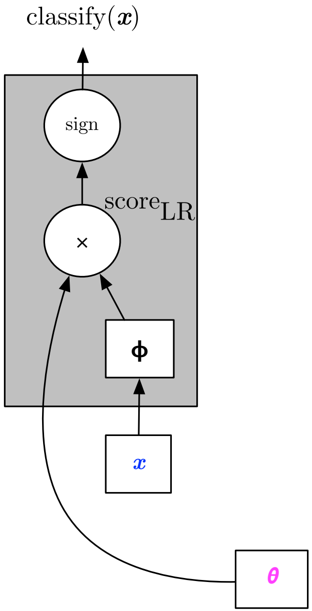
- A collection of feature functions, denoted $\phi_1, \ldots \phi_d$, each mapping $\mathcal{V}^* \rightarrow \mathbb{R}$.
  - **The designer of the system chooses the features.**
- A coefficient or "weight" for every feature, denoted $\theta_1, \ldots, \theta_d$, each $\in \mathbb{R}$.
  - **The weights are "parameters" that are chosen automatically by applying a learning algorithm.**

The label set is $\mathcal{L} = \{+1, -1\}$.

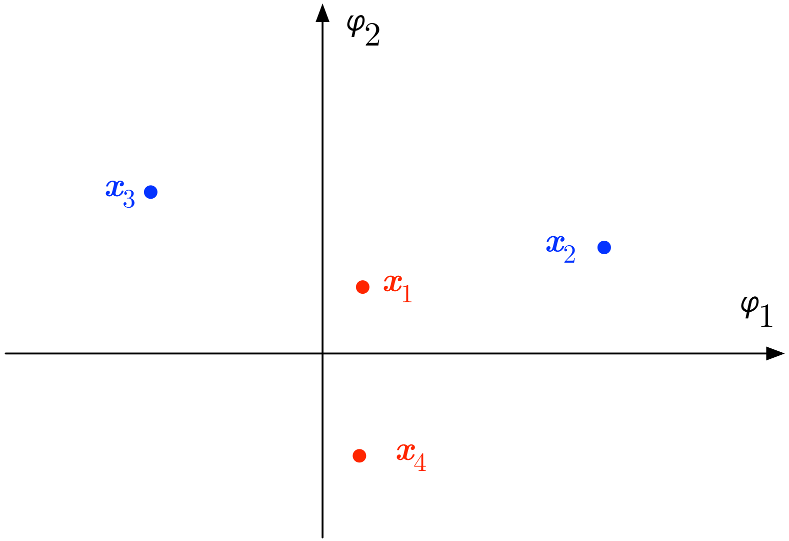
$$\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{j=1}^{d} \theta_j \phi_j(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x})$$

$$\mathrm{classify}_{\mathrm{LR}}(\boldsymbol{x}) = \mathrm{sign}(\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}))$$

Computation Graph View of LR Classifier



Geometric View of LR



Learning a Logistic Regression Classifier

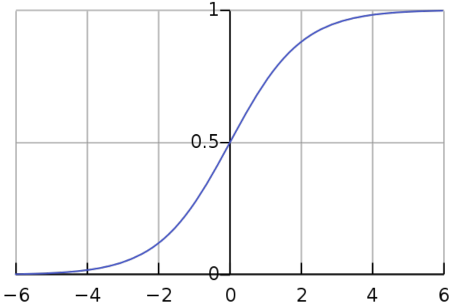
Learning requires us to choose the weight vector, $\boldsymbol{\theta}$.

There are many ways you could do this; logistic regression tells you what vector you should choose based on a probabilistic view of the classifier (but not exactly *how* to find it).

Reflection

Recall the bias feature, $\phi^{bias}(\boldsymbol{x}) = 1$. What role does it play in the geometric interpretation of the model?

Standard Logistic Function



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

Probabilistic View of LR

Our model actually defines a *probability* distribution over the labels $\mathcal{L} = \{+1, -1\}$:

$$p_{\mathrm{LR}}(Y = +1 \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}))$$

Probabilistic View of LR

Our model actually defines a *probability* distribution over the labels $\mathcal{L} = \{+1, -1\}$:

$$p_{\mathrm{LR}}(Y = +1 \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}))$$
$$p_{\mathrm{LR}}(Y = -1 \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = 1 - \sigma(\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(-\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}))$$

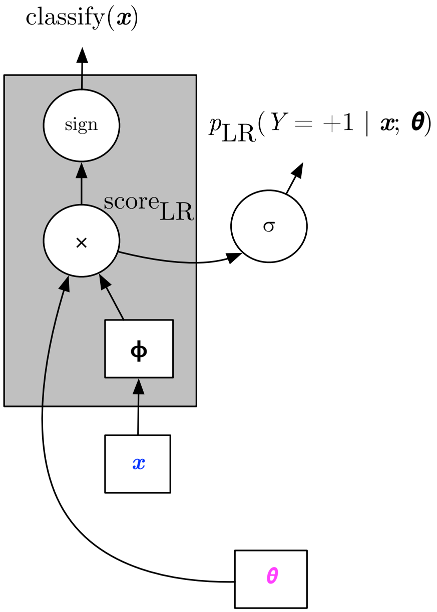
Probabilistic View of LR

Our model actually defines a *probability* distribution over the labels $\mathcal{L} = \{+1, -1\}$:

$$\begin{aligned}
p_{\mathrm{LR}}(Y = +1 \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) &= \sigma(\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta})) \\
p_{\mathrm{LR}}(Y = -1 \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) &= 1 - \sigma(\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(-\mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta})) \\
p_{\mathrm{LR}}(Y = y \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) &= \sigma(y \cdot \mathrm{score}_{\mathrm{LR}}(\boldsymbol{x}; \boldsymbol{\theta}))
\end{aligned}$$

Note: recorded lecture has a mistake on the line above (at 47:35); there should *not* be a minus sign in front of y.

Computation Graph View of LR Probability



$\text{classify}(\boldsymbol{x})$
sign
score_{LR}
$\times$
σ
$p_{\text{LR}}(Y = +1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
$\boldsymbol{\phi}$
$\boldsymbol{x}$
$\boldsymbol{\theta}$

Probabilistic View of LR

This suggests using the principle of maximum likelihood to estimate $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^d} \prod_{i=1}^{n} p_{\mathrm{LR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta})$$

Probabilistic View of LR

This suggests using the principle of maximum likelihood to estimate $\boldsymbol{\theta}$:

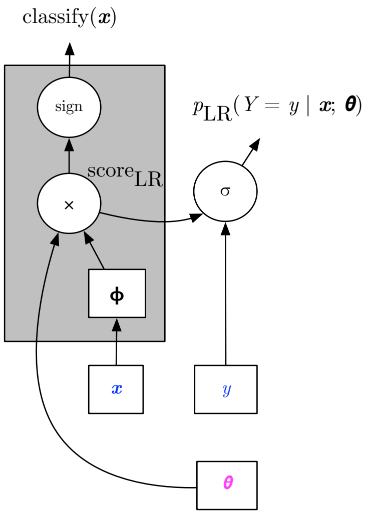
$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \prod_{i=1}^{n} p_{\mathrm{LR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{i=1}^{n} \log p_{\mathrm{LR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta})\end{aligned}$$

Probabilistic View of LR

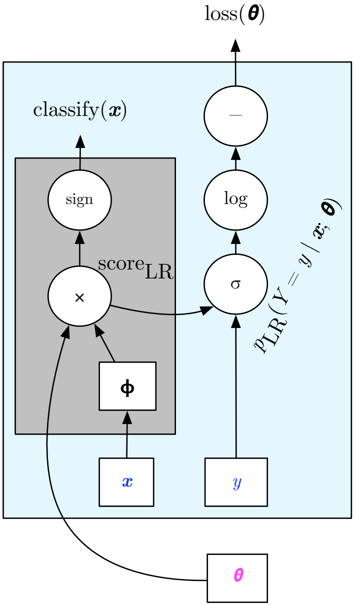
This suggests using the principle of maximum likelihood to estimate $\boldsymbol{\theta}$:

$$
\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \prod_{i=1}^{n} p_{\mathrm{LR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{i=1}^{n} \log p_{\mathrm{LR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{i=1}^{n} \underbrace{-\log p_{\mathrm{LR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta})}_{\text{sometimes called “log loss” or “cross entropy”}}
\end{aligned}
$$

Computation Graph View of LR Probability of Correct Label y

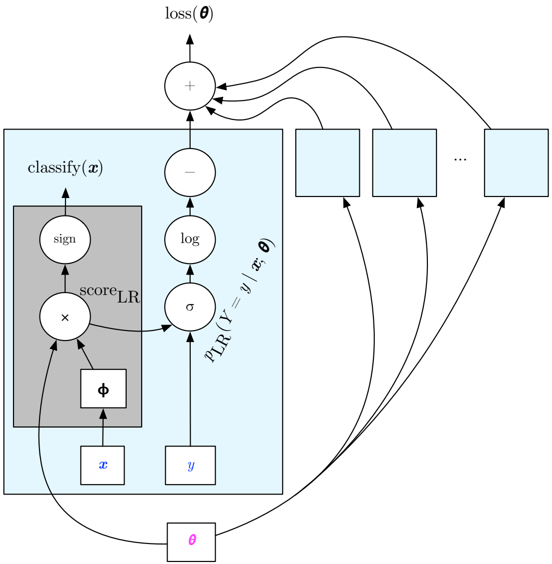

classify($\boldsymbol{x}$)
sign
score_{LR}
×
$p_{\text{LR}}(Y = y \mid \boldsymbol{x}; \boldsymbol{\theta})$
σ
$\boldsymbol{\phi}$
$\boldsymbol{x}$
y
$\boldsymbol{\theta}$

Computation Graph View of Log Loss (One Instance)



$\text{loss}(\boldsymbol{\theta})$
$-$
$\text{classify}(\boldsymbol{x})$
sign
log
score_{LR}
$\times$
σ
$p_{\text{LR}}(Y = y \mid \boldsymbol{x}; \boldsymbol{\theta})$
$\boldsymbol{\phi}$
$\boldsymbol{x}$
y
$\boldsymbol{\theta}$

Computation Graph View of Log Loss (Many Instances)



$\mathrm{loss}(\boldsymbol{\theta})$
$+$
$\mathrm{classify}(\boldsymbol{x})$
$-$
...
sign
log
$\mathrm{score}_{\mathrm{LR}}$
$\times$
σ
$p_{\mathrm{LR}}(Y = y \mid \boldsymbol{x}; \boldsymbol{\theta})$
$\boldsymbol{\phi}$
$\boldsymbol{x}$
y
$\boldsymbol{\theta}$

Learning for Logistic Regression

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \underbrace{\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right)}_{\text{loss}(\boldsymbol{\theta})}$$

Learning for Logistic Regression

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \underbrace{\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right)}_{\text{loss}(\boldsymbol{\theta})}$$

- You can efficiently implement the objective function "loss" given your data and your features ϕ.

Learning for Logistic Regression

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \underbrace{\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right)}_{\mathrm{loss}(\boldsymbol{\theta})}$$

- You can efficiently implement the objective function "loss" given your data and your features ϕ.
- Because it is continuous and differentiable, and the optimization problem is unconstrained, you can use the *gradient* of loss to iteratively move closer to a minimum.

Learning for Logistic Regression

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \underbrace{\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right)}_{\text{loss}(\boldsymbol{\theta})}$$

- You can efficiently implement the objective function "loss" given your data and your features $\boldsymbol{\phi}$.
- Because it is continuous and differentiable, and the optimization problem is unconstrained, you can use the *gradient* of loss to iteratively move closer to a minimum.
- Provable: the function is convex, so these methods will converge to a global minimum. More about this in Eisenstein (2019) Appendix B.

Practical Point: Computing the Gradient

Deriving the gradient of loss with respect to $\boldsymbol{\theta}$, denoted $\nabla_{\boldsymbol{\theta}}\mathrm{loss}$, is left as an exercise.
Hint: use the chain rule from calculus and work backward through the computation graph on slide 47.

Stochastic Gradient Descent

Goal: minimize $\sum_{i=1}^{N} g_i(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

Input: initial value $\boldsymbol{\theta}$, number of epochs T, learning rate α

```
For $t \in \{1, \ldots, T\}$:
    ▶ Choose a random permutation $\pi$ of $\{1, \ldots, N\}$.
    ▶ For $i \in \{1, \ldots, N\}$:
            $\boldsymbol{\theta} \leftarrow \mathbf{w} - \alpha \cdot \nabla_{\boldsymbol{\theta}} g_{\pi(i)}$
```

Output: $\boldsymbol{\theta}$

Reflection

We can prove that SGD will eventually get very close to a global minimum of a *convex* objective function. What do you think will happen if we apply SGD to a function that is not convex?

The Main Dish

Multinomial Logistic Regression

We can generalize LR to an arbitrary label set $\mathcal{L}$.
We need:

1. A more powerful definition of feature functions.
2. An update to the probability distribution.

Input/Output Features

In LR, $\phi_j : \mathcal{V}^* \to \mathbb{R}$ (features only see inputs).

In MLR, $f_j : \mathcal{V}^* \times \mathcal{L} \to \mathbb{R}$ (features consider potential output value, too).

- (We deliberately use "f" instead of "ϕ" here.)

General template:

$$f_{\ell,\phi}(\boldsymbol{x}, y) = \phi(\boldsymbol{x}) \cdot \mathbf{1}\left\{y = \ell\right\}$$

E.g., if $\mathcal{L} = \{\text{sports}, \text{politics}, \text{health}\}$, then we have separate features $f^{\mathit{freq.}}_{\text{sports},\text{vodka}}(\boldsymbol{x}, y)$, $f^{\mathit{freq.}}_{\text{politics},\text{vodka}}(\boldsymbol{x}, y)$, and $f^{\mathit{freq.}}_{\text{health},\text{vodka}}(\boldsymbol{x}, y)$.

Multinomial Logistic Regression

A multinomial logistic regression model is defined by:

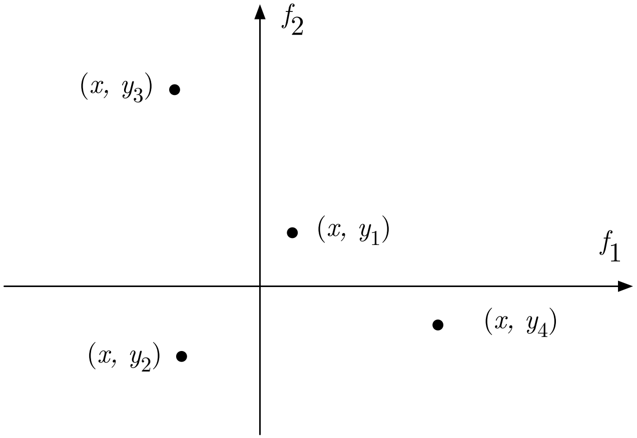
- A collection of feature functions, denoted $f_1, \ldots f_d$, each mapping $\mathcal{V}^* \times \mathcal{L} \rightarrow \mathbb{R}$.
  - **The designer of the system chooses the features.**
- A coefficient or "weight" for every feature, denoted $\theta_1, \ldots, \theta_d$, each $\in \mathbb{R}$.
  - **The weights are "parameters" that are chosen automatically by applying a learning algorithm.**

$$\mathrm{score}_{\mathrm{MLR}}(\boldsymbol{x}, y; \boldsymbol{\theta}) = \sum_{j=1}^{d} \theta_j f_j(\boldsymbol{x}, y) = \boldsymbol{\theta}^\top \mathbf{f}(\boldsymbol{x}, y)$$

$$\mathrm{classify}_{\mathrm{MLR}}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{L}} \mathrm{score}_{\mathrm{MLR}}(\boldsymbol{x}, y; \boldsymbol{\theta})$$

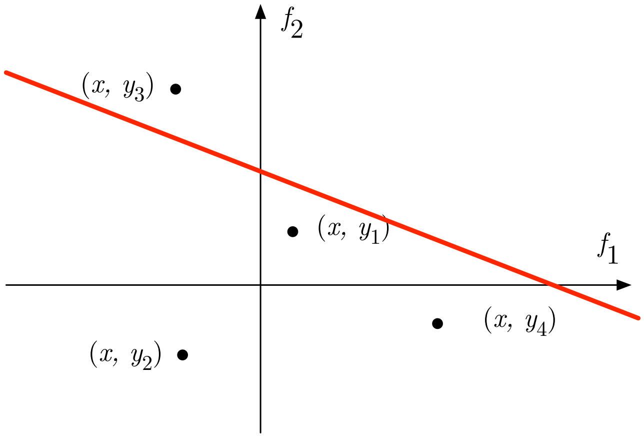
Geometric View of MLR

Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



Geometric View of MLR

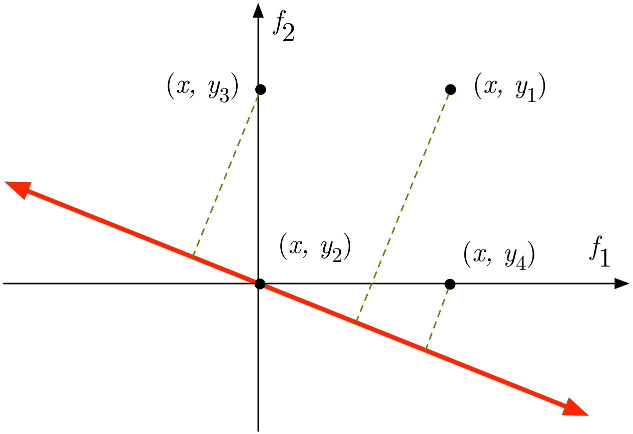
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



$$\boldsymbol{\theta} \cdot \mathbf{f} = \theta_1 f_1 + \theta_2 f_2 = 0$$

Geometric View of MLR

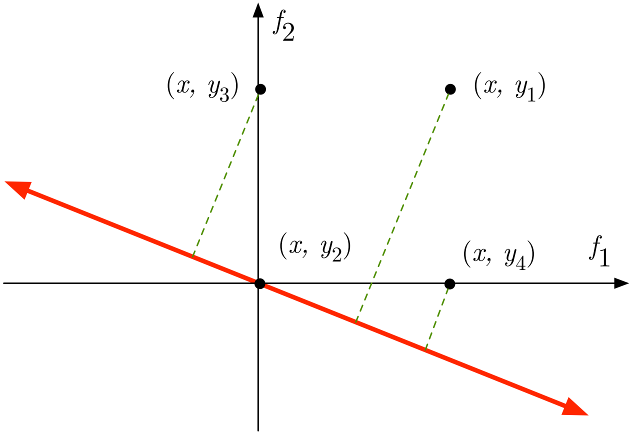
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



$$\text{distance}(\boldsymbol{\theta} \cdot \mathbf{f} = 0, \mathbf{f}_0) = \frac{|\boldsymbol{\theta} \cdot \mathbf{f}_0|}{\|\boldsymbol{\theta}\|_2} \propto |\boldsymbol{\theta} \cdot \mathbf{f}_0|$$

Geometric View of MLR

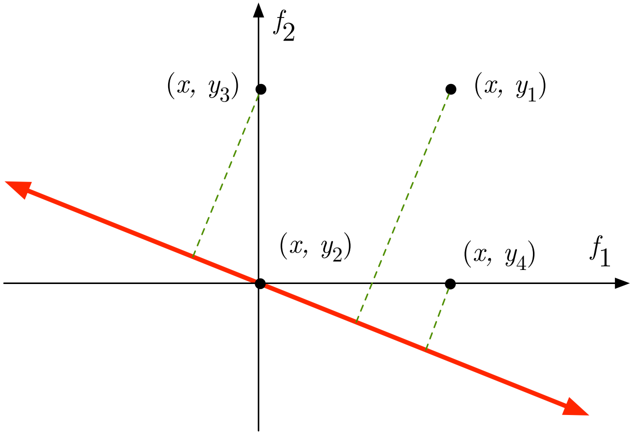
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



$$\boldsymbol{\theta} \cdot \mathbf{f}(x, y_1) > \boldsymbol{\theta} \cdot \mathbf{f}(x, y_3) > \boldsymbol{\theta} \cdot \mathbf{f}(x, y_4) > 0 \geq \boldsymbol{\theta} \cdot \mathbf{f}(x, y_2)$$

Geometric View of MLR

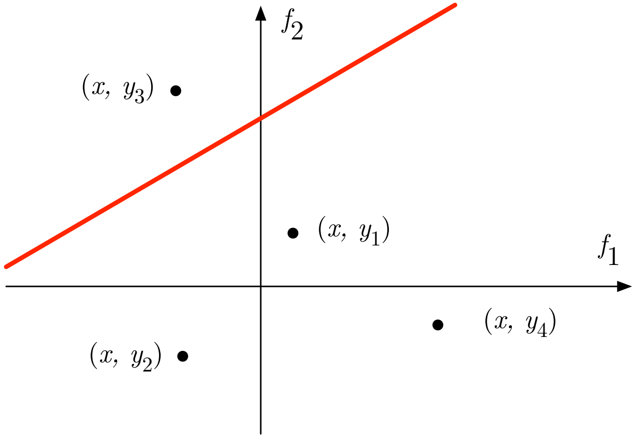
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



$$\text{score}(x, y_1) > \text{score}(x, y_3) > \text{score}(x, y_4) > \text{score}(x, y_2)$$

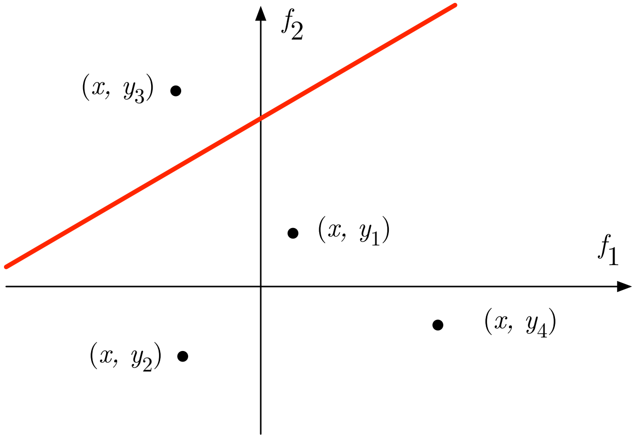
Geometric View of MLR

Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



Geometric View of MLR

Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, f_1 and f_2.



$$\text{score}(x, y_3) > \text{score}(x, y_1) > \text{score}(x, y_2) > \text{score}(x, y_4)$$

Probabilistic View of MLR

Our model defines a *probability* distribution over the labels $\mathcal{L}$.

Probabilistic View of MLR

Our model defines a *probability* distribution over the labels $\mathcal{L}$.

First, we need to introduce a new function from vectors to vectors.

$$\begin{aligned}\operatorname{softmax}\left(\langle t_1, t_2, \ldots, t_k\rangle\right) &= \left\langle \frac{e^{t_1}}{\sum_{j=1}^{k} e^{t_j}}, \frac{e^{t_2}}{\sum_{j=1}^{k} e^{t_j}}, \ldots, \frac{e^{t_k}}{\sum_{j=1}^{k} e^{t_j}} \right\rangle \\ &= \frac{\exp \mathbf{t}}{\|\exp \mathbf{t}\|_1}\end{aligned}$$

Note the use of element-wise exponential:
$\exp(\mathbf{t}) = \langle \exp t_1, \exp t_2, \ldots, \exp t_k\rangle$.

Probabilistic View of MLR

Our model defines a *probability* distribution over the labels $\mathcal{L}$.

$$p_{\mathrm{MLR}}(Y \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \mathrm{softmax}\left(\langle \mathrm{score}_{\mathrm{MLR}}(\boldsymbol{x}, \ell; \boldsymbol{\theta}) \rangle_{\ell \in \mathcal{L}}\right)$$

Probabilistic View of MLR

Our model defines a *probability* distribution over the labels $\mathcal{L}$.

$$p_{\text{MLR}}(Y \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \text{softmax}\left(\langle \text{score}_{\text{MLR}}(\boldsymbol{x}, \ell; \boldsymbol{\theta})\rangle_{\ell \in \mathcal{L}}\right)$$
$$Z(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{\ell' \in \mathcal{L}} \exp \text{score}_{\text{MLR}}(\boldsymbol{x}, \ell'; \boldsymbol{\theta})$$

Probabilistic View of MLR

Our model defines a *probability* distribution over the labels $\mathcal{L}$.

$$\begin{aligned}
p_{\mathrm{MLR}}(Y \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) &= \operatorname{softmax}\left(\langle \mathrm{score}_{\mathrm{MLR}}(\boldsymbol{x}, \ell; \boldsymbol{\theta})\rangle_{\ell \in \mathcal{L}}\right) \\
Z(\boldsymbol{x}; \boldsymbol{\theta}) &= \sum_{\ell' \in \mathcal{L}} \exp \mathrm{score}_{\mathrm{MLR}}(\boldsymbol{x}, \ell'; \boldsymbol{\theta}) \\
p_{\mathrm{MLR}}(Y = \ell \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) &= \frac{\exp \mathrm{score}_{\mathrm{MLR}}(\boldsymbol{x}, \ell; \boldsymbol{\theta})}{Z(\boldsymbol{x}; \boldsymbol{\theta})}
\end{aligned}$$

Probabilistic View of MLR

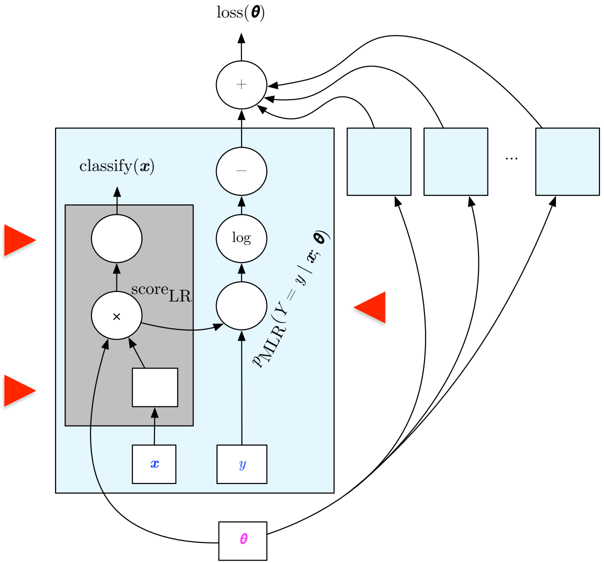
This slide is almost identical to slide 42!

This suggests using the principle of maximum likelihood to estimate $\boldsymbol{\theta}$:

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \prod_{i=1}^{n} p_{\mathrm{MLR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{i=1}^{n} \log p_{\mathrm{MLR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{i=1}^{n} \underbrace{- \log p_{\mathrm{MLR}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta})}_{\text{sometimes called ``log loss'' or ``cross entropy''}}\end{aligned}$$

Reflection: Computation Graph View of MLR

What do you need to change from the LR case?


loss($\boldsymbol{\theta}$)
+
classify($\boldsymbol{x}$)
−
log
score$_{\text{LR}}$
×
$p_{\text{MLR}}(Y = y \mid \boldsymbol{x}; \boldsymbol{\theta})$
$\boldsymbol{x}$
y
$\boldsymbol{\theta}$
...

Learning for Multinomial Logistic Regression

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{i=1}^{n} - \underbrace{\boldsymbol{\theta}^\top \mathbf{f}(\boldsymbol{x}_i, y_i)}_{\text{“hope”}} + \underbrace{\log \sum_{\ell \in \mathcal{L}} \exp(\boldsymbol{\theta}^\top \mathbf{f}(\boldsymbol{x}_i, \ell))}_{\text{“fear”}}$$

See slide 31; all points are the same!

(M)LR Tends to Overfit

If a particular feature f_j is usually positive, then it always improves the loss to increase θ_j.

Regularization: discourage every θ_j from getting too large in magnitude.

Regularization

$$\arg\min_{\boldsymbol{\theta}} \operatorname{loss}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_p^p$$

where $\lambda > 0$ is a “hyperparameter” and $p = 2$ or 1.

ℓ_1 Regularization

This case warrants a little more discussion:

$$\min_{\mathbf{w}} \operatorname{loss}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

Note that:

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d} |\theta_j|$$

- This results in **sparsity** (i.e., many $\theta_j = 0$).

ℓ_1 Regularization

This case warrants a little more discussion:

$$\min_{\mathbf{w}} \text{loss}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

Note that:

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d} |\theta_j|$$

- This results in **sparsity** (i.e., many $\theta_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.

ℓ_1 Regularization

This case warrants a little more discussion:

$$\min_{\mathbf{w}} \text{loss}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

Note that:

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d} |\theta_j|$$

- This results in **sparsity** (i.e., many $\theta_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.
  - Do not confuse it with data *sparseness* (a problem to be overcome)!

ℓ_1 Regularization

This case warrants a little more discussion:

$$\min_{\mathbf{w}} \operatorname{loss}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

Note that:

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d} |\theta_j|$$

- This results in **sparsity** (i.e., many $\theta_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.
  - Do not confuse it with data *sparseness* (a problem to be overcome)!
- This is not differentiable at $\theta_j = 0$.

ℓ_1 Regularization

This case warrants a little more discussion:

$$\min_{\mathbf{w}} \text{loss}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

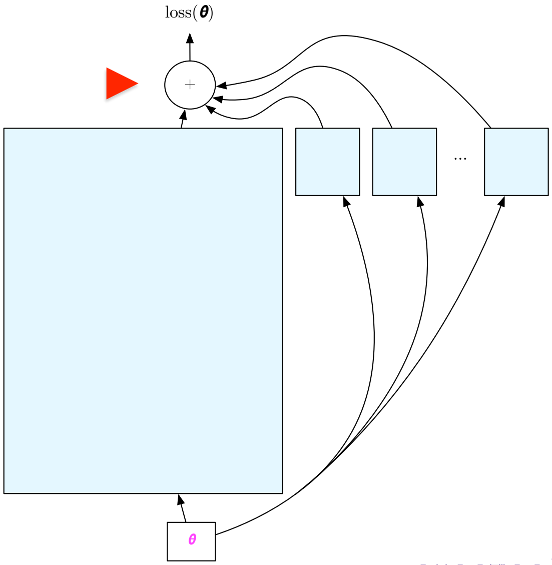
Note that:

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d} |\theta_j|$$

- This results in **sparsity** (i.e., many $\theta_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.
  - Do not confuse it with data *sparseness* (a problem to be overcome)!
- This is not differentiable at $\theta_j = 0$.
- Optimization: special solutions for batch (e.g., Andrew and Gao, 2007) and stochastic (e.g., Langford et al., 2009) settings.

Reflection: Computation Graph View of MLR

What do you need to change for regularization?



MLR Learning

If we had more time, we'd study this problem more carefully!

Here's what you must remember:

- There is no closed form for the objective function; you must use a numerical optimization algorithm like stochastic gradient descent.
- MLR is powerful but expensive ($Z(\boldsymbol{x}_i; \boldsymbol{\theta})$).
- Regularization is very important; we don't actually do MLE. If you want to be absolutely precise, you're minimizing the regularized log loss.

Digestif: Connections

Slight changes to the loss function lead to other well-known learning methods.

- Perceptron: change "fear" to $\max_{\ell \in \mathcal{L}} \operatorname{score}(\boldsymbol{x}, \ell; \boldsymbol{\theta})$
- Linear support vector machine: change "fear" to $\max_{\ell \in \mathcal{L}} \operatorname{score}(\boldsymbol{x}, \ell; \boldsymbol{\theta}) +$ (cost of substituting ℓ for y)

Digestif: Connections

Slight changes to the loss function lead to other well-known learning methods.

- Perceptron: change "fear" to $\max_{\ell \in \mathcal{L}} \mathrm{score}(\boldsymbol{x}, \ell; \boldsymbol{\theta})$
- Linear support vector machine: change "fear" to $\max_{\ell \in \mathcal{L}} \mathrm{score}(\boldsymbol{x}, \ell; \boldsymbol{\theta}) +$ (cost of substituting ℓ for y)

The model I presented as "MLR" has gone by other names:

- Maximum entropy model, because it is provable that $p_{\mathrm{MLR}}(Y \mid \boldsymbol{X}; \boldsymbol{\theta}^*)$ is the distribution with the greatest entropy (uncertainty about Y) under the constraint that $\mathbb{E}_p \mathbf{f} = \tilde{\mathbb{E}} \mathbf{f}$. See Berger et al. (1996).
- Exponential model, because it is a member of the generalized exponential family.

On Data

For machine learning methods, the math can be demanding!

This makes it easy to forget the importance of the data and how we represent it (features).

On Features

Feature engineering is something some people love and others hate.

On Features

Feature engineering is something some people love and others hate.

There have been many attempts to automate it, either by throwing in a huge number and letting the learner decide (e.g., via sparse regularization), or searching for new, complex features by combining simpler ones, or learning them "from scratch."

On Features

Feature engineering is something some people love and others hate.

There have been many attempts to automate it, either by throwing in a huge number and letting the learner decide (e.g., via sparse regularization), or searching for new, complex features by combining simpler ones, or learning them “from scratch.”

Responsible impact: just because you have excluded features that you don’t want your model to know about doesn’t mean you’ve excluded all the *correlates* of those features!

References I

Galen Andrew and Jianfeng Gao. Scalable training of ℓ_1-regularized log-linear models. In *Proc. of ICML*, 2007.

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Jacob Eisenstein. *Introduction to Natural Language Processing*. MIT Press, 2019.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. In *NeurIPS*, 2009.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2011. URL `http://www.morganclaypool.com/doi/pdf/10.2200/S00361ED1V01Y201105HLT013.pdf`.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Extras

Cross-Validation

Remember that $\hat{\mathrm{A}}$, $\hat{\mathrm{P}}$, $\hat{\mathrm{R}}$, and $\hat{F}_1$ are all *estimates* of the classifier's quality under the true data distribution.

- Estimates are noisy!

K-fold cross-validation:

- Partition the training set into K non-overlapping "folds" $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K$.
- For $i \in \{1, \ldots, K\}$:
  - Train on $\boldsymbol{x}_{1:n} \setminus \boldsymbol{x}^i$, using $\boldsymbol{x}^i$ as development data.
  - Estimate quality on the ith development set: $\hat{\mathrm{A}}^i$
- Report the average:

$$\hat{\mathrm{A}} = \frac{1}{K} \sum_{i=1}^{K} \hat{\mathrm{A}}^i$$

  and perhaps also the standard error.

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Is $\mathrm{classify}_1$ better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Is $\mathrm{classify}_1$ better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

How much larger must $\hat{\mathrm{A}}_1$ be than $\hat{\mathrm{A}}_2$ to *reject* H_0?

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Is $\mathrm{classify}_1$ better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

How much larger must $\hat{\mathrm{A}}_1$ be than $\hat{\mathrm{A}}_2$ to *reject* H_0?

Frequentist view: how (im)probable is the observed difference, given $H_0 = \text{true}$?

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Is $\mathrm{classify}_1$ better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

How much larger must $\hat{\mathrm{A}}_1$ be than $\hat{\mathrm{A}}_2$ to *reject* H_0?

Frequentist view: how (im)probable is the observed difference, given $H_0 =$ true?

Caution: statistical significance is neither necessary nor sufficient for research significance or practical usefulness!

A Hypothesis Test for Text Classifiers

McNemar (1947)

1. The null hypothesis: $\mathrm{A}_1 = \mathrm{A}_2$
2. Pick significance level α, an "acceptably" high probability of incorrectly rejecting H_0.
3. Calculate the test statistic, k (explained in the next slide).
4. Calculate the probability of a *more extreme* value of k, assuming H_0 is true; this is the p-value.
5. Reject the null hypothesis if the p-value is less than α.

The p-value is p(this observation $\mid H_0$ is true), not the other way around!

McNemar's Test: Details

Assumptions: independent (test) samples and binary measurements. Count test set error patterns:

	classify_1 is incorrect	classify_1 is correct	
classify_2 is incorrect	c_{00}	c_{10}	
classify_2 is correct	c_{01}	c_{11}	$m \cdot \hat{\mathrm{A}}_2$
		$m \cdot \hat{\mathrm{A}}_1$	

If $\mathrm{A}_1 = \mathrm{A}_2$, then c_{01} and c_{10} are each distributed according to $\mathrm{Binomial}(c_{01} + c_{10}, \frac{1}{2})$.

$$\text{test statistic } k = \min\{c_{01}, c_{10}\}$$

$$p\text{-value} = \frac{1}{2^{c_{01}+c_{10}-1}} \sum_{j=0}^{k} \binom{c_{01} + c_{10}}{j}$$

Other Tests

Different tests make different assumptions.

Sometimes we calculate an interval that would be "unsurprising" under H_0 and test whether a test statistic falls in that interval (e.g., t-test and Wald test).

In many cases, there is no closed form for estimating p-values, so we use random approximations (e.g., permutation test and paired bootstrap test).

If you do lots of tests, you need to correct for that!

Read lots more in Smith (2011), appendix B.