Multiclass classification

examples

| label  | apple | orange | apple | banana | pineapple |

Same setup where we have a set of features for each example
Rather than just two labels, now have 3 or more

Real world multiclass classification

- document classification
- protein classification
- handwriting recognition
- face recognition
- sentiment analysis
- autonomous vehicles

most real-world applications tend to be multiclass

- emotion recognition

Assignment 4
Assignment 3 back soon
If you need assignment feedback…
**Multiclass: current classifiers**

Any of these work out of the box?
With small modifications?

---

**k-Nearest Neighbor (k-NN)**

To classify an example $d$:
- Find $k$ nearest neighbors of $d$
- Choose as the label the majority label within the $k$ nearest neighbors

No algorithmic changes!

---

**Decision Tree learning**

Base cases:
1. If all data belong to the same class, pick that label
2. If all the data have the same feature values, pick majority label
3. If we’re out of features to examine, pick majority label
4. If we don’t have any data left, pick majority label of parent
5. If some other stopping criteria exists to avoid overfitting, pick majority label

Otherwise:
- calculate the “score” for each feature if we used it to split the data
- pick the feature with the highest score, partition the data based on that data value and call recursively

No algorithmic changes!

---

**Perceptron learning**

Hard to separate three classes with just one line 😊
Black box approach to multiclass

Abstraction: we have a generic binary classifier, how can we use it to solve our new problem

binary classifier

+1

-1

optionally: also output a confidence/score

Can we solve our multiclass problem with this?

Approach 1: One vs. all (OVA)

Training: for each label \( L \), pose as a binary problem

- all examples with label \( L \) are positive
- all other examples are negative

<table>
<thead>
<tr>
<th></th>
<th>apple vs. not</th>
<th>orange vs. not</th>
<th>banana vs. not</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>+1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>orange</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>apple</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>orange</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>banana</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>banana</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

How do we classify?

OVA: linear classifiers (e.g. perceptron)
OVA: linear classifiers (e.g. perceptron)

How do we classify?

banana vs. not
apple vs. not
pineapple vs. not

OVA: linear classifiers (e.g. perceptron)

How do we classify?

banana vs. not
apple vs. not
pineapple vs. not

OVA: linear classifiers (e.g. perceptron)

How do we classify?

banana vs. not
apple vs. not
pineapple vs. not

OVA: linear classifiers (e.g. perceptron)

How do we classify?

banana vs. not
apple vs. not
pineapple vs. not
OVA: linear classifiers (e.g. perceptron)

How do we classify?

Classify:
- If classifier doesn't provide confidence (this is rare) and there is ambiguity, pick one of the ones in conflict
- Otherwise:
  - pick the most confident positive
  - if none vote positive, pick least confident negative

What does the decision boundary look like?
OVA: classify, perceptron

Classify:
- If classifier doesn’t provide confidence (this is rare) and there is ambiguity, pick majority in conflict
- Otherwise:
  - pick the most confident positive
  - if none vote positive, pick least confident negative

How do we calculate this for the perceptron?

\[ \text{prediction} = b + \sum_{i=1}^{n} w_i f_i \]
Distance from the hyperplane

Approach 2: All vs. all (AVA)

Training:
For each pair of labels, train a classifier to distinguish between them

for \( i = 1 \) to number of labels:
  for \( k = i+1 \) to number of labels:
    train a classifier to distinguish between \( \text{label}_i \) and \( \text{label}_k \):
      - create a dataset with all examples with \( \text{label}_i \) labeled positive and all examples with \( \text{label}_k \) labeled negative
      - train classifier on this subset of the data
AVA classify

To classify example e, classify with each classifier $f_{jk}$

We have a few options to choose the final class:
- Take a majority vote
- Take a weighted vote based on confidence
  - $y = f_k(e)$
  - $\text{score}_j \leftarrow y$
  - $\text{score}_k \leftarrow -y$

Here we’re assuming that $y$ encompasses both the prediction (+1, -1) and the confidence, i.e. $y = \text{prediction} \times \text{confidence}$.

How does this work?

In general?

Take a weighted vote based on confidence
- $y = f_k(e)$
- $\text{score}_j \leftarrow y$
- $\text{score}_k \leftarrow -y$

If $y$ is positive, classifier thought it was of type $j$:
- raise the score for $j$
- lower the score for $k$

If $y$ is negative, classifier thought it was of type $k$:
- lower the score for $j$
- raise the score for $k$
OVA vs. AVA

Train/classify runtime?

Error? Assume each binary classifier makes an error with probability $\varepsilon$.

OVA vs. AVA

Train time:
AVA learns more classifiers, however, they’re trained on much smaller data this tends to make it faster if the labels are equally balanced.

Test time:
AVA has more classifiers

Error (see the book for more justification):
- AVA trains on more balanced data sets
- AVA tests with more classifiers and therefore has more chances for errors
- Theoretically:
  - OVA: $\varepsilon \cdot (\text{number of labels} - 1)$
  - AVA: $2 \varepsilon \cdot (\text{number of labels} - 1)$

Approach 3: Divide and conquer

Pros/cons vs. AVA?

Multiclass summary

If using a binary classifier, the most common thing to do is OVA.

Otherwise, use a classifier that allows for multiple labels:
- DT and k-NN work reasonably well
- We’ll see a few more in the coming weeks that will often work better
Multiclass evaluation

<table>
<thead>
<tr>
<th>label</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>orange</td>
</tr>
<tr>
<td>orange</td>
<td>orange</td>
</tr>
<tr>
<td>apple</td>
<td>apple</td>
</tr>
<tr>
<td>banana</td>
<td>pineapple</td>
</tr>
<tr>
<td>banana</td>
<td>banana</td>
</tr>
<tr>
<td>pineapple</td>
<td>pineapple</td>
</tr>
</tbody>
</table>

How should we evaluate?

Multiclass evaluation

<table>
<thead>
<tr>
<th>label</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>orange</td>
</tr>
<tr>
<td>orange</td>
<td>orange</td>
</tr>
<tr>
<td>apple</td>
<td>apple</td>
</tr>
<tr>
<td>banana</td>
<td>pineapple</td>
</tr>
<tr>
<td>banana</td>
<td>banana</td>
</tr>
<tr>
<td>pineapple</td>
<td>pineapple</td>
</tr>
</tbody>
</table>

Accuracy: 4/6

Multiclass evaluation imbalanced data

<table>
<thead>
<tr>
<th>label</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>orange</td>
</tr>
<tr>
<td>apple</td>
<td>apple</td>
</tr>
<tr>
<td>banana</td>
<td>pineapple</td>
</tr>
<tr>
<td>banana</td>
<td>banana</td>
</tr>
<tr>
<td>pineapple</td>
<td>pineapple</td>
</tr>
</tbody>
</table>

Any problems?

Data imbalance!

Macroaveraging vs. microaveraging

**microaveraging:** average over examples (this is the “normal” way of calculating)

**macroaveraging:** calculate evaluation score (e.g. accuracy) for each label, then average over labels

What effect does this have?

Why include it?
Macroaveraging vs. microaveraging

**microaveraging**: average over examples (this is the "normal" way of calculating)

**macroaveraging**: calculate evaluation score (e.g. accuracy) for each label, then average over labels

- Puts more weight/emphasis on rarer labels
- Allows another dimension of analysis

Confusion matrix

entry (i, j) represents the number of examples with label i that were predicted to have label j

Another way to understand both the data and the classifier

<table>
<thead>
<tr>
<th></th>
<th>Classic</th>
<th>Country</th>
<th>Disco</th>
<th>Hiphop</th>
<th>Jazz</th>
<th>Rock</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classic</strong></td>
<td>86</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td><strong>Country</strong></td>
<td>1</td>
<td>57</td>
<td>5</td>
<td>1</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td><strong>Disco</strong></td>
<td>0</td>
<td>6</td>
<td>55</td>
<td>4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td><strong>Hiphop</strong></td>
<td>0</td>
<td>15</td>
<td>28</td>
<td>90</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td><strong>Jazz</strong></td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>37</td>
<td>12</td>
</tr>
<tr>
<td><strong>Rock</strong></td>
<td>6</td>
<td>19</td>
<td>11</td>
<td>0</td>
<td>27</td>
<td>48</td>
</tr>
</tbody>
</table>
Confusion matrix

BLAST classification of proteins in 850 superfamilies

Multilabel vs. multiclass classification

- Is it edible?
- Is it sweet?
- Is it a fruit?
- Is it a banana?
- Is it an apple?
- Is it an orange?
- Is it a pineapple?
- Is it yellow?
- Is it round?

Any difference in these labels/categories?

Multiclass vs. multilabel

Multiclass: each example has one label and exactly one label.

Multilabel: each example has zero or more labels. Also called annotation.

Multilabel applications?
Multilabel

- Image annotation
- Document topics
- Labeling people in a picture
- Medical diagnosis

Ranking problems

Suggest a simpler word for the word below:

vital

Suggest a simpler word

Suggest a simpler word for the word below:

vital

Suggest a simpler word for the word below:

acquired

<table>
<thead>
<tr>
<th>word</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>important</td>
<td>13</td>
</tr>
<tr>
<td>necessary</td>
<td>12</td>
</tr>
<tr>
<td>essential</td>
<td>11</td>
</tr>
<tr>
<td>needed</td>
<td>8</td>
</tr>
<tr>
<td>critical</td>
<td>3</td>
</tr>
<tr>
<td>crucial</td>
<td>2</td>
</tr>
<tr>
<td>mandatory</td>
<td>1</td>
</tr>
<tr>
<td>required</td>
<td>1</td>
</tr>
<tr>
<td>vital</td>
<td>1</td>
</tr>
</tbody>
</table>
Suggest a simpler word

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>gotten</td>
<td>12</td>
</tr>
<tr>
<td>received</td>
<td>9</td>
</tr>
<tr>
<td>gained</td>
<td>8</td>
</tr>
<tr>
<td>obtained</td>
<td>5</td>
</tr>
<tr>
<td>got</td>
<td>3</td>
</tr>
<tr>
<td>purchased</td>
<td>2</td>
</tr>
<tr>
<td>bought</td>
<td>2</td>
</tr>
<tr>
<td>got hold of</td>
<td>1</td>
</tr>
<tr>
<td>acquired</td>
<td>1</td>
</tr>
</tbody>
</table>

Suggest a simpler word

Vital

Important

Necessary

Essential

Needed

Crucial

Critical

Mandatory

Required

Vital

Gotten

Received

Gained

Obtained

Got

Purchased

Bought

Got hold of

Acquired

Training data

List of synonyms

List ranked by simplicity

Ranking problems in general

Real-world ranking problems?
Netflix My List

Search

Ranking Applications
- reranking N-best output lists
  - machine translation
  - computational biology
  - parsing
  - ...
- flight search
  - ...

Black box approach to ranking
Abstraction: we have a generic binary classifier, how can we use it to solve our new problem

Can we solve our ranking problem with this?
Predict better vs. worse

Train a classifier to decide if the first input is better than second:
- Consider all possible pairings of the examples in a ranking
- Label as positive if the first example is higher ranked, negative otherwise

\[ f_1, f_2, \ldots, f_n \]

\[ f_1, f_2, \ldots, f_n \]

\[ f_1, f_2, \ldots, f_n \]

\[ f_1, f_2, \ldots, f_n \]

\[ f_1, f_2, \ldots, f_n \]

How can we do this?
We want features that compare the two examples.
Combined feature vector

Many approaches! Will depend on domain and classifier

Two common approaches:
1. difference:
   \[ f'_1 = a_i - b_i \]
2. greater than/less than:
   \[ f'_i = \begin{cases} 
   1 & \text{if } a_i > b_i \\
   0 & \text{otherwise} 
   \end{cases} \]

Training

Testing
Testing

for each binary example $e_{jk}$:

$\text{label}[j] += f_{jk}(e_{jk})$

$\text{label}[k] -= f_{jk}(e_{jk})$

rank according to label scores

An improvement?

new examples

binary label

Are these two examples the same?
Weighted binary classification

<table>
<thead>
<tr>
<th>ranking1</th>
<th>new examples</th>
<th>weighted label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 ... n</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>1 2 3 ... n</td>
<td>+2</td>
</tr>
<tr>
<td></td>
<td>1 2 3 ... n</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1 2 3 ... n</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>1 2 3 ... n</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>1 2 3 ... n</td>
<td>-1</td>
</tr>
</tbody>
</table>

Weight based on distance in ranking

Testing

If the classifier outputs a confidence, then we've learned a distance measure between examples.

During testing we want to rank the examples based on the learned distance measure.

Ideas?

Testing

If the classifier outputs a confidence, then we've learned a distance measure between examples.

During testing we want to rank the examples based on the learned distance measure.

Sort the examples and use the output of the binary classifier as the similarity between examples!
Ranking evaluation

<table>
<thead>
<tr>
<th>ranking</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>1</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>2</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>3</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>4</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>5</td>
</tr>
</tbody>
</table>

Idea 1: accuracy

<table>
<thead>
<tr>
<th>ranking</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>1</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>2</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>3</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>4</td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>5</td>
</tr>
</tbody>
</table>

1/5 = 0.2

Any problems with this?

Doesn't capture “near” correct

<table>
<thead>
<tr>
<th>ranking</th>
<th>prediction</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

1/5 = 0.2

Idea 2: correlation

<table>
<thead>
<tr>
<th>ranking</th>
<th>prediction</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>f₁, f₂, ..., fₙ</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Look at the correlation between the ranking and the prediction