Multiclass classification

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

Most real-world applications tend to be multiclass

Document classification

Handwriting recognition

Protein classification

Face recognition

Sentiment analysis

Emotion recognition

Autonomous vehicles
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 the 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

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

OVA: linear classifiers (e.g. perceptron)

How do we classify?
OVA: linear classifiers (e.g. perceptron)

13

pineapple vs. not

apple vs. not

How do we classify?

14

banana vs. not

pineapple vs. not

apple vs. not

How do we classify?

15

apple vs. not

banana vs. not

pineapple vs. not

How do we classify?

16

banana

OR

pineapple

none?
OVA: linear classifiers (e.g. perceptron)

- Banana vs. not
- Apple vs. not
- Pineapple vs. not

How do we classify?

OVA: 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?

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 label $i$ and label $k$:
    - create a dataset with all examples with label $i$, labeled positive and all examples with label $k$, labeled negative
    - train classifier on this subset of the data

AVA training visualized
AVA classify

apple vs orange
+1
+1
-1

apple vs banana
+1
+1
-1

What class?

In general?

AVA classify

To classify example e, classify with each classifier $f_k$

We have a few options to choose the final class:
- Take a majority vote
- Take a weighted vote based on confidence
  - $y = f_k(e)$
  - Score $j$ += $y$
  - Score $k$ -= $y$

Here we're assuming that $y$ encompasses both the prediction (+1,-1) and the confidence, i.e. $y$ = prediction * confidence.

AVA classify

Take a weighted vote based on confidence
- $y = f_k(e)$
- Score $j$ += $y$
- Score $k$ -= $y$

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 $\varepsilon$

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, so often it is slower

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: $\varepsilon (\text{number of labels} - 1)$
  - AVA: $2 \varepsilon (\text{number of labels} - 1)$

Approach 3: Divide and conquer

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>Classic</td>
<td>86</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>Country</td>
<td>1</td>
<td>57</td>
<td>5</td>
<td>1</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Disco</td>
<td>0</td>
<td>6</td>
<td>55</td>
<td>4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Hiphop</td>
<td>0</td>
<td>15</td>
<td>28</td>
<td>90</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>Jazz</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>37</td>
<td>12</td>
</tr>
<tr>
<td>Rock</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 yellow?
- Is it sweet?
- Is it round?

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 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

Suggest a simpler word for the word below:

<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

<table>
<thead>
<tr>
<th>vital</th>
<th>acquired</th>
</tr>
</thead>
<tbody>
<tr>
<td>important</td>
<td>gotten</td>
</tr>
<tr>
<td>necessary</td>
<td>received</td>
</tr>
<tr>
<td>essential</td>
<td>gained</td>
</tr>
<tr>
<td>needed</td>
<td>obtained</td>
</tr>
<tr>
<td>critical</td>
<td>got</td>
</tr>
<tr>
<td>crucial</td>
<td>purchased</td>
</tr>
<tr>
<td>mandatory</td>
<td>bought</td>
</tr>
<tr>
<td>required</td>
<td>got hold of</td>
</tr>
<tr>
<td>acquired</td>
<td></td>
</tr>
</tbody>
</table>

Ranking problems in general

Ranking1  Ranking2  Ranking3

training data: a set of rankings where each ranking consists of a set of ranked examples

Real-world ranking problems?
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

![Diagram of binary classifier](binary_classifier.png)

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
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

new examples | binary label
---|---
+1
-1
+1
-1
+1
-1
-1

Our binary classifier only takes one example as input

Many approaches! Will depend on domain and classifier

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

New examples 🡏 Existing features 🡏 Label 🡏

+1 +1
-1 +1
-1

\[ f'_1, f'_2, \ldots, f'_n \]

Extract features 🡏 Train classifier 🡏 Binary classifier 🡏

Testing

Unranked 🡏 Ranking?

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

Extract features 🡏 Binary classifier 🡏
What is the ranking? Algorithm?

for each binary example $e_{jk}$:

- label[j] += $f_j(e_{jk})$
- label[k] -= $f_k(e_{jk})$

rank according to label scores

An improvement?

Are these two examples the same?

Weighted binary classification

Weight based on distance in ranking
### Weighted binary classification

New examples with weighted label:

-1

In general, can weight with any consistent distance metric.

Can we solve this problem?

### 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>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Ideas?
**Idea 1: accuracy**

<table>
<thead>
<tr>
<th>ranking</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>1</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>2</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>3</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>4</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>5</td>
</tr>
</tbody>
</table>

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>$f_1 f_2 \ldots f_n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$f_1 f_2 \ldots f_n$</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

$1/5 = 0.2$