Intro to deep learning

Dr. Janoś Gabler, University of Bonn

Lecture 6: Classification via feature extraction

Topics

- More numpy features
- Extracting features from a transformer model
- Use them for classification

Motivation

- After today, you can use state of the art transformer models to create features for your own classifier
- This means you can do high quality sentiment analysis with non-standard categories
- No GPU needed
- Today we can do 3 tasks in a row (without slides in between) because by now you have a solid foundation

More numpy features

Masked numpy arrays

```
>>> import numpy as np
>>> a = np.ma.array(
...     [1, 2, 3],
...     mask=[False, False, True],
... )
>>> a
```

```
masked_array(data=[1, 2, --],
             mask=[False, False,  True],
       fill_value=999999)
```

```
>>> a.data
```

```
array([1, 2, 3])
```

```
>>> a.sum()
```

```
3
```

- creating it requires a mask
- `.data` returns unmasked array
- masked elements are ignored in calculations.
- masked means, entry in the mask is True!
- Similar to pandas operations that ignore NaNs

Masked arrays in higher dimensions

```
>>> b = np.ma.array(
...   [[1, 2], [3, 4]],
...   mask=[[True, False], [False, True]],
... )
>>> b
```

```
masked_array(
  data=[[--, 2],
        [3, --]],
  mask=[[ True, False],
        [False,  True]],
  fill_value=999999)
```

```
>>> b.sum(axis=0)
```

```
masked_array(data=[3, 2],
             mask=[False, False],
       fill_value=999999)
```

```
>>> b.sum(axis=1).data
```

```
array([2, 3])
```

- Mask needs to have same shape as array
- Axis argument behaves as normal
- If result is not a scalar, it as a masked array too
- Only access unmasked data if you know it is safe

Boolean arrays

```
>>> np.array([1, 0, 1]).astype(bool)
array([ True, False,  True])
>>> np.array([1, 0, 15]).astype(bool)
array([ True, False,  True])
>>> np.array([1, -1, 15]).astype(bool)
array([ True,  True,  True])
>>> np.array([True, False, True]).sum()
2
```

- Integers can be converted to bool
  - 0 $\longrightarrow$ False
  - nonzero $\longrightarrow$ True
- For calculations they are implicitly converted back to ints
  - False $\longrightarrow$ 0
  - True $\longrightarrow$ 1

Repeating arrays

```
>>> a = np.array([1, 2, 3])
>>> a.repeat(2)
```

```
array([1, 1, 2, 2, 3, 3])
```

```
>>> b = np.array([[1, 2], [3, 4]])
>>> b.repeat(2)
```

```
array([1, 1, 2, 2, 3, 3, 4, 4])
```

```
>>> b.repeat(2, axis=1)
```

```
array([[1, 1, 2, 2],
       [3, 3, 4, 4]])
```

```
>>> a.reshape(-1, 1).repeat(2, axis=1)
```

```
array([[1, 1],
       [2, 2],
       [3, 3]])
```

- Repeat duplicates elements n times
- Without axis, result is flattened
- Versatile together with reshaping
- Tip for complex cases:
  - First introduce new dimensions via reshaping
  - Then repeat along these axes
- Always prefer broadcasting over repetition!

Task 1

(5 min)

Feature extraction

Steps for the feature extraction

- Tokenize the entire dataset using `DatasetDict.map`
- Write a `map` compatible function to extract last hidden states
  - convert inputs to torch tensors
  - evaluate model
  - convert output to numpy
  - average over unmasked tokens
- Create arrays we can use in sklearn

Tokenize the entire dataset

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("rotten_tomatoes")
>>> model_name = "distilbert-base-uncased"
>>> tokenizer = AutoTokenizer.from_pretrained(model_name)
>>> def tokenize(batch):
...     return tokenizer(batch["text"], padding=True, trun

>>> ds_encoded = ds.map(tokenize, batched=True, batch_size
>>> ds_encoded.column_names
```

```
{'train': ['text', 'label', 'input_ids', 'attention_mask']
 'validation': ['text', 'label', 'input_ids', 'attention_m
 'test': ['text', 'label', 'input_ids', 'attention_mask']}
```

- We did all of this last week
- This is just a condensed summary

Create tiny model inputs to practice

```
>>> import torch
>>> batch = ds_encoded["train"][:2]
>>> input_ids = torch.tensor(batch["input_ids"])
>>> input_ids.shape
```

```
torch.Size([2, 78])
```

```
>>> attention_mask = torch.tensor(batch["attention_mask"])
>>> attention_mask.shape
```

```
torch.Size([2, 78])
```

- `batch` has the same format as what we get when using `map` with `batched=True` on `ds_encoded`
- `shape[0]` is 2 because we have two tweets
- `shape[1]` is 78 because that is the number of tokens in the longest tweet

Using the model

```
>>> from transformers import AutoModel
>>> model = AutoModel.from_pretrained(model_name)
>>> with torch.no_grad():
...     output = model(input_ids, attention_mask)
...     lhs = output.last_hidden_state.cpu().numpy()
>>> lhs.shape
```

```
(2, 78, 768)
```

- The shape is [`batch_size`, `n_tokens`, `hidden_dim`]
- `hidden_dim` is the model specific length of the hidden states
- Thus, there is one hidden state vector for each individual token!
- use `no_grad` to save resources

Task 2

(8 min)

Why do we need post-processing?

- Currently, we would get 78 * 768 = 59904 features
- Some of them correspond to padding tokens
- Want to reduce size and discard invalid information
- Practical solution:
  - Average over token dimension
  - Ignore rows where attention mask is 0

Questions

- Are we allowed to do that?
- Won’t this discard too much information?
- Remember that we are not doing econometrics

Task 3

(8 min)

Task 4

(10 min)

Task 5

(10 min)

How to improve performance?

- Experiment with other classification models
- Tune hyperparameters of classification models
  - We have reached a number of features where penalties make sense
- Add features from another transformer model
- Address class imbalance by resampling the data
- Try other post-processing
  - Keep first
  - Keep last valid

Enacom

- Free start up coaching from university
- Can get help for many things
  - How to develop an idea into a product
  - How to select and apply for grants
  - Legal advice
- Connect with other founders

My Startup Idea (Postponed)

- The statistics package of the future
- You talk to the package in natural language
- A model generates high quality code and answers your statistics questions
- Web interface
  - No installation needed
  - Powerful hardware
- Paired with an open source implementation of modern statistical methods
- Status: Postponed for lack of a full-time co-founder

How Enacom helped me

- Forced me to clarify the vision
- Information on funding opportunities
- Practical tips from Jakob