Intro to deep learning

Dr. Janoś Gabler, University of Bonn

Lecture 6: Classification via feature extraction

Topics

- Extracting features from a transformer model
- Use them for classification
- Compare the performance of pre-trained classifiers and your classifier

What we will have to do

- Tokenize the entire emotions dataset using `DatasetDict.map`
- Write a `map` compatible function to extract last hidden states
  - process inputs
  - evaluate model
  - convert output to numpy
  - do some post processing
- 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']}
```

- Do all imports and loading from scratch
- Use this as cheat sheet when you have to do it in practice

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])
```

- This batch will have 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!