Natural Language Processing

Neural Classification

Ana Marasović
University of Utah

Some slides based on lectures from Greg Durrett

Byte-Pair-Encoding (BPE) – **In Python (Demo)**

utah-cs6340-f24_tokenization.ipynb

https://github.com/huggingface/tokenizers +
notebooks/examples/tokenizer_training.ipynb at main · huggingface/notebooks · GitHub
+ https://huggingface.co/learn/nlp-course/en/chapter6/5

https://github.com/openai/tiktoken

https://github.com/karpathy/minbpe/tree/master

Tiktokenizer (Demo)

Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization.** .DefaultCellStyle is a single token
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization.**
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization.**
- Why is LLM bad at simple arithmetic? **Tokenization.** Tokenization of numbers [Integer tokenization is insane]
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization.** Whitespaces
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization.** Special tokens
- What is this weird warning I get about a "trailing whitespace"? **Tokenization.**
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization.** [SolidGoldMagikarp — LessWrong]
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization.**
- Why is LLM not actually end-to-end language modeling? **Tokenization.**
- What is the real root of suffering? **Tokenization.**

[`Let's build the GPT Tokenizer' by Andrej Karpathy]

Goals & Overview of August 26 Lecture

Goal: Learn one modern algorithm for tokenizing text

- Which units for tokens?
- BPE tokenizer
- Few additional notes

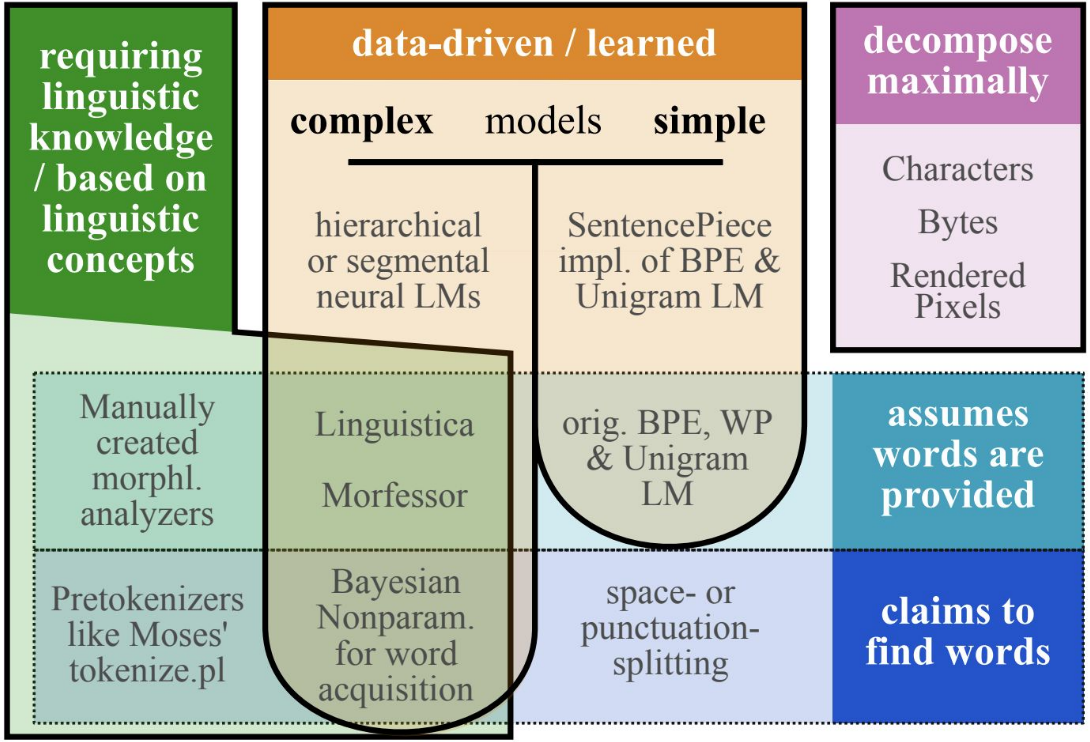


Figure 1: A taxonomy of segmentation and tokenization algorithms and research directions

[Mielke et al., 2021]

Other subword tokenizers

WordPiece [Schuster and Nakajima, 2012]

- Doesn't merge most often co-occuring pair but pairs that increase the likelihood that an n-gram based language model trained with this updated vocabulary reaches on data
- WordPiece tokenization - Hugging Face NLP Course

UnigramLM [Kudo, 2018]

- Unigram tokenization - Hugging Face NLP Course

SentencePiece [Kudo and Richardson, 2018]

- Issue: Not all languages use spaces to separate words.
- One possible solution is to use language specific pre-tokenizers
- To solve this problem more generally:
  Treat the input as a raw input stream, thus including the space in the set of characters to use

4 components of a supervised ML for binary classification

1. **A feature representation of the input**
   - Tokenize text; Subword tokenization
   - Map a list of tokens into a high-dimensional feature vector
2. **A classification function that decides which class to apply to an instance**
   - Extra: Probabilistic classifier
   - Logistic regression
3. **An objective function that we want to optimize for to learn appropriate model parameters**
   - Minimizing negative log likelihood
4. **An algorithm for finding optimal model parameters/weights for the objective function**
   - Gradient descent

4 components of a supervised ML for binary classification

1. **A feature representation of the input**
   - Tokenize text; Subword tokenization
   - Map a list of tokens into a high-dimensional feature vector
2. **A classification function that decides which class to apply to an instance**
   - Extra: Probabilistic classifier
   - Logistic regression
   - Feedforward neural network
3. **An objective function that we want to optimize for to learn appropriate model parameters**
   - Minimizing negative log likelihood
4. **An algorithm for finding optimal model parameters/weights for the objective function**
   - Gradient descent

Goals & Overview of Today's Lecture

Goal: Learn a neural classification function for multiclass classification

- Motivation: Nonlinear transformations
- Multiclass logistic regression
- Feedforward neural network (FFNN)
- Training FFNN
- Practicalities

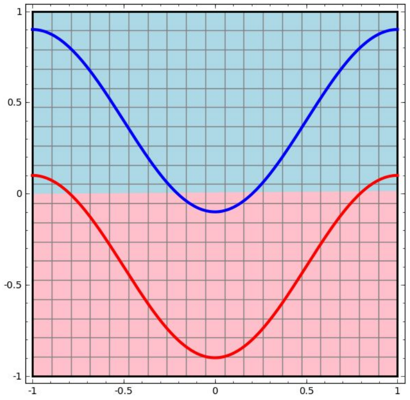
Goals & Overview of Today's Lecture

Goal: Learn a <u>neural</u> classification function for <u>multiclass</u> classification

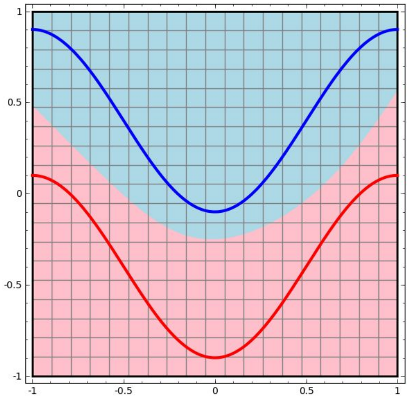
- Motivation: Nonlinear transformations
- Multiclass logistic regression
- Feedforward neural network (FFNN)
- Training FFNN
- Practicalities

Motivation

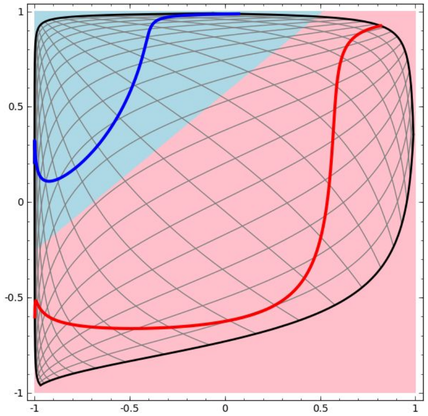
Linear classification



Neural classification



Linear classification in the transformed space



Neural networks transform the data into a ("nicer") so-called **latent** or **hidden feature space**

Source: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

point-wise application of a non-linear function

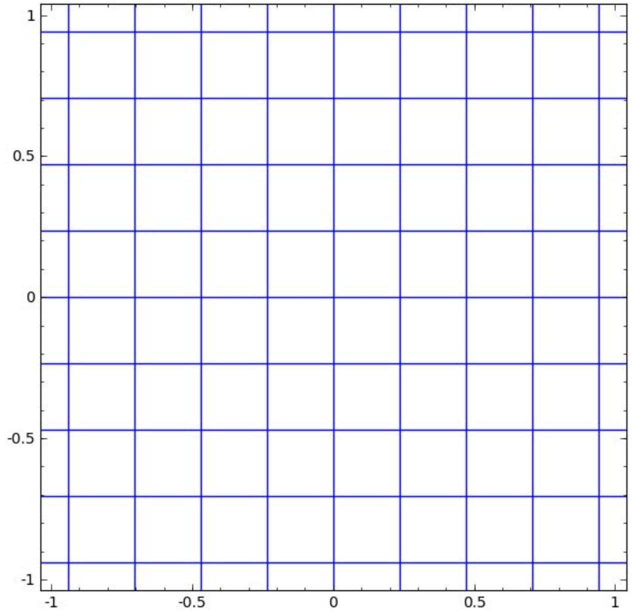
n-dim. feature vector

Visualization of $z = g(W \cdot f(x))$

dxn weight matrix

Pick a vector at the beginning of the visualization
Track what is happening to it

1. A linear transformation by `W`
   - `W` scales, rotates, or otherwise linearly changes the original vector
2. A translation by the vector b (the bias term)
   - b shifts all the points in the linearly transformed space by the same amount and in the same direction
3. Point-wise application of g
   - It can bend, stretch, compress different parts of the linearly transformed space

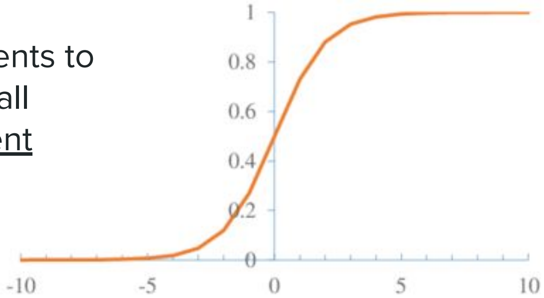


Source: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Nonlinear functions (or activation functions)

Sigmoid

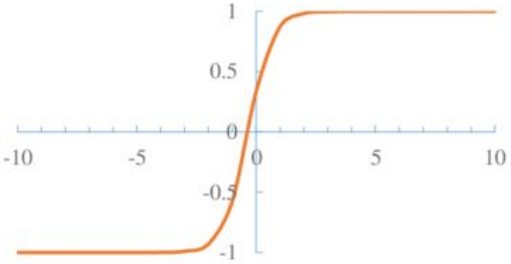
- Can cause gradients to become very small (vanishing gradient problem)



(a)

Tanh

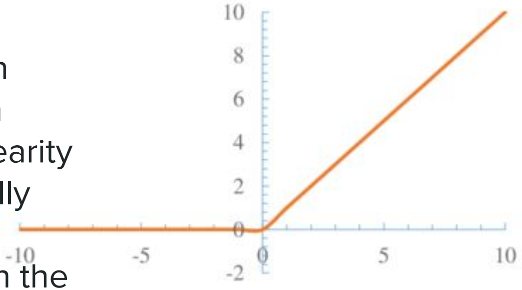
- Squashes between -1 and 1
- Can cause gradients to become very small (vanishing gradient problem)



(b)

ReLU

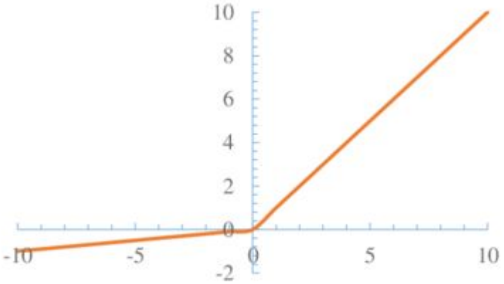
- The most common activation function
- Introduces nonlinearity & is computationally efficient
- Doesn't saturate in the positive domain (no vanishing gradient problem)



(c)

Leaky ReLU function

- Dying ReLUs: Getting stuck in th inactive (zero) part of the ReLU function, where th gradient is zero
- Address this



(d)

Source: https://www.researchgate.net/figure/Nonlinear-function-a-Sigmoid-function-b-Tanh-function-c-ReLU-function-d-Leaky_fig3_323617663

Deep neural network; Deep learning

Repeat the operations that we have seen before

$$z_1 = g(W_1 \cdot f(x))$$

$$z_2 = g(W_2 \cdot z_1)$$

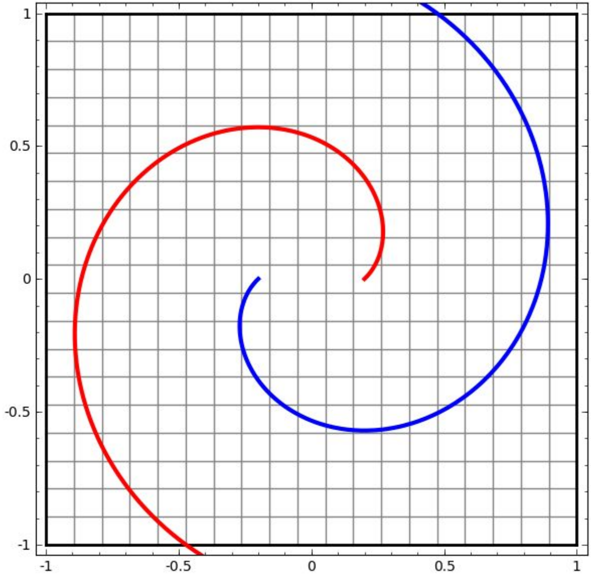
$$z_3 = g(W_3 \cdot z_2)$$

$$\vdots$$

$$z_L = g(W_{L-1} \cdot z_{L-1})$$

Should be L

Each of these operations is called a layer



Source: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Goals & Overview of Today's Lecture

Goal: Learn a neural classification function for multiclass classification

- Motivation: Nonlinear transformations
- Multiclass logistic regression
- Feedforward neural network (FFNN)
- Training FFNN
- Practicalities

Reminder: The sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}} \ldots \text{sigmoid function}$$

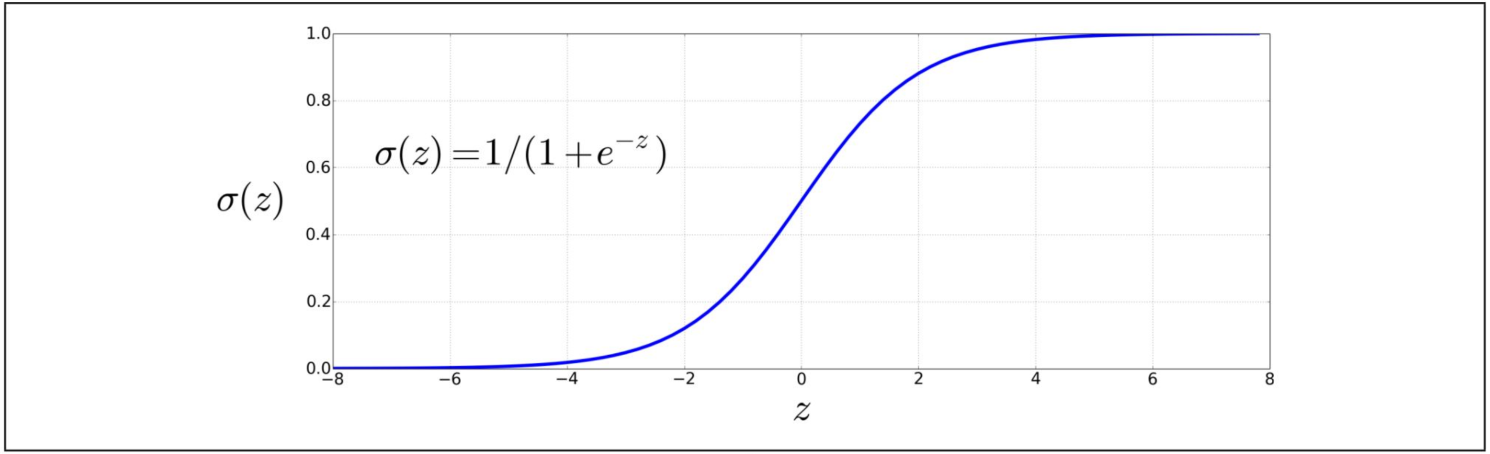


Figure 5.1 The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $(0,1)$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

Source: [Jurafsky & Martin]

Reminder: Logistic Regression for **Binary Classification**

Foundation for many techniques in this course, including neural networks

$$\sigma(z) = \frac{1}{1+e^{-z}} \ldots \text{sigmoid function}$$

$$\mathbb{P}(y=1|x) = \sigma(w^T f(x)) = \frac{1}{1+e^{-w^T f(x)}}$$

$$\mathbb{P}(y=0|x) = 1-\sigma(w^T f(x)) = 1-\frac{1}{1+e^{-w^T f(x)}} = \frac{e^{-w^T f(x)}}{1+e^{-w^T f(x)}}$$

Multiclass classification

A classification problem where the goal is to assign one label from a set of multiple (more than two) possible classes to each input instance

Output space Y: A set of all possible class labels

Two techniques:

1. <u>One weight vector per class</u>
2. Different features per class

a weight vector for class y

$$\text{argmax}_{y \in Y} \, w_y^T f(x)$$

$$\text{argmax}_{y \in Y} \, w^T f(x, y)$$

Multiclass logistic regression

$$p(y = y_i|x) = \frac{e^{w_{y_i}^T f(x)}}{\boxed{\sum_{y_j \in Y} e^{w_{y_j}^T f(x)}}}$$

normalizing to have the sum of probabilities be 1

$$w_1^T f(x) = -1.1 \rightarrow p(y = y_1|x) = 0.036$$

$$w_2^T f(x) = 2.1 \rightarrow p(y = y_2|x) = 0.89$$

$$w_3^T f(x) = -0.4 \rightarrow p(y = y_3|x) = 0.07$$

Multiclass logistic regression; Softmax

$$p(y = y_i|x) = \frac{e^{w_{y_i}^T f(x)}}{\sum_{y_j \in Y} e^{w_{y_j}^T f(x)}}$$

normalizing to have the sum of probabilities be 1

This is an instance of **softmax** function

$$\boldsymbol{z} = [z_1, \ldots, z_K]$$

$$\mathrm{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$w_1^T f(x) = -1.1 \rightarrow p(y = y_1|x) = 0.036$$

$$w_2^T f(x) = 2.1 \rightarrow p(y = y_2|x) = 0.89$$

$$w_3^T f(x) = -0.4 \rightarrow p(y = y_3|x) = 0.07$$

Multiclass logistic regression – Compactly written

$$\boldsymbol{y} = [y_1, \dots, y_m]$$

$$p(y = y_i | x) = \frac{e^{w_{y_i}^T f(x)}}{\sum\limits_{y_j \in Y} e^{w_{y_j}^T f(x)}}, w_{y_i} \in \mathbb{R}^d$$

$$W_o = \begin{bmatrix} w_{y_1}^T \\ \vdots \\ w_{y_m}^T \end{bmatrix} \in \mathbb{R}^{m \times d}$$

$$p(\boldsymbol{y}|x) = \mathrm{softmax}(W_o f(x)) \in \mathbb{R}^m$$

$$y_{\text{pred}} = \mathrm{argmax}_i \; p(\boldsymbol{y}|x)$$

Goals & Overview of Today's Lecture

Goal: Learn a neural classification function for multiclass classification

- Motivation: Nonlinear transformations
- Multiclass logistic regression
- Feedforward neural network (FFNN)
- Training FFNN
- Practicalities

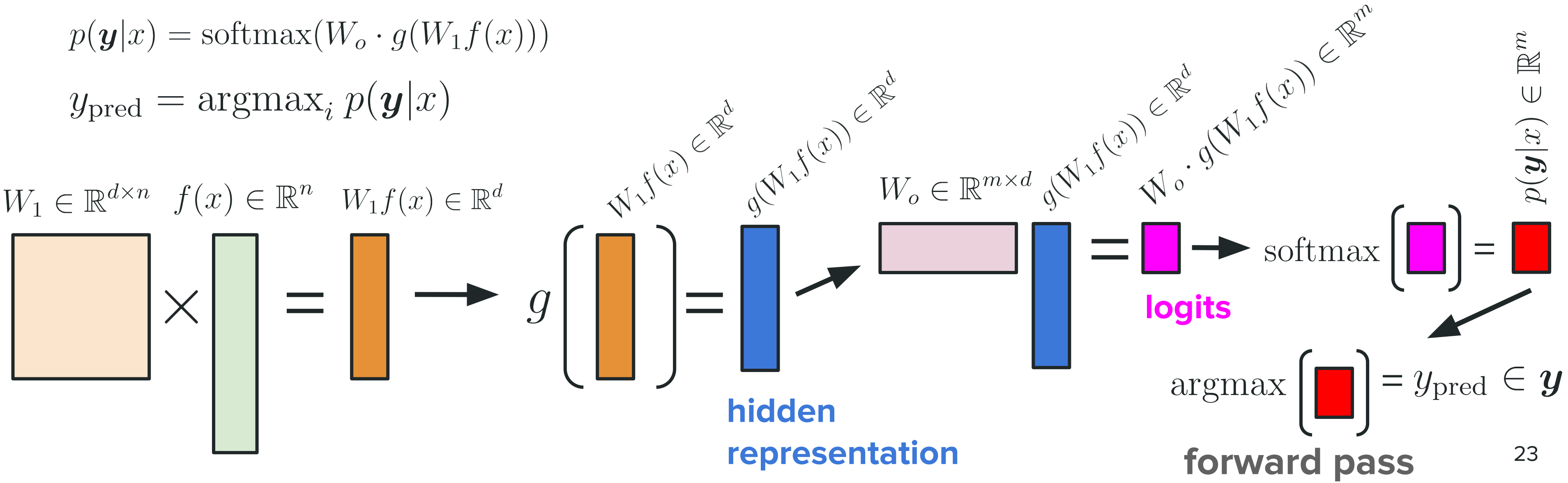
Feedforward neural networks (FNNs)

$$\boldsymbol{y} = [y_1, \ldots, y_m]$$

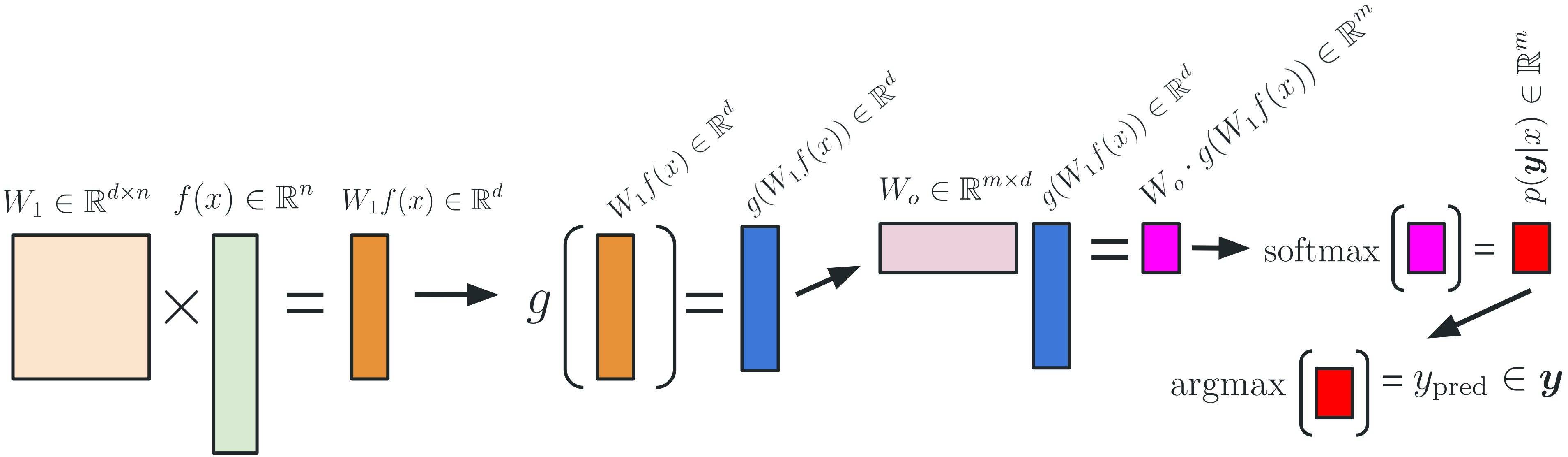
$$W_o = \begin{bmatrix} w_{y_1}^T \\ \vdots \\ w_{y_m}^T \end{bmatrix} \in \mathbb{R}^{m \times d}$$

$$p(\boldsymbol{y}|x) = \text{softmax}(W_o \cdot g(W_1 f(x)))$$

$$y_{\text{pred}} = \text{argmax}_i \; p(\boldsymbol{y}|x)$$

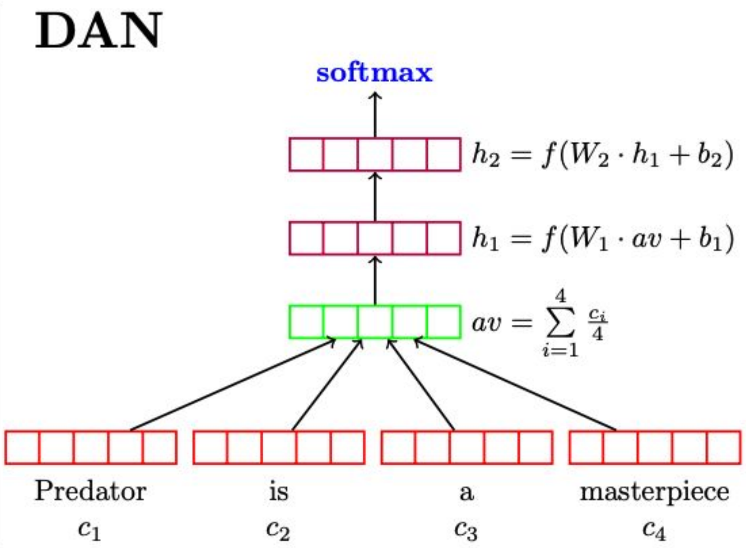


Feedforward neural networks (FNNs) with **Embeddings**



1. Randomly initialize token embeddings & treat them as additional weights to learn (🏋)
2. Initialize token embeddings with existing options (word2vec, Glove; next time!)
   a. Keep them fixed (❄)
   b. Treat them as additional weights to learn (🏋)
3. Represent a given text as the average of embeddings of tokens that appear in the text

Deep Averaging Network (DAN)



[Iyyer et al., 2015]

Goals & Overview of Today's Lecture

Goal: Learn a <u>neural</u> classification function for <u>multiclass</u> classification

- Motivation: Nonlinear transformations
- Multiclass logistic regression
- Feedforward neural network (FFNN)
- Training FFNN
- Practicalities

Training FNNs

$$\mathcal{L}(x, i^*) = \log p(y = i^* | x) = \log(\text{softmax}(W_o \cdot z) \cdot e_{i^*}) \ldots \text{ log likelihood}$$

$$i^* \ldots \text{ index of the gold label}$$

$$e_i \in \mathbb{R}^{1 \times m} \ldots \text{ zero everywhere except at the i-th dimension}$$

$$\boldsymbol{z} = [z_1, \ldots, z_K] \qquad \text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$\mathcal{L}(x, i^*) = W_o z \cdot e_{i^*} - \log \sum_j \exp(W_o z) \cdot e_j$$

Now that we have a loss function, what's next?

Reminder: Gradient Descent: Intuition

Goal: $w^* = \text{argmin}_w \; \text{loss}(w)$

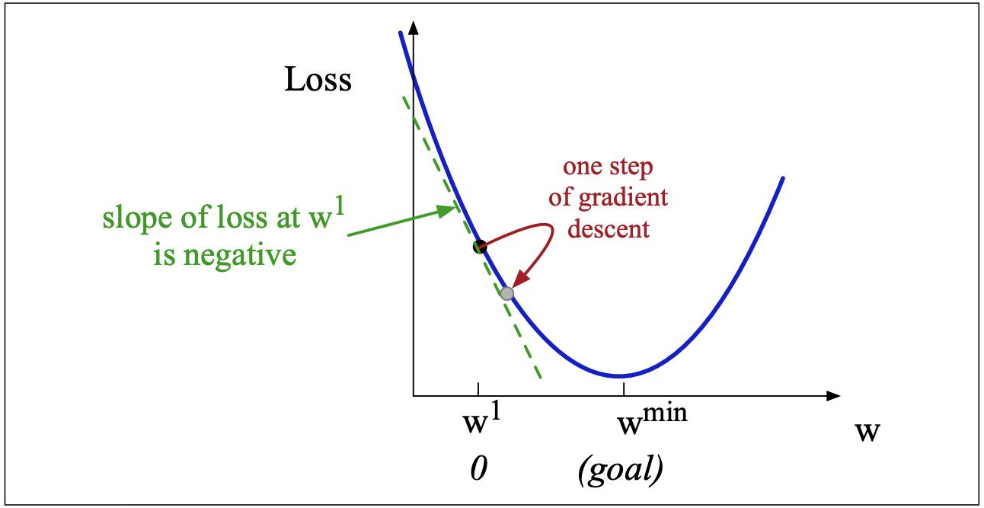


Figure 5.4 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 the value at the second step, and so on.

Source: [Jurafsky & Martin]

Reminder: Gradient Descent: Algorithm

```
init parameters w (e.g. to zeros)
choose a learning rate α (e.g. to 0.1)
for _ in n_epochs:
    for (x_i, y_i) in shuffled_train_data:
        Optionally tweak the learning rate
        Compute gradient ∇w of the loss function L(x_i, y_i, w) with respect to w
        Update the parameters: w = w - α * ∇_w L
```

Training FNNs

$$\mathcal{L}(x, i^*) = \log p(y = i^* | x) = \log(\text{softmax}(W_o \cdot z) \cdot e_{i^*}) \ldots \text{ log likelihood}$$

$$i^* \ldots \text{ index of the gold label}$$

$$e_i \in \mathbb{R}^{1 \times m} \ldots \text{ zero everywhere except at the i-th dimension}$$

$$\boldsymbol{z} = [z_1, \ldots, z_K] \qquad \text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$\mathcal{L}(x, i^*) = W_o z \cdot e_{i^*} - \log \sum_j \exp(W_o z) \cdot e_j$$

Now that we have a loss function, what's next?

The gradient w.r.t. `W_o` can be computed the same as for logistic regression, treating `z` as the feature
But what about the gradient w.r.t. `W_1`?

Backpropagation

But what about the gradient w.r.t. `W_1`?

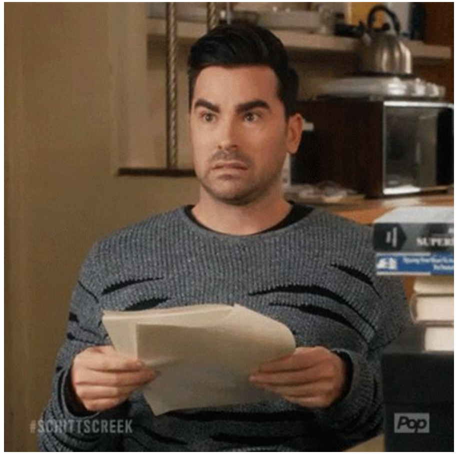
Apply the chain rule

$$\frac{\partial \mathcal{L}(x, i^*)}{\partial W_{1_{i,j}}} = \frac{\partial \mathcal{L}(x, i^*)}{\partial z} \cdot \frac{\partial z}{\partial W_{1_{i,j}}}$$

$$\frac{\partial z}{\partial W_{1_{i,j}}} = \frac{\partial g(a)}{\partial a}$$

$$a = W_1 f(x)$$

Are we going to compute derivatives ourselves every time?



No, we will use frameworks that we will do them for us!

```
import torch
from torchvision.models import resnet18, ResNet18_Weights
model = resnet18(weights=ResNet18_Weights.DEFAULT)
data = torch.rand(1, 3, 64, 64)
labels = torch.rand(1, 1000)
prediction = model(data) # forward pass
loss = (prediction - labels).sum()
loss.backward() # backward pass; autograd calculates and stores the gradients for each model
parameter in the parameter's .grad attribute.
optim = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
optim.step() #gradient descent; optimizer adjusts each parameter by its gradient stored in .grad
```

Goals & Overview of Today's Lecture

Goal: Learn a <u>neural</u> classification function for <u>multiclass</u> classification

- Motivation: Nonlinear transformations
- Multiclass logistic regression
- Feedforward neural network (FFNN)
- Training FFNN
- **Practicalities [next time]**