HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

How did we learn

DeepLearning540 - Lesson 06

Peter Steinbach

Department of Information Services and Computing, Helmholtz-Zentrum Dresden-Rossendorf / 3rd March 2021

Recap: Fitting the Penguin Dataset

inputs

- 3 features (`flipper_length_mm`, `bill_depth_mm`, `bill_length_mm`)
- 3 inputs means $\vec{x} \in \mathbb{R}^3$

outputs

- probability for being of `Adelie` or `not Adelie`
- using one-hot-encoding
- 2 outputs means $\vec{y} \in \mathbb{R}^2$

Note

classification with supervised learning!

Code that needs (some) explanation

```python
# create model
model.compile(loss='categorical_crossentropy',
                     optimizer='sgd',
                     metrics=['accuracy'])
```

supervised learning in one slide[1]

- given m input pairs in a dataset $\mathcal{D} = \{\langle \vec{x}_i, y_i \rangle \dots \}$ with $x \in \mathbb{R}^n$, $y \in \mathbb{R}$
- we would like to train a model f with parameters ϑ such that

$$\vec{y} = f(\vec{x}, \vartheta)$$

- we optimize a loss function $\mathcal{L}$ in such a fashion that

$$\vartheta \approx \arg\min_{\vartheta} \mathcal{L}(\vec{y}^{true}, f(\vec{x}, \vartheta))$$

- during optimisation with gradient descent, parameters ϑ at step s are given by

$$\vartheta_{s+1} = \vartheta_s + \eta \nabla_{\vartheta} L(\vec{y}^{true}, f(\vec{x}, \vartheta_s))$$

[1] credits to Uwe Schmidt

Weight Update Rule

$$\vartheta_{s+1} = \vartheta_s + \eta \nabla_\vartheta L(\vec{y}^{true}, f(\vec{x}, \vartheta_s))$$

gradient descent

free parameter η known as the learning rate

Classic Gradient Descent

- using the full training set
- problem: does not scale with size of dataset

"Online" Gradient Descent

- using the one datum at a time
- problem: expensive to compute, strongly depends on *eta*

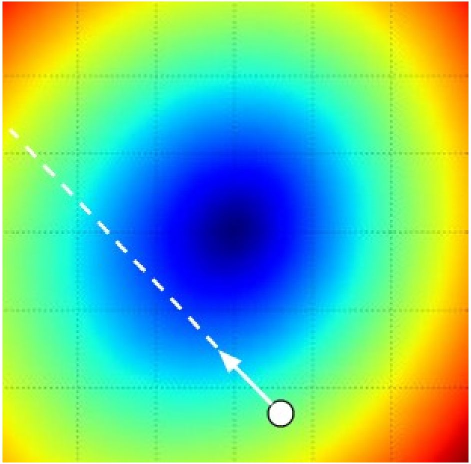
Stochastic gradient descent

$$\vartheta_{s+1} = \vartheta_s + \frac{\eta}{k} \sum_{b=1}^{k} \nabla_\vartheta L_b(\vec{y}_b^{true}, f(\vec{x}_b, \vartheta_s))$$

mini-batch based stochastic gradient descent

- randomly split training set in mini batches, e.g. $b = 32$
- completing one batch is referred to as a `step`
- completing the entire dataset is referred to as an `epoch`
- after each epoch: batches are randomized again

loss landscape



from Stanford CS231n

Loss function: categorical cross-entropy

in general

$$H(p,q) = -\sum_{x\in\mathcal{X}} p(x)\log q(x)$$
$$= H(p) + D_{KL}(p||q)$$

- from information theory
- defined above between two discrete probability distributions p and q
- relates to the Kullback-Leibler divergence

for binary classification

- probability p refers to the true label
- probability q refers to the predicted label (as output of the softmax layer)

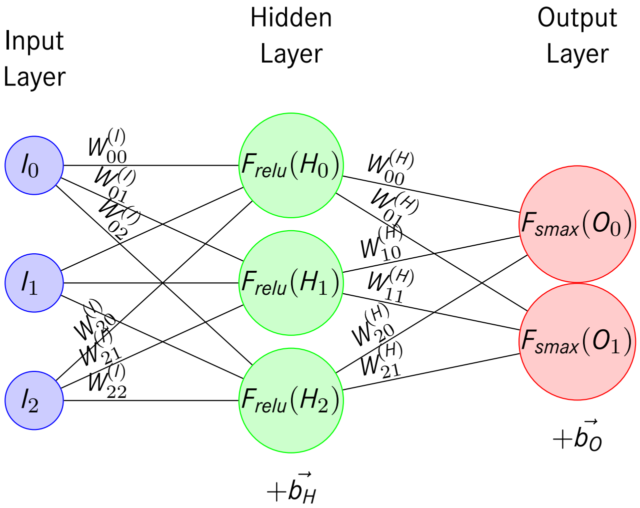
$$L(\vec{y}^{true}, f(\vec{x},\vartheta)) = -\sum_{x\in\mathcal{X}} p(x)\log q(x)$$
$$= -p_{y=1}\log q_{\hat{y}=1}$$
$$- (1-p_{y=1})\log(1-q_{\hat{y}=1})$$

Softmax Activation

$$\sigma(x_i) = \frac{\exp x_i}{\sum_{j=1}^{n} \exp x_j}$$

- mapping of input $x \in \mathbb{R}^n$ to output $\mathbb{R}^n$
- common activation layer for classification networks in last layer
- outputs are $(0, 1)$
- converts input to probability distribution over predicted output classes

A Simple MLP

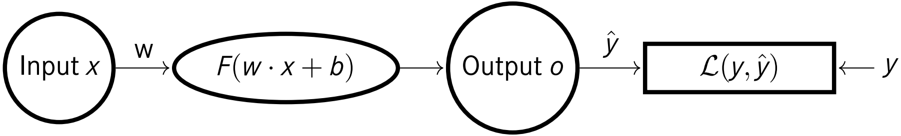


But how to compute the gradient?

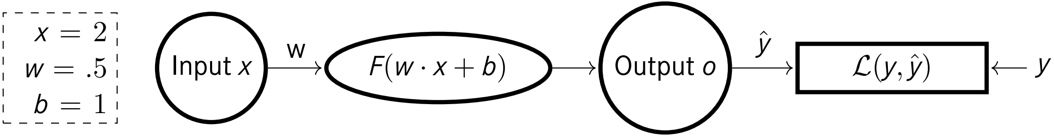
$$\nabla_\vartheta L\left(\vec{y}^{true}, f(\vec{x}, \vartheta)\right)$$

$$(\vartheta \in \{W^{(i)}, b_i\})$$

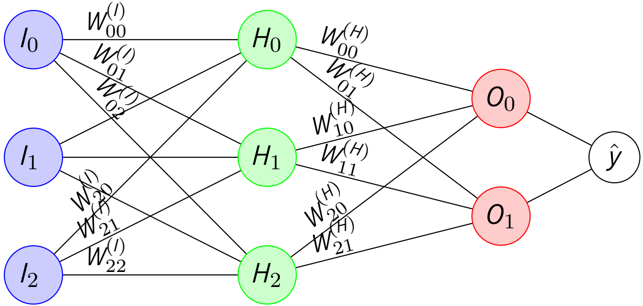
Backpropagation in 1D



Backpropagation in 1D, plugging in numbers



For a single weight



$$\frac{\partial \mathcal{L}}{\partial W_{00}^{I}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_0} \frac{\partial O_0}{\partial W_{00}^{H}} \frac{\partial W_{00}^{H}}{\partial W_{00}^{I}} + \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_1} \frac{\partial O_1}{\partial W_{01}^{H}} \frac{\partial W_{01}^{H}}{\partial W_{00}^{I}}$$

Take Aways

- (mini-batch) stochastic gradient descent (SGD) typically default optimizers for deep learning
- weight update rule governs how to update model parameters
- loss function defines optimisation landscape
- gradient for weight update obtained by backpropagation (chain rule)

Further Reading

- Excellent overview of optimisation algorithms ruder.io/optimizing-gradient-descent
- a classic resource getting introduced to gradient descent at Stanford CS231n course
- cross-entropy well explained for the statistically inclined
- Mathematics for Machine Learning book by Deisenroth, A. Aldo Faisal, and Cheng Soon Ong is an excellent resource to learn about mathematical tools used for ML
- some of the material is based on and inspired by Sebastian Raschka's lecture https://youtu.be/oY6-i2Ybin4