01_Softmax_Regression

December 27, 2023

1 Classification task

- In a regression task, a model must be able to predict the target (output) associated to a new observation (input). Targets may belong to an infinite set of possible values, and not even have known minimum or maximum values.
- A regression model is appropriate to predict a quantity associated to an observation (for example, the price of a house, the position of a particle, or the rating of a movie).
- In a classification task, a model must also be able to predict the target (output) associated to a new observation (input). However, the target belongs to a known finite set of possible values that may not be ordered.
- A classification model is appropriate to predict a class associated to an observation (for example, whether an image depicts a dog or a cat, whether an e-mail is spam, or whether a movie will be liked).
- As we will see, the fact that the targets may not be ordered requires a classification task to be treated differently from a regression task.

1.1 Example

- Suppose that each observation corresponds to an 1024×1024 color image encoded into a vector with $d = 1024 \cdot 1024 \cdot 3$ elements (each pixel has three components: red, green, and blue).
- Suppose that our task is to identify the animal depicted in the image, which is certainly a dog, bird, or cat.
- We could choose a number to represent each class (for example: 1 for dog, 2 for bird, and 3 for cat) and treat this task as a regression task.
- However, our choice of numbers implicitly introduces an order (in our example, we have implicitly biased the model to behave as if dogs were more similar to birds than to cats). In order to see this, recall that a regression loss penalizes a prediction based on its squared distance to the target.

2 One-hot encoding

- One-hot encoding allows dealing with targets (and features) that do not admit an order.

- In an one-hot encoding, each possible value is assigned to a distinct vector where all elements are zero except for one, which is one.
- In the previous example, we could assign $[1, 0, 0]^T$ for dog, $[0, 1, 0]^T$ for bird, and $[0, 0, 1]^T$ for cat. Note that each of these vectors has as many elements as there are classes in the problem.
- Note that the (Euclidean) distance between any pair of these vectors is the same.

3 Softmax regression

- Softmax regression is one of the simplest approaches for classification, just as linear regression is one of the simplest approaches for regression.
- The name softmax regression is somewhat unfortunate, since the softmax regression model is a classification model.
- A softmax regression model receives an observation (input) and predicts a target vector (output).
- A softmax regression model is trained based on a dataset composed of pairs of inputs and outputs, where each output is a vector that results from one-hot encoding a class.
- In our example, a softmax regression model would receive a vector $\mathbf{x}$ with $d = 1024 \cdot 1024 \cdot 3$ elements and predict a vector $\hat{\mathbf{y}}$ with 3 elements.
- The predicted class for a new observation $\mathbf{x}$ would be based on the highest value in the corresponding prediction vector $\hat{\mathbf{y}}$
- In our example, if $\hat{\mathbf{y}} = [0.1, 0.7, 0.2]^T$ is the prediction for observation $\mathbf{x}$, we would say that the model predicts the class "bird".

3.1 Model

- The softmax regression prediction can be computed in two steps, which define the *forward pass*.

3.1.1 First step

- Let q denote the number of classes in the regresion task (in our example, $q = 3$). Suppose that the classes are numbered from 1 to q. How the classes are ordered will not affect the performance of this model.
- The first step computes the so-called logits $\mathbf{o} = [o_1, o_2, \dots, o_q]^T$ for an observation $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$ as

$$\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b},$$

where $\mathbf{W} \in \mathbb{R}^{q \times d}$ is a weight matrix and $\mathbf{b} = [b_1, b_2, \dots, b_q]^T$ is a bias vector.

- The i-th element o_i of the logits $\mathbf{o}$ can be interpreted as a *score* for the i-th class, in the sense that increasing o_i relative to the other elements of $\mathbf{o}$ makes predicting class i more likely.

- The i-th row of $\mathbf{W}$ and the i-th element of $\mathbf{b}$ can be interpreted as parameters associated to the i-th class.

3.1.2 Second step

- The elements of the logits $\mathbf{o}$ are real numbers. However, we know that the elements of the target vector $\mathbf{y}$ are always between zero and one.
- The elements of the target vector $\mathbf{y}$ also sum to one, so that they can be interpreted as probabilities.
- For example, if $\mathbf{y} = [0, 1, 0]^T$ is the target vector for observation $\mathbf{x}$, we may say that the probability of $\mathbf{x}$ depicting a bird is one.
- In order to enable a similar interpretation, the prediction vector $\hat{\mathbf{y}}$ for a new observation $\mathbf{x}$ will be constrained to have elements between zero and one that sum to one.
- One of the simplest (differentiable) functions that can transform the logits $\mathbf{o}$ into a prediction vector $\hat{\mathbf{y}}$ with these properties is the softmax function.
- The prediction vector $\hat{\mathbf{y}}$ for a new observation $\mathbf{x}$ can be computed based on the logits $\mathbf{o}$ as

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{o}),$$

where, for every $j \in \{1, 2, \dots, q\}$,

$$\hat{y}_j = \text{softmax}(\mathbf{o})_j = \frac{e^{o_j}}{e^{o_1} + e^{o_2} + \dots + e^{o_q}}.$$

- Intuitively, e^{o_j} is positive and increases with o_j. By dividing e^{o_j} by $e^{o_1} + e^{o_2} + \dots + e^{o_q}$, the sum $\sum_j \hat{y}_j$ is guaranteed to be 1.
- The predicted class for observation $\mathbf{x}$ is a class j such that $\hat{y}_j = \max_k \hat{y}_k$.

4 Vectorized softmax regression

- Vectorization is the process of writing an algorithm using efficient operations on vectors, matrices, or higher-order tensors.
- Let $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ denote a dataset with n examples, where each observation $\mathbf{x}^{(i)}$ corresponds to a one-hot code $\mathbf{y}^{(i)}$. Let q denote the number of classes in the problem, so that $\mathbf{y}^{(i)}$ has q elements for every i.
- Consider an observation matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ that contains one row for each observation and one column for each feature, so that

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)^T} \\ \mathbf{x}^{(2)^T} \\ \vdots \\ \mathbf{x}^{(n)^T} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \\ x_1^{(n)} & x_2^{(n)} & \dots & x_d^{(n)} \end{bmatrix}.$$

- Consider a target matrix $\mathbf{Y} \in [0,1]^{n \times q}$ that contains one row for each one-hot code, so that

$$\mathbf{Y} = \begin{bmatrix} {\mathbf{y}^{(1)}}^T \\ {\mathbf{y}^{(2)}}^T \\ \vdots \\ {\mathbf{y}^{(n)}}^T \end{bmatrix} = \begin{bmatrix} y_1^{(1)} & y_2^{(1)} & \dots & y_q^{(1)} \\ y_1^{(2)} & y_2^{(2)} & \dots & y_q^{(2)} \\ \vdots & \vdots & \ddots & \\ y_1^{(n)} & y_2^{(n)} & \dots & y_q^{(n)} \end{bmatrix}.$$

- In that case, the logit matrix $\mathbf{O} \in \mathbb{R}^{n \times q}$ may be computed as

$$\mathbf{O} = \mathbf{X}\mathbf{W} + \mathbf{B},$$

  where $\mathbf{W} \in \mathbb{R}^{d \times q}$ is a weight matrix and $\mathbf{B} \in \mathbb{R}^{n \times q}$ is a bias matrix given by replicating a row vector $\mathbf{b}^T = [b_1, b_2, \dots, b_q]$ across n rows.

- Note that we have redefined the weight matrix $\mathbf{W}$ to facilitate vectorization. In this version, the i-the column of $\mathbf{W}$ contains parameters associated to the i-th class.

- In practice, the bias vector $\mathbf{b}$ does not need to be replicated to generate $\mathbf{B}$ if broadcasting is used.

- Finally, the prediction matrix $\hat{\mathbf{Y}}$ that contains one row for each prediction vector is given by

$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{O}),$$

where the softmax function is applied individually to each **row** of the logit matrix $\mathbf{O}$. This involves computing $e^{\mathbf{O}}$ (elementwise exponentiation) and then dividing each of its elements by a corresponding sum across rows.

- The i-th row of the prediction matrix $\hat{\mathbf{Y}}$ contains the (transposed) prediction vector $\hat{\mathbf{y}}^{(i)}$ for the i-th observation $\mathbf{x}^{(i)}$, which can be compared against the target vector $\mathbf{y}^{(i)}$ for the i-th observation.

- Therefore, we seek parameters (weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$) that make $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ as similar as possible.

5 Negative log-likelihood loss function

- In order to train a model for a given dataset, we once again need to be able to measure how well a given model performs on this dataset.

- Because the outputs of the softmax regression model can be intepreted as (conditional) probabilities of classes given an observation, it is natural to seek parameters that would make the dataset as likely as possible. This is called maximum likelihood estimation.

- Using the mean squared error for *linear regression* also has a probabilistic interpretation: if we assume that each target is sampled from a normal distribution whose mean is given by a linear model applied to the corresponding observation, the parameters that minimize the mean squared error make the dataset as likely as possible.

- Let $\hat{\mathbf{y}}$ denote the prediction vector computed by a softmax regression model for an observation $\mathbf{x}$ with target vector $\mathbf{y}$. The probability assigned to the correct class by the model is given by

$$\hat{y}_1^{y_1} \hat{y}_2^{y_2} \cdots \hat{y}_q^{y_q} = \prod_{k=1}^{q} \hat{y}_k^{y_k},$$

  since $y_k = 1$ if and only if the correct class is k, and $y_k = 0$ otherwise.

- Maximum likelihood estimation suggests maximizing the probability that each observation $\mathbf{x}$ in the dataset is assigned to the correct class by the model.

- Maximizing the previous expression is equivalent to maximizing the so-called log-likelihood given by

$$\log \left[\prod_{k=1}^{q} \hat{y}_k^{y_k} \right] = \sum_{k=1}^{q} y_k \log \hat{y}_k,$$

  since $\hat{y}_k > 0$ for every k and log (implicitly base e) is an increasing function.

- For the i-th example in the dataset, maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood function $l^{(i)}$ given by

$$l^{(i)}(\mathbf{W}, \mathbf{b}) = -\sum_{k=1}^{q} y_k^{(i)} \log \hat{y}_k^{(i)}.$$

- Note that the model (weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$) affects the negative log-likelihood for the i-th example $l^{(i)}(\mathbf{W}, \mathbf{b})$ through the prediction vector $\hat{\mathbf{y}}^{(\mathbf{i})} = [\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \dots, \hat{y}_q^{(i)}]^T$.

- Note that if the i-th observation $\mathbf{x}^{(i)}$ belongs to class k, then $l^{(i)}(\mathbf{W}, \mathbf{b}) = -y_k \log \hat{y}_k = -\log \hat{y}_k$, since all other terms in the summation above will be zero. Therefore, the loss $l^{(i)}(\mathbf{W}, \mathbf{b})$ decreases as $\hat{y}_k$ increases.

6 Cross-Entropy Loss

- Under additional assumptions, the cross-entropy loss is an appropriate way to combine the negative log-likelihood loss for each example into a single measure of how well a model performs in the training dataset.

- The cross-entropy loss C for a softmax regression model defined by a weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$ is given by

$$C = \sum_{i=1}^{n} l^{(i)}(\mathbf{W}, \mathbf{b}) = -\sum_{i=1}^{n} \sum_{k=1}^{q} y_k^{(i)} \log \hat{y}_k^{(i)}.$$

- In order to employ gradient descent, we must compute the gradient ∇C of the loss C with respect to the model parameters $\mathbf{W}$ and $\mathbf{b}$.

- We will leave that task to our automatic differentiation tools, although it is not that difficult to find an analytic expression for this gradient.

- In order to imagine the gradient ∇C as a vector, you can imagine that the weight matrix $\mathbf{W}$ is *flattened* into a vector and concatenated with the bias vector $\mathbf{b}$ into a long parameter vector with $d \cdot q + q$ elements.

7 Prediction and Evaluation

- After the model is trained, we can predict the probability of each class given a new observation.
- We can assign the class with the highest predicted probability to a new observation.
- A prediction is correct if it matches the actual class.
- The *accuracy* of a model is defined as the fraction of examples in a given dataset for which the model makes correct predictions (100% accuracy means always correct, 0% accuracy means never correct).
- The accuracy of the model on a *validation dataset* (data not used for training) can be used to assess generalization.

8 Recommended reading

- Dive into Deep Learning: Chapters 3.4, 3.5, 3.6, and 3.7.

9 [Storing this notebook as a pdf]

[ ]:
```
%%capture
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic

# Set the path to this notebook below (add \ before spaces). The output `pdf`␣
↪will be stored in the corresponding folder.
!jupyter nbconvert --to pdf /content/gdrive/My\ Drive/Colab\ Notebooks/nndl/
↪week_04/lecture/01_Softmax_Regression.ipynb

# If having issues, save this notebook (File > Save) and restart the session␣
↪(Runtime > Restart session) before running this cell. To debug, remove the␣
↪first line (`%%capture`).
```