Softmax Regression

- Regression is the hammer we reach for when we want to answer *how much?* or *how many?* questions.
    - If you want to predict the number of pounds (price) at which a house will be sold, or the number of days that a patient will remain hospitalized before being discharged, then you are probably looking for a regression model.
- In many cases, we are interested in *classification*: asking not "how much" but "which one":
    - Does this email belong in the spam folder or the inbox?
    - Does this image depict a donkey, a dog, a cat, or a rooster?

Classification Problem

- **Simple Example:** a simple image classification problem.
    - Each input is 2×2 grayscale image.
        - Each pixel value is a single scalar, giving us four features x_1, x_2, x_3, x_4.
    - We also assume that each image belongs to one of the "cat", "chicken", and "dog" classes

Classification Problem

Labels as one-hot encoding

- We have to choose how to represent the labels.
- We have the following obvious choice: choose $y \in \{1, 2, 3\}$, for $\{\text{dog}, \text{cat}, \text{chicken}\}$ respectively.
    - This introduces ordering which is not desired.
- Better choice: the **one-hot encoding**.
    - A vector with as many components as the number of classes.
    - The component corresponding to a particular instance's class is set to 1 and all other components are set to 0.
- In our case, a label y would be a three-dimensional vector, with $(1, 0, 0)$ corresponding to "cat", $(0, 1, 0)$ to "chicken", and $(0, 0, 1)$ to "dog":

$$y \in \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}.$$

Network Architecture

- We need a model with multiple outputs, one per element of the one-hot encoding (i.e. one per class), i.e. 3 outputs in total.
- For a training example, we will pass the features through the model and get as output 3 numbers.
    - These will not be exactly zeros and ones.

- The location of the maximum indicates the correct class.

- Since we have 4 features we need, *for each output*, 4 weights and 1 bias which will be used to linearly combine the input features to give us an output.

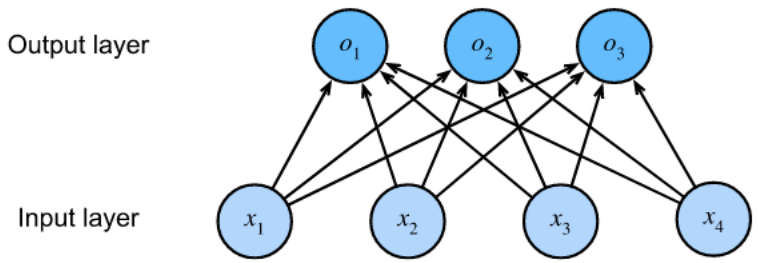
- We compute 3 outputs or *logits*, o_1, o_2, and o_3, for each input:

$$
\begin{aligned}
o_1 &= x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14} + b_1, \\
o_2 &= x_1 w_{21} + x_2 w_{22} + x_3 w_{23} + x_4 w_{24} + b_2, \\
o_3 &= x_1 w_{31} + x_2 w_{32} + x_3 w_{33} + x_4 w_{34} + b_3.
\end{aligned}
$$

- To express the model more compactly, we can use linear algebra: $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$,
    - All features of a given example are gathered in vector $\mathbf{x}$
    - All weights are gathered into a 3×4 matrix $\mathbf{W}$
    - All biases are gathered in vector $\mathbf{b}$
    - All outputs are gathered in vector $\mathbf{o}$

Network Architecture

- We can depict this calculation with a neural network diagram:



- Just as in linear regression, softmax regression is also a single-layer neural network.
  - Since the calculation of each output, o_1, o_2, and o_3, depends on all inputs, x_1, x_2, x_3, and x_4, softmax regression can also be implemented with a fully-connected layer.

Softmax Operation

- For training the model, our approach will be to interpret the outputs of our model as probabilities.
    - We will then optimize the model parameters to produce probabilities that maximize the likelihood of the observed data.
- One option to consider is to interpret the logits o as probabilities. However:
    - Nothing constrains these numbers to sum to 1.
    - Logits o can take negative values.
        - Both these violate the basic axioms of probability.
- The **Softmax function** can be used to transform our logits such that they become nonnegative and sum to 1.
    - The model remains differentiable.

Softmax Operation

- First exponentiate each logit (ensuring non-negativity) and then divide by their sum (ensuring they sum to 1):

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \quad \text{where} \quad \hat{y}_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)}.$$

- It is easy to see $\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$ with $0 \leq \hat{y}_j \leq 1$ for all j.
  - Thus, $\hat{\mathbf{y}}$ is a proper probability distribution whose element values can be interpreted accordingly.
  - Softmax operation does not change the ordering among the logits $\mathbf{o}$,
- During prediction we can still pick out the most likely class by

$$\underset{j}{\mathrm{argmax}}\ \hat{y}_j = \underset{j}{\mathrm{argmax}}\ o_j.$$

- For example, if $\hat{y}_1$, $\hat{y}_2$, and $\hat{y}_3$ are 0.1, 0.8, and 0.1, then we predict category 2, which (in our example) represents "chicken".

- Although softmax is a nonlinear function, the outputs of softmax regression are still *determined* by a linear transformation of input features; thus, softmax regression is a linear model.

Vectorization for Minibatches

- We typically carry out training in minibatches of data.
- Assume minibatch $\mathbf{X}$ of examples with feature dimensionality d and batch size n. Moreover, assume q classes in the output. Then:
  - $\mathbf{X}$ are in $\mathbb{R}^{n \times d}$, $\mathbf{W} \in \mathbb{R}^{d \times q}$, and $\mathbf{b} \in \mathbb{R}^{1 \times q}$.

$$\mathbf{O} = \mathbf{X}\mathbf{W} + \mathbf{b},$$
$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{O}).$$

- Since each row in $\mathbf{X}$ represents a data point, the softmax operation can be computed *rowwise*:
  - for each row of $\mathbf{O}$, exponentiate all entries and then normalize them by the sum.

Loss Function

- For training the model, we need a loss function to measure the quality of our predicted probabilities.
- We will rely on Maximum Likelihood estimation
    - the very same concept encountered when providing a probabilistic justification for the mean squared error objective in linear regression

Loss Function

- The softmax function gives us a vector $\hat{\mathbf{y}}$, which can be interpreted as the estimated (conditional) probabilities of each class given an input $\mathbf{x}$.
    - E.g., $\hat{y}_1 = P(y = \text{cat} \mid \mathbf{x})$.
- Suppose that the entire dataset $\{\mathbf{X}, \mathbf{Y}\}$ has n examples,
- i—th example has feature vector $\mathbf{x}^{(i)}$ and a one-hot label vector $\mathbf{y}^{(i)}$.
    - If the correct class for the i—th example is the k—th class, then the k—th element of $\mathbf{y}^{(i)}$, denoted as $y_k^{(i)}$ will be 1 and all other elements will be 0.
    - According to Maximum Likelihood we want to maximize the probability that the model assigns the correct class to the i—th example:

$$P(y = y_k^{(i)} \mid \mathbf{x}^{(i)}) = \hat{y}_k$$

Loss Function

- This can be also written as:

$$P(y = y_k^{(i)} \mid \mathbf{x}^{(i)}) = P(y = y_k^{(i)} \mid \mathbf{x}^{(i)})^{y_k^{(i)}}$$
$$= \prod_{j=1}^{q} P(y = y_j^{(i)} \mid \mathbf{x}^{(i)})^{y_j^{(i)}}$$
$$= \prod_{j=1}^{q} \hat{y}_j^{y_j^{(i)}}$$

- This is equivalent to minimizing the negative log-likelihood:

$$l(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}) = -\sum_{j=1}^{q} y_j^{(i)} \log \hat{y}_j^{(i)}$$

Cross-Entropy Loss

- The total loss is calcluated over all examples:

$$CE = \sum_{i=1}^{n} l(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})$$

- This is called **Cross-Entopy Loss**.
- Since $\mathbf{y}$ is a one-hot vector of length q, the sum over all its coordinates j vanishes for all but one term.
- Since all $\hat{y}_j$ are predicted probabilities, their logarithm is never larger than 0.
- Consequently, the loss function cannot be minimized any further if we correctly predict the actual label with *certainty*, i.e., if the predicted probability $P(\mathbf{y} \mid \mathbf{x}) = 1$ for the actual label $\mathbf{y}$.
- Note that this may also not be possible when the input features are not sufficiently informative to classify every example perfectly.

Cross-Entropy Loss Derivatives

- Plugging softmax output into the definition of the loss:

$$\begin{aligned} l(\mathbf{y}, \hat{\mathbf{y}}) &= -\sum_{j=1}^{q} y_j \log \frac{\exp(o_j)}{\sum_{k=1}^{q} \exp(o_k)} \\ &= \sum_{j=1}^{q} y_j \log \sum_{k=1}^{q} \exp(o_k) - \sum_{j=1}^{q} y_j o_j \\ &= \log \sum_{k=1}^{q} \exp(o_k) - \sum_{j=1}^{q} y_j o_j. \end{aligned}$$

- Consider now the derivative with respect to any logit o_j:

$$\partial_{o_j} l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\exp(o_j)}{\sum_{k=1}^{q} \exp(o_k)} - y_j = \text{softmax}(\mathbf{o})_j - y_j.$$

- The derivative is the difference between the probability produced by our model, and the elements in the one-hot label vector.
- Very similar to what we saw in regression, where the gradient was the difference between the observation y and estimate $\hat{y}$.
- This fact makes computing gradients easy in practice.

Model Prediction and Evaluation

- After training the model, given any example features, we can predict the probability of each output class.
- We use the class with the highest predicted probability as the output class.
- The prediction is correct if it is consistent with the actual class (label).

Screen Shot 2021-02-15 at 12.10.03