T05460301-01

Deep Learning (Fall 2023)

Ikbeom Jang

ijang@hufs.ac.kr
CES HUFS

Linear Neural Networks

- Softmax Regression

Contents

Classification vs. Regression

- Regression is used when we want to answer *how much*? or *how many*? questions.
  - Predict the number of dollars (price) at which a house will be sold.
  - Predict the number of wins a baseball team might have.
  - Predict the number of days that a patient will remain hospitalized before being discharged.

- In practice, we are more often interested in *classification*: asking not “how much” but “which one”:
  - Does this email belong in the spam folder or the inbox?
  - Is this customer more likely to sign up or not to sign up for a subscription service?
  - Does this image depict a donkey, a dog, a cat, or a rooster?
  - Which movie is Aston most likely to watch next?

- The word *classification* to describe two subtly different problems:
  - Those where we are interested only in hard assignments of examples to categories (classes)
  - Those where we wish to make soft assignments, i.e., to assess the probability that each category applies.

Classification Problem

- A simple image classification problem:
  - Each input consists of a 2×2 grayscale image.
  - We represent each pixel value with a single scalar, giving us four features x_1, x_2, x_3, x_4.
  - We assume that each image belongs to one among the categories "cat", "chicken", and "dog".

- Next, choose how to represent the labels:
  - If the categories had some natural ordering among them
    - Cast this problem as regression and keep the labels as in their format.

  - If the categories do not have orderings among them.
    - Use *one-hot encoding,* which is a vector with as many components as we have categories.
    - The component corresponding to particular instance's category 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 $(1,0,0)$ = "cat", $(0,1,0)$ = "chicken", and $(0,0,1)$ = "dog":

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

Network Architecture

- To estimate the conditional probabilities associated with all the possible classes, we need a model with multiple outputs, one per class.

- To address classification with linear models, we will need as many affine functions as we have outputs.
    - Each output will correspond to its own affine function.

- In our case, since we have 4 features and 3 possible output categories, we will need
    - 12 scalars to represent the weights (w with subscripts),
    - 3 scalars to represent the biases (b with subscripts).

- We compute these three logits, o_1, o_2, and o_3, for each input:

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

Network Architecture

- We can depict the calculation in (3.4.2) with the neural network diagram shown in Fig. 3.4.1.
    - Softmax regression is a single-layer neural network.
    - The output layer of softmax regression can be described as fully-connected layer.

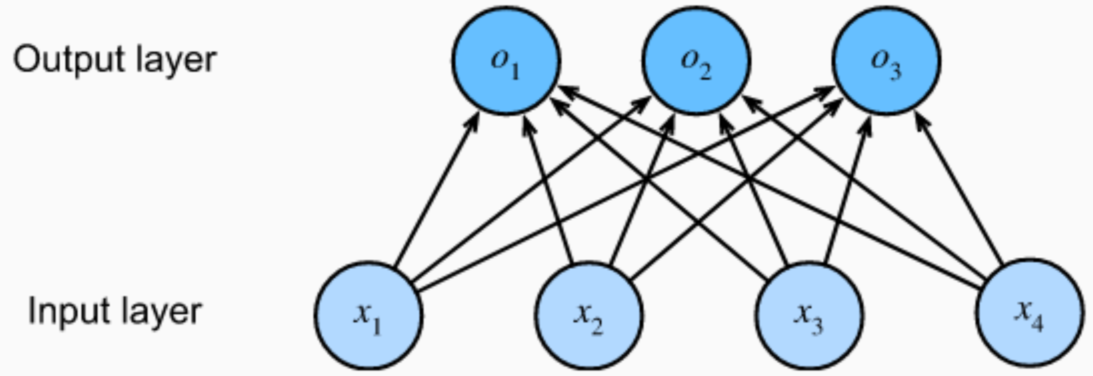


Fig. 3.4.1 Softmax regression is a single-layer neural network.

- We can use linear algebra notation: $\boldsymbol{o} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$
    - This form better suited both for mathematics, and for writing code.

Softmax Operation

- The main approach that we are going to take here is to interpret the outputs of our model as probabilities.
  - We will optimize our parameters to produce probabilities that maximize the likelihood of the observed data.
  - To generate predictions, we will set a threshold, for example, choosing the label with the maximum predicted probabilities.

- Put formally, we would like any output $\hat{y}_j$ to be interpreted as the probability that a given item belongs to class j.
- Then we can choose the class with the largest output value as our prediction $argmax_j \ y_j$.
  - For example, if $\hat{y}_1$, $\hat{y}_2$, and$\hat{y}_3$ are 0.1, 0.8, and 0.1, respectively, then we predict category 2, which (in our example) represents "chicken".

- We CANNOT interpret the logits o directly as our outputs of interest because:
  - Nothing constrains these numbers to sum to 1.
  - Depending on the inputs, they can take negative values. These violate basic axioms of probability.

Softmax Operation

- To interpret our outputs as probabilities,
  - We must guarantee that (even on new data), they will be nonnegative and sum up to 1.
  - We need a training objective that encourages the model to estimate faithfully probabilities.

- Of all instances when a classifier outputs 0.5, we hope that half of those examples will actually belong to the predicted class.
  - This is a property called calibration.

- To transform our logits to become nonnegative and sum to 1, while requiring that the model remains differentiable,
  - We first exponentiate each logit (ensuring non-negativity)
  - Then divide by their sum (ensuring that they sum to 1)

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \quad \text{where } \hat{y}_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)} \tag{3.4.3}$$

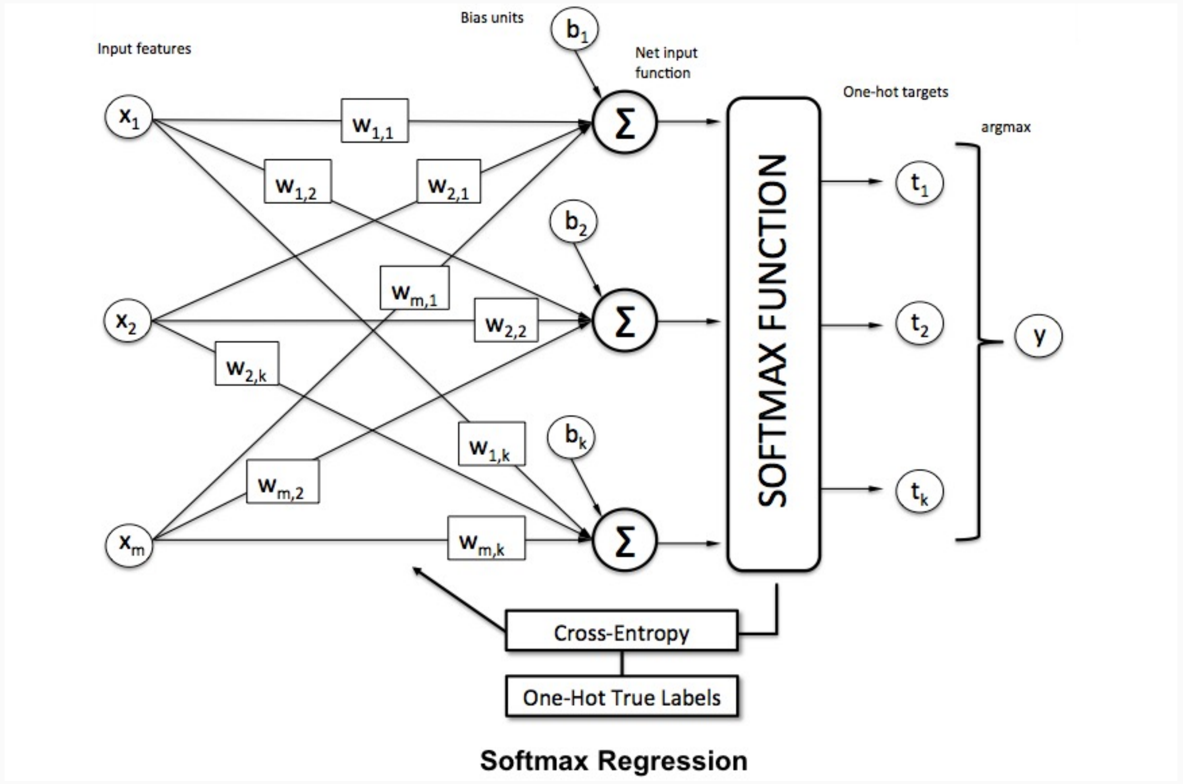
  - It is easy to see $\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$ with $0 \leq \hat{y}_j \leq 1$ for all j.

Softmax Operation

- The softmax operation does not change the ordering among the logits o.
  - Therefore, during prediction we can still pick out the most likely class by

$$\operatorname{argmax}_j \hat{y}_j = \operatorname{argmax}_j o_j \qquad (3.4.4)$$

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



Softmax Regression

https://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html

Vectorization for Minibatches

- To improve computational efficiency and take advantage of GPUs, we carry out vector calculations for minibatches of data.

- Assume we are given a minibatch $\boldsymbol{X}$ of examples with feature dimensionality (number of inputs) d and batch size n.
- Moreover, assume that we have q categories in the output. Then we have:
  - The minibatch features $\boldsymbol{X}$ are in $\mathbb{R}^{n \times d}$.
  - The weights $\boldsymbol{W} \in \mathbb{R}^{d \times q}$.
  - The bias satisfies $b \in \mathbb{R}^{1 \times q}$.

$$\begin{aligned} \boldsymbol{O} &= \boldsymbol{XW} + \boldsymbol{b}, \\ \hat{\boldsymbol{Y}} &= \operatorname{softmax}(\mathbf{O}) \end{aligned} \tag{3.4.5}$$

- This accelerates the dominant operation into a matrix-matrix product $\boldsymbol{XW}$ vs. the matrix-vector products we would be executing if we processed one example at a time.

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

- Triggering broadcasting during the summation $\boldsymbol{XW} + \boldsymbol{b}$, both the minibatch logits $\boldsymbol{O}$ and output probabilities $\hat{\boldsymbol{Y}}$ are $n \times q$ matrices.

Loss Function

- Next, we need a loss function to measure the quality of our predicted probabilities.

- We will rely on maximum likelihood estimation
  - The very same concept that we encountered when providing a probabilistic justification for the mean squared error objective in linear regression

Loss Function: Log-Likelihood

- The softmax function gives us a vector $\hat{y}$, which we can interpret as estimated conditional probabilities of each class given any input x, $e.g., \hat{y}_1 = P(y = cat \mid \boldsymbol{x})$.

- Suppose that the entire dataset $\{\boldsymbol{X}, \boldsymbol{Y}\}$ has n examples, where the example indexed by i consists of a feature vector $x^{(i)}$and a one-hot label vector $y^{(i)}$.
- We can compare the estimates with reality by checking how probable the actual classes are according to our model, given the features:

$$P(\boldsymbol{Y} \mid \boldsymbol{X}) = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}) \tag{3.4.6}$$

- According to maximum likelihood estimation, we maximize $P(\boldsymbol{Y} \mid \boldsymbol{X})$, which is equivalent to minimizing the negative log-likelihood:

$$-logP(\boldsymbol{Y} \mid \boldsymbol{X}) = \sum_{i=1}^{n} -logP\left(y^{(i)} \mid x^{(i)} \right) = \sum_{i=1}^{n} l(y^{(i)}, \hat{y}^{(i)}), \tag{3.4.7}$$

where for any pair of label y and model prediction $\hat{y}$ over q classes, the loss function l is

$$l(y, \hat{y}) = -\sum_{j=1}^{q} y_j \log \hat{y}_j \tag{3.4.8}$$

Loss Function: Log-Likelihood

- The loss function in (3.4.8) is commonly called the *cross-entropy loss*.
- Since 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 $P(\boldsymbol{y} \mid \boldsymbol{x}) = 1$ for the actual label $\boldsymbol{y}$.
    - Note that this is often impossible. For example, there might be label noise in the dataset.
    - It may also not be possible when the input features are not sufficiently informative to classify every example perfectly.

Loss Function: Softmax and Derivatives

- Plugging (3.4.3) into the definition of the loss in (3.4.8) and using the definition of the softmax we obtain:

$$\begin{aligned} l(\mathbf{y}, \hat{\mathbf{y}}) &= -\sum_{j=1}^{q} y_j \log \frac{\exp(o_j)}{\sum_{k=1}^{q} \exp(o_k)} \\ &= \sum_{j=1}^{q} y_j \log \sum_{k=1}^{q} \exp(o_k) - \sum_{j=1}^{q} y_j o_j \\ &= \log \sum_{k=1}^{q} \exp(o_k) - \sum_{j=1}^{q} y_j o_j \end{aligned} \tag{3.4.9}$$

- Consider the derivative with respect to any logit o_j. We get

$$\partial_{o_j} l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\exp(o_j)}{\sum_{k=1}^{q} \exp(o_k)} - y_j = \operatorname{softmax}(\mathbf{o})_j - y_j \tag{3.4.10}$$

In other words, the derivative is the difference between the probability assigned by our model, as expressed by the softmax operation, and what actually happened, as expressed by elements in the one-hot label vector.

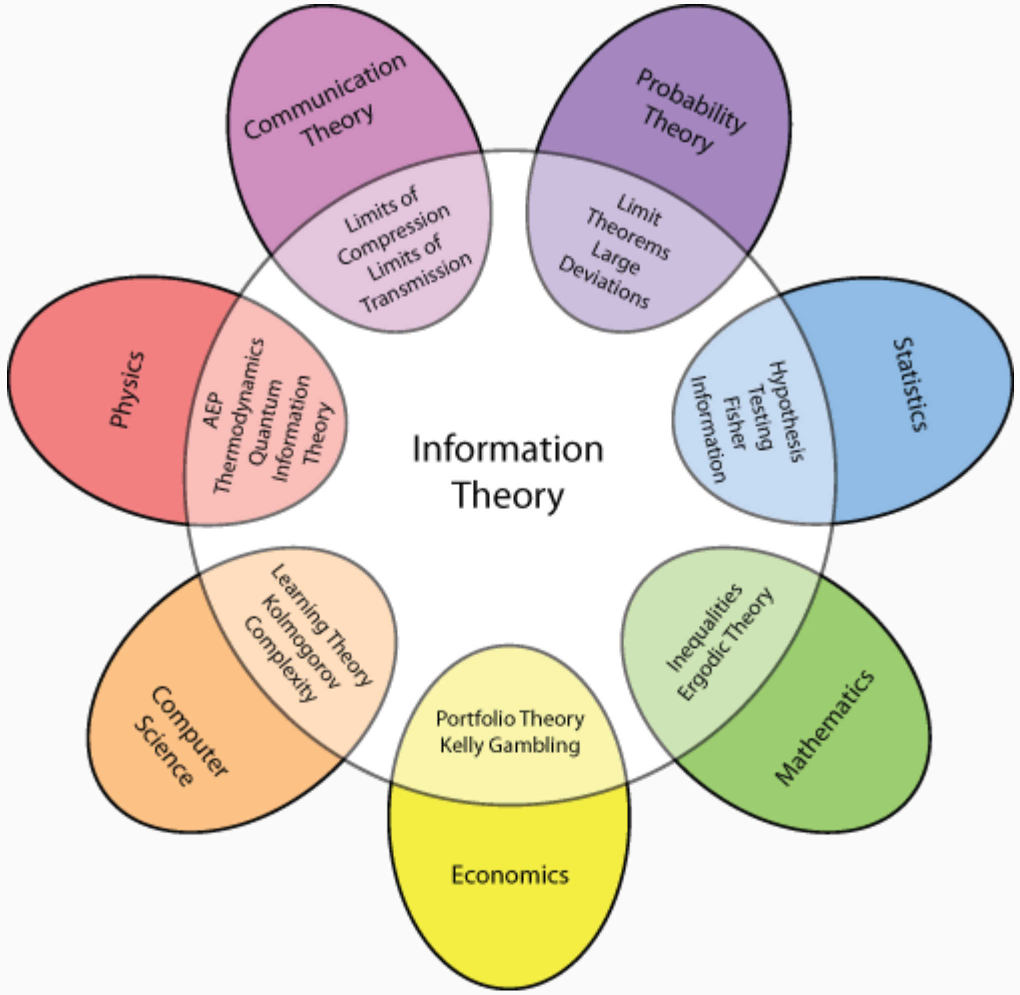
Loss Function: Cross-Entropy Loss

- Consider the case where we observe not just a single outcome but an entire distribution over outcomes.
  - We can use the same representation as before for the label y.
  - The math that we used previously to define the loss l in (3.4.8) still works out fine, just that the interpretation is slightly more general.
  - The only difference is that rather than a vector containing only binary entries, say (0,0,1), we now have a generic probability vector, say (0.1,0.2,0.7) .

- The expected value of the loss for a distribution over labels is called the *cross-entropy loss* and it is one of the most commonly used losses for classification problems.

Information Theory Basics

- Information theory deals with the problem of encoding, decoding, transmitting, and manipulating information (also known as data) in as concise form as possible.


Communication Theory
Limits of Compression Limits of Transmission
Probability Theory
Limit Theorems Large Deviations
Statistics
Hypothesis Testing Fisher Information
Mathematics
Inequalities Ergodic Theory
Economics
Portfolio Theory Kelly Gambling
Computer Science
Learning Theory Kolmogorov Complexity
Physics
AEP Thermodynamics Quantum Information Theory
Information Theory

https://medium.com/machine-learning-bootcamp/demystifying-information-theory-e21f3af09455

Information Theory Basics: Entropy

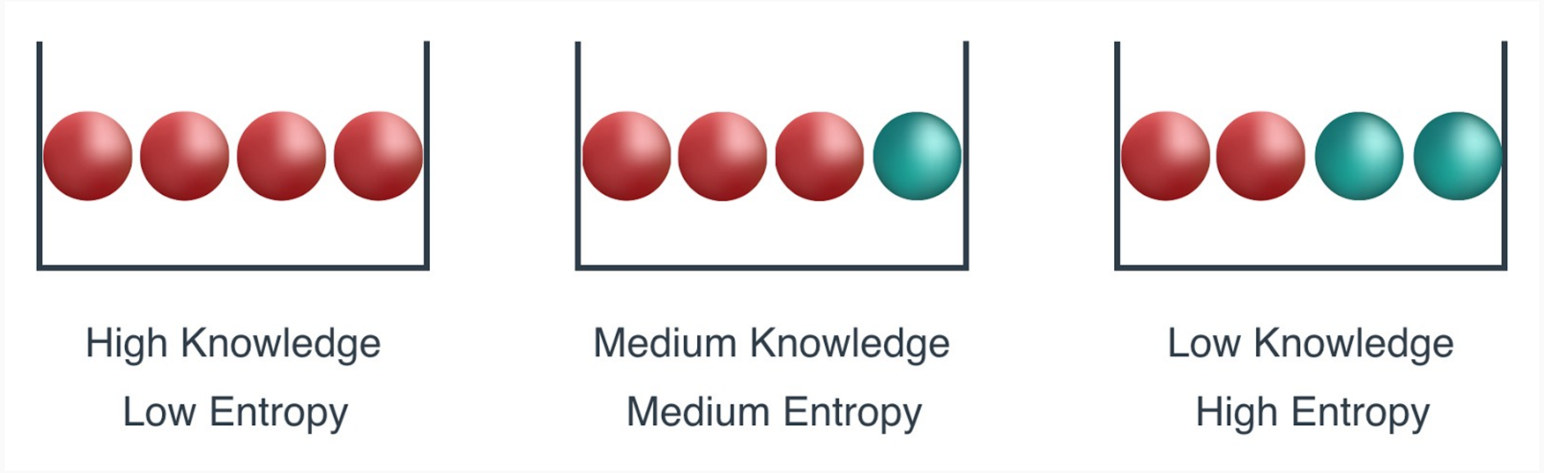
- The central idea in information theory is to quantify the information content in data.
  - This quantity places a hard limit on our ability to compress the data.
  - This quantity is called the entropy of a distribution P , and it is captured by the following equation:

$$H[P] = \sum_j -P(j) \log P(j) \tag{3.4.11}$$

- One of the fundamental theorems of information theory states that in order to encode data drawn randomly from the distribution P, we need at least $H[P]$ "nats" to encode it.
  - A "nat" is the equivalent of bit but when using a code with base e rather than one with base 2.
  - Thus, one nat is $\frac{1}{\log(2)} \approx 1.44$ bit.

Information Theory Basics: Surprisal

- What compression has to do with prediction? Imagine that we have a stream of data that we want to compress.
  - If it is always easy for us to predict the next token, then this data is easy to compress!
  - If we cannot perfectly predict every event, then we might sometimes be surprised. Our surprise is greater when we assigned an event lower probability.



https://medium.com/udacity/shannon-entropy-information-gain-and-picking-balls-from-buckets-5810d35d54b4

- Claude Shannon settled on $\log \frac{1}{P(j)} = -\log P(j)$ to quantify one's surprisal at observing an event j having assigned it a (subjective) probability $P(j)$.

- The entropy defined in (3.4.11) is the expected surprisal when one assigned the correct probabilities that truly match the data-generating process.

Information Theory Basics: Cross-Entropy Revisited

- If the entropy is level of surprise experienced by someone who knows the true probability.
  - So, what is cross-entropy?

- The cross-entropy from P to Q, denoted $H(P, Q)$, is the expected surprisal of an observer with subjective probabilities Q upon seeing data that were actually generated according to probabilities P.
  - The lowest possible cross-entropy is achieved when $P = Q$. In this case, the cross-entropy from P to Q is $H(P, P) = H(P)$.

- In short, we can think of the cross-entropy classification objective in two ways:
  1. maximizing the likelihood of the observed data;
  2. minimizing our surprisal (and thus the number of bits) required to communicate the labels.

Model Prediction and Evaluation

- After training the softmax regression 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).

- We will use accuracy to evaluate the model's performance.
  - This is equal to the ratio between the number of correct predictions and the total number of predictions.

Summary

- The softmax operation takes a vector and maps it into probabilities.
- Softmax regression applies to classification problems. It uses the probability distribution of the output class in the softmax operation.
- Cross-entropy is a good measure of the difference between two probability distributions.
  It measures the number of bits needed to encode the data given our model.