CMPT 825 NLP - Spring 2020 - Midterm Exam

1. Write your name, your SFU username, the class name and your student ID number on the answer sheet.
2. Put away all electronics and keep your backpack/bag away from you.
3. Keep your student ID with you. Show it to us when handing in your exam.
4. The exam is closed-book but you can bring a single letter-sized double-sided crib sheet of your own hand-written information into the exam.
5. You must hand in the crib sheet and question booklet along with your answer booklet at the end of the exam.
6. You must write with a pen. No pencils.

(1) (10pts) **Language Modeling**

Consider a language model over character sequences that computes the probability of a word based on the characters in that word, so if word $w = c_0, c_1, \ldots, c_n$ then $P(w) = P(c_0, \ldots, c_n)$. Let us assume that the language model is defined as a bigram character model $P(c_i \mid c_{i-1})$ where

$$P(c_0, \ldots, c_n) = \prod_{i=1,2,\ldots,n} P(c_i \mid c_{i-1}) \tag{1}$$

For convenience we assume that we have explicit word boundaries: $c_0 =$ <s> and $c_n =$ </s> where <s> is the *begin sentence marker* and </s> is the *end of sentence marker*.

Based on this model, for the English word *booking* the probability would be computed as:

$$P(booking) = P(b \mid \texttt{<s>}) \times P(o \mid b) \times P(o \mid o) \times P(k \mid o) \times P(i \mid k) \times P(n \mid i) \times P(g \mid n) \times P(\texttt{</s>} \mid g)$$

The inflection *ing* is a suffix and is generated after the stem *book* with probability

$$P(ing) = P(i \mid k) \times P(n \mid i) \times P(g \mid n) \times P(\texttt{</s>} \mid g)$$

In Semitic languages, like Arabic and Hebrew, the process of inflection works a bit differently. In Arabic, for a word like *kitab* the stem would be *k-t-b* where the place-holders '-' for inflection characters have been added for convenience. We will assume that each word is made up of a sequence of consonant-vowel sequences CVCVCV… and the vowels always form the inflection.

a. (4pts) Provide the definition of an n-gram model that will compute the probability for the word *kitab* and *k-t-b* as follows:

$$P(kitab) = P(k \mid \texttt{<s>}) \times P(t \mid k) \times P(b \mid t) \times P(i \mid b) \times P(a \mid i) \times P(\texttt{</s>} \mid a)$$

$$P(k\text{-}t\text{-}b) = P(k \mid \texttt{<s>}) \times P(t \mid k) \times P(b \mid t) \times P(\texttt{-} \mid b) \times P(\texttt{-} \mid \texttt{-}) \times P(\texttt{</s>} \mid \texttt{-})$$

Write down the equation for this n-gram model in the same mathematical notation as equation (1).

Answer:

$$P(c_0, \ldots, c_n) = \begin{cases} \prod_{i=1}^{n} P(c_i \mid c_{i-1}) & \text{if } n \leq 3 \\ \Big(P(c_1 \mid c_0) \times \prod_{i=3,5,\ldots}^{\ell} P(c_i \mid c_{i-2})\Big) \times & \\ \Big(P(c_2 \mid c_{\ell_o}) \times \prod_{i=4,6,\ldots}^{\ell} P(c_i \mid c_{i-2}) \times P(c_n \mid c_{\ell_e})\Big) & \text{if } n > 3 \end{cases}$$

Define $\ell = n - (n \mod 2)$ and ℓ_o is the last odd number less than ℓ and ℓ_e is the last even number less than ℓ. As long as the boundary cases are right for the bigrams, we don't penalize off by one in the length, and we don't penalize for $n \leq 3$.

b. (2pts) Using your n-gram model show how $P(kitab) = P(ktb) \times P(ia)$.

Answer:

$$\begin{aligned}
P(kitab) &= P(c_0 = \texttt{<s>}, c_1 = k, c_2 = i, c_3 = t, c_4 = a, c_5 = b, c_6 = \texttt{</s>}) \\
&= P(ktb) \times P(ia, \texttt{</s>}) \\
P(ktb) &= P(c_1 = k \mid c_0 = \texttt{<s>}) \times P(c_3 = t \mid c_1 = k) \times P(c_5 = b \mid c_3 = t) \\
&\quad \text{this term corresponds to the first bracket in the eqn above} \\
P(ia) &= P(c_2 = i \mid c_{\ell_o} = c_5 = b) \times P(c_4 = a \mid c_2 = i) \times P(c_n = c_6 = \texttt{</s>} \mid c_{\ell_e} = c_4 = a) \\
&\quad \text{corresponds to the second bracket in the eqn above}
\end{aligned}$$

c. (4pts) For bigram probabilities $P(c_i \mid c_{i-1})$, Katz backoff smoothing is defined as follows:

$$P_{katz}(c_i \mid c_{i-1}) = \begin{cases} \frac{r^*(c_{i-1},c_i)}{r(c_{i-1})} & \text{if } r(c_{i-1}, c_i) > 0 \\ \alpha(c_{i-1}) P_{katz}(c_i) & \text{otherwise} \end{cases}$$

where $r(\cdot)$ provides the (unsmoothed) frequency from training data and $r^*(\cdot)$ is the Good-Turing estimate of the frequency r defined as follows:

$$r^*(c_{i-1}, c_i) = (r(c_{i-1}, c_i) + 1) \times \frac{n_{r(c_{i-1},c_i)+1}}{n_{r(c_{i-1},c_i)}}$$

where $n_{r(c_{i-1},c_i)}$ is the number of different c_{i-1}, c_i types observed with count $r(c_{i-1}, c_i)$. We assume that linear interpolation has provided all missing $n_{r(\cdot)}$ values required.

$\alpha(c_{i-1})$ is chosen to make sure that $P_{katz}(c_i \mid c_{i-1})$ is a proper probability. Provide the equation for $\alpha(c_{i-1})$.

Answer:

Step by step derivation below. We are just looking for the end result.

$$\begin{aligned}
\sum_{c_i} \left(\frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} + \alpha(c_{i-1}) P_{katz}(c_i) \right) &= 1 \\
\sum_{c_i : r(c_{i-1},c_i) > 0} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} + \alpha(c_{i-1}) \sum_{c_i : r(c_{i-1},c_i) = 0} P_{katz}(c_i) &= 1 \\
\alpha(c_{i-1}) \sum_{c_i : r(c_{i-1},c_i) = 0} P_{katz}(c_i) &= 1 - \left(\sum_{c_i : r(c_{i-1},c_i) > 0} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} \right) \\
\alpha(c_{i-1}) &= \frac{1 - \left(\sum_{c_i : r(c_{i-1},c_i) > 0} \frac{r^*(c_{i-1},c_i)}{r(c_{i-1})} \right)}{\sum_{c_i : r(c_{i-1},c_i) = 0} P_{katz}(c_i)} \\
\alpha(c_{i-1}) &= \frac{1 - \left(\sum_{c_i : r(c_{i-1},c_i) > 0} \frac{r^*(c_{i-1},c_i)}{r(c_{i-1})} \right)}{1 - \left(\sum_{c_i : r(c_{i-1},c_i) > 0} P_{katz}(c_i) \right)}
\end{aligned}$$

Also acceptable is the somewhat less precise answer which assumes $\sum_{c_i} P_{katz}(c_i) = 1$:

$$\alpha(c_{i-1}) = 1 - \sum_{c_i} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})}$$

(2) (10 pts) **Classification**

In word sense disambiguation, we want to predict the sense of a word given the context. Assume that you are given a set of sentences as training data, and you want to build a classifier that predicts the most likely sense for the word *train*. Assume also that we only have the following two senses for the word *train*:

1. a series of railroad cars
2. the long back section of a gown that trails behind the wearer

We are given the following sentences as a training corpus:

	sentence	sense
1	the train is late	1
2	the bridal train is white	2
3	take the train back to Vancouver	1
4	the train in the back is pretty but costly	2
5	go to the back of the train	1

To simplify the problem, we will use a stop list (i.e. these words should be ignored) consisting of `in`, `is`, `the`, `of` and consider a window of 1 word before and 1 word after the target word *train*. We will consider the context after removing stop words.

a. (1pt) Write out the context words for the training examples and specify the training vocabulary.

Answer:

After removing stop words and taking the window of 1 word before and after, we get:

	context words	sense
1	late	1
2	bridal white	2
3	take back	1
4	back	2
5	back	1

The vocabulary is V = {late, bridal, white, take, back} and $|V| = 5$
Note: It is also acceptable for V to include a "UNK" token.

b. (4pts) Using a naive Bayes classifier with add-one smoothing, what is the most likely sense of *train* for the input sentence "the white train"? Write out the probabilities you will need to estimate for the model (assume that we use counts of the token appearing as a context word as our features).

Answer: Note: Partial credit will be given based on the answer to part a).
Let n be the number of tokens in our training data, the priors for the senses are:

$$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}, \quad \hat{P}(\text{sense} = 1) = \frac{4}{7}, \quad \hat{P}(\text{sense} = 2) = \frac{3}{7}$$

We will want to estimate the probability of generating each word given the sense with $\alpha = 1$:

$$\hat{P}(w_i|c_j) \quad = \quad \frac{\text{Count}(w_i, c_j) + \alpha}{\sum_{w \in V}[\text{Count}(w, c_j) + \alpha]}$$

$$\hat{P}(w_i|\,\text{sense} = 1) = \begin{cases} \frac{3}{9}, & w_i \in \{\text{back}\} \\ \frac{2}{9}, & w_i \in \{\text{late, take}\} \\ \frac{1}{9}, & w_i \in \{\text{bridal, white}\} \end{cases} \qquad \hat{P}(w_i|\,\text{sense} = 2) = \begin{cases} \frac{2}{8}, & w_i \in \{\text{bridal, white, back}\} \\ \frac{1}{8}, & w_i \in \{\text{late, take}\} \end{cases}$$

Answer: After dropping the stop words and taking a context of window of 1 for "the white train", we get the following context words: "white". The most likely class is then:

$$\hat{c} = \arg\max_c \hat{P}(c) \prod_{i=1}^{k} \hat{P}(w_i|c)$$

$$\text{sense 1}: \quad \hat{P}(\text{sense} = 1)\hat{P}(w_1 = \text{white}|\,\text{sense} = 1) = \frac{4}{7} \times \frac{1}{9} < \frac{1}{7} \times \frac{1}{2}$$

$$\text{sense 2}: \quad \hat{P}(\text{sense} = 2)\hat{P}(w_1 = \text{white}|\,\text{sense} = 2) = \frac{3}{7} \times \frac{2}{8} > \frac{1}{7} \times \frac{1}{2}$$

Thus, the naive Bayes classifier will predict sense 2.

c. (2pts) Consider the phrase "the white train stopped", what sense would be predicted by the naive Bayes classifier you created above? Do you need to make any adjustment to your model? If so, please explain.

Answer: The context words for the phrase are "white" and "stopped". We will need to handle the unknown word "stopped".

The following answers are accepted for handling the unknown word:

- Because "stopped" is an unknown word, we can handle it by ignoring it, in which case we get the same context as in part a) and sense 2 as the most likely sense.
- If "UNK" is not already in the vocabulary, it needs to be added to the vocabulary. Otherwise, the model will give $\hat{P}(w_i = \text{stopped}) = 0$ and we won't be able to predict which sense it should be.
  The probabilities will need to be adjusted to account for the "UNK" token.

$$\hat{P}(w_i|\,\text{sense} = 1) = \begin{cases} \frac{3}{10}, & w_i \in \{\text{back}\} \\ \frac{2}{10}, & w_i \in \{\text{late, take}\} \\ \frac{1}{10}, & w_i \in \{\text{bridal, white, UNK}\} \end{cases} \qquad \hat{P}(w_i|\,\text{sense} = 2) = \begin{cases} \frac{2}{9}, & w_i \in \{\text{bridal, white, back}\} \\ \frac{1}{9}, & w_i \in \{\text{late, take, UNK}\} \end{cases}$$

When computing the most likely sense, we will need to multiply by $\hat{P}(w_i = \text{stopped})$.

$$\text{sense 1}: \quad \hat{P}(\text{sense} = 1)\hat{P}(w_1 = \text{white}|\,\text{sense} = 1)\hat{P}(w_2 = \text{stopped}|\,\text{sense} = 1) = \frac{4}{7} \times \frac{1}{10} \times \frac{1}{10}$$

$$\text{sense 2}: \quad \hat{P}(\text{sense} = 2)\hat{P}(w_1 = \text{white}|\,\text{sense} = 2)\hat{P}(w_2 = \text{stopped}|\,\text{sense} = 2) = \frac{3}{7} \times \frac{2}{9} \times \frac{1}{9}$$

After fixing the model to handle "UNK", the naive Bayes classifier will predict sense 2.

d. (1pt) Now, consider a log-linear model (aka logistic regression classifier),

$$P(y|x, \theta) = \frac{\exp(\theta \cdot \phi(x, y))}{\sum_{y'} \exp(\theta \cdot \phi(x, y'))}$$

Assume that we have feature functions of the following form:

$$\phi_k(x, y) = \begin{cases} 1 & \text{if } y = \text{sense 1 and white is a context word of } x \\ 0 & \text{otherwise} \end{cases}$$

How many parameters would there be (i.e. what is the size of θ)?

Answer: The number of parameters will be equal to the number of classes × vocabulary size: $2 \times |V|$

e. (2pts) Explain when naive Bayes would be favored over logistic regression, and when logistic regression would be expected to work better.

Answer: When naive Bayes works better:

- Naive Bayes assumes features are conditionally independent. When the independence assumption hold, the Naive Bayes classifier is optimal classifier (it can often work well even when the assumptions are violated).
- Simple and easy to implement, efficient to train
- Can work well even for small amounts of training data (logistic regression tends to overfit on small amount of data).

When logistic regression works better:

- Logistic regression can handle features that are correlated. This allows for exploration of feature combinations that would violate the independence assumption of Naive Bayes.
- With large amount of data, logistic regression will tend to perform better (since features are often correlated).

(3) (10pts) **Word embeddings**

a. (2pt) Provide at least two advantages of dense vectors over sparse vectors.

Answer: Point for each correct answer (up to 2pts). Acceptable answers are:

- Dense vectors are able to better capture similarity between similar words.
- These dense vectors are often shorter, which makes them easier to feed into a machine learning framework as features for training (less parameters to train).
- Models that use dense vectors generalize better to rare words

b. (2pt) Explain the difference between Word2Vec CBOW and Word2Vec SkipGram.

Answer: Word2Vec CBOW builds a classifier to predicts the center word while Word2Vec SkipGram model predicts the context words.

c. (2pt) Explain what is Negative Sampling in the Word2Vec SkipGram model and why is it used.

Answer: Negative sampling randomly picks words as negative examples for training the predictor. It is used because taking all other words as negatives would be prohibitively expensive.

d. (2pt) Explain how TF-IDF is computed by explaining the terms that go into the TF-IDF and the rationale behind them.

Answer: The TF-IDF is composed of the term frequency (TF) and the inverse document frequency (IDF). The term frequency accounts for how often a word appears in the corpus, and the inverse document frequency downweights words such as "the" that appear in a lot of documents but doesn't contribute to the meaning of the document.

For reference, the TF-IDF for word t in document d that we looked at in class is given by:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{Count}(t,d) & \text{if Count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \text{idf}_t = \log\left(\frac{N}{\text{df}_t}\right)$$

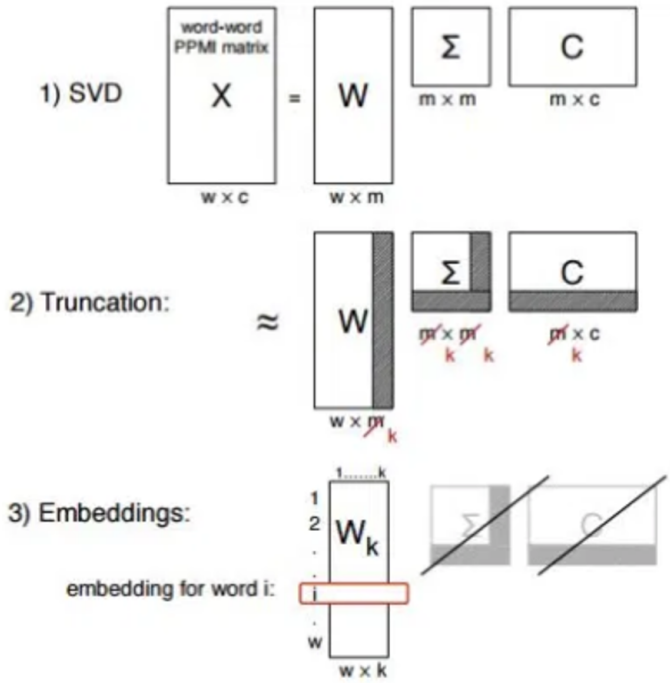
where N is the total number of documents and df_t is the number of documents term t appeared in.

Note that there can be variations in how the two terms (the tf and idf) are computed.

e. (2pt) Explain one way to take a very high-dimensional sparse term-context matrix (of either TF-IDF or PPMI values) and using it to create lower-dimensional dense vectors.

Answer: Full credit is given for providing one of the answers below (explaining how SVD can be used to decompose the matrix, or explaining how the GloVe objective is obtained from the word-word co-occurrence matrix).

- Given such a large sparse matrix, we can create dense word embeddings by doing Singular Value Decomposition (SVD) and keeping vectors corresponding to the top k singular values. The process is illustrated in the figure below.



- The way GloVe embeddings are created represents another way of getting dense word embeddings from word-word co-occurrence counts. In GloVe, the word and context vectors, w_i and $\tilde{w}_j$ (along with the biases b_i and $\tilde{b}_j$) are obtained by optimizing the following loss function:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

where X_{ij} is the number of times word j appears in the context of word i. The final dense word embedding is the sum $w_i + \tilde{w}_i$.

(4) (10pts) **Neural networks**

a. (2pt) Explain what are the advantages of multi-layer neural network classifiers over logistic regression classifiers, and what are the disadvantages.

Answer:

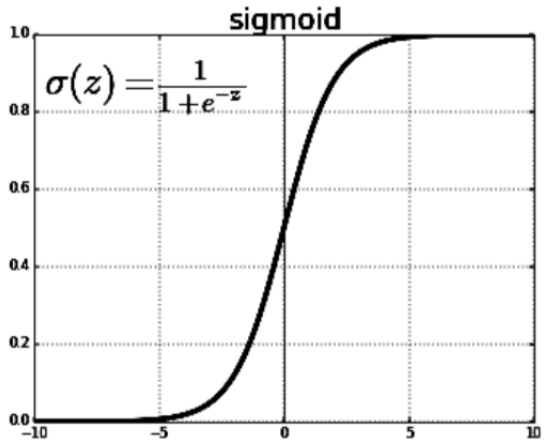
Advantages:

- With multi-layer neural networks, we can now learn feature representations instead of hand-engineering features.

Disadvantages:

- Neural networks are non-convex and more difficult to train than logistic regression classifiers. Typically, they have more parameters and will require a longer time to train and larger datasets.

b. (2pt) Consider the sigmoid activation function. Please explain what are some properties of the sigmoid activation function that makes it less popular in recent times for training neural networks?



Answer:

- Because the sigmoid is almost flat for very large and very small values, its gradient will be almost 0 for those inputs. Once the weights become saturated during a particular training epoch, the small gradients make it very hard to change weights thereafter, which will make training much much slower.
- The output of the sigmoid is also not zero-centered and this can potentially cause issues with zigzagging gradients during training when fed in as input to the next neural network layer (usually of the form $f = w \cdot x + b$).

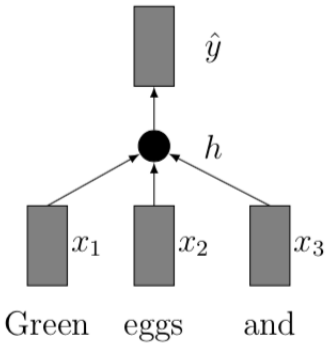
c. (2pt) What are some alternatives to the sigmoid activation function? Please explain what properties makes these activation functions more attractive.

Answer: Possible alternatives are ReLU, Leaky ReLU, Swish, etc. These functions are designed to have gradients that doesn't suffer as much from the saturation issue as the sigmoid and (the ReLU still suffers from that for values < 0). This means that these activation functions suffers less from the problem of vanishing gradients.

More detailed responses can also include:

- Both ReLU and Leaky ReLU also have simple gradients that are computationally friendly
- ReLU also encourages sparsity in parameters
- The Swish function is smooth and non-monotonic (and can be loosely viewed as providing a interpolation between the linear function and the ReLU). The Swish function also has a parameter β that can be trained, giving it more flexibility (there are also members of the RELU/rectified family that are parameterized like the LeakyReLU but when parameterized becomes known as the ParametricdReLU/PReLU).

d. (4pts) Consider the following neural network for predicting the next word given words $x_1, x_2, \ldots, x_3$. What is the derivative of J with respect to $\mathbf{x}$, $\frac{\partial J}{\partial \mathbf{x}}$? Hint: $\tanh(z)' = 1 - \tanh(z)^2$



$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}$$

$$\mathbf{h} = \tanh(W\mathbf{x} + \mathbf{b})$$
$$\hat{\mathbf{y}} = \text{softmax}(U\mathbf{h} + \mathbf{d})$$
$$J = \text{CE}(\mathbf{y}, \hat{\mathbf{y}})$$
$$CE = -\sum_i y_i \log(\hat{y}_i).$$

Answer: Let $\mathbf{z_1} = W\mathbf{x} + \mathbf{b}$ and $\mathbf{z_2} = U\mathbf{h} + \mathbf{d}$.

$$\begin{aligned}
\delta_1 &= \frac{\partial CE}{\partial \mathbf{z_2}} = (\hat{\mathbf{y}} - \mathbf{y}) \\
\delta_2 &= \frac{\partial CE}{\partial \mathbf{h}} = \delta_1 \frac{\partial \mathbf{z_2}}{\partial \mathbf{h}} = U^T \delta_1 \\
\delta_3 &= \frac{\partial CE}{\partial \mathbf{z_1}} = \delta_2 \frac{\partial \mathbf{h}}{\partial \mathbf{z_1}} = \delta_2 \cdot (1 - \tanh(\mathbf{z_1})^2) \\
\frac{\partial J}{\partial \mathbf{x}} &= \frac{\partial CE}{\partial \mathbf{x}} = \delta_3 \frac{\partial \mathbf{z_1}}{\partial \mathbf{x}} = W^T \delta_3
\end{aligned}$$

Note for δ_1, we used the following derivatives of softmax and cross-entropy.
Let's consider the partial derivative of the ith output of $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$ with respect to the jth input of $\mathbf{z}$.

$$\begin{aligned}
\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial}{\partial z_j} \frac{\exp z_i}{\sum_k \exp z_k} &= \frac{\frac{\partial \exp z_i}{\partial z_j} \sum_k \exp z_k - \exp z_i \frac{\partial \sum_k \exp z_k}{\partial z_j}}{(\sum_k \exp z_k)^2} \\
&= \frac{\mathbb{1}_{\{i=j\}} \exp z_i \sum_k \exp z_k - \exp z_i \exp z_j}{(\sum_k \exp z_k)^2} \\
&= \frac{\exp z_i}{\sum_k \exp z_k} \left(\mathbb{1}_{\{i=j\}} - \frac{\exp z_j}{\sum_k \exp z_k} \right) \\
\frac{\partial \hat{y}_i}{\partial z_j} &= \hat{y}_i (\mathbb{1}_{\{i=j\}} - \hat{y}_j)
\end{aligned}$$

Note that $\mathbb{1}$ is the indicator function with $\mathbb{1}_{\{i=j\}} = 1$ when $i = j$ and 0 otherwise.
Now, the partial derivative of the cross entropy loss (with a softmax) with respect to the jth input of $\mathbf{z}$ is:

$$\begin{aligned}
\frac{\partial CE}{\partial z_j} &= \frac{\partial}{\partial z_j} \left(-\sum_i y_i \log(\hat{y}_i) \right) = -\sum_i y_i \frac{\partial}{\partial z_j} \log(\hat{y}_i) = -\sum_i y_i \frac{\partial \log(\hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j} \\
&= -\sum_i y_i \frac{1}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j} = -\sum_i y_i \frac{1}{\hat{y}_i} \hat{y}_i (\mathbb{1}_{\{i=j\}} - \hat{y}_j) = -\sum_i y_i (\mathbb{1}_{\{i=j\}} - \hat{y}_j) \\
&= -y_j + \hat{y}_j \sum_i y_i = \hat{y}_j - y_j
\end{aligned}$$

Note that $\sum_i y_i = 1$ since $\hat{\mathbf{y}}$ is a one-hot vector. Thus, we have

$$\frac{\partial CE}{\partial \mathbf{z}} = \hat{\mathbf{y}} - \mathbf{y}$$