This is your machine learning system?

Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.

What if the answers are wrong?

Just stir the pile until they start looking right.

https://xkcd.com/1838/

CENG 501
Deep Learning
Week 3

Sinan Kalkan
What is machine learning?

\[ x \in I \]

\[ y = f(x) \]

\[ y \in O \]

\[ f(\ )? \]

Previously on CENG501!
# General Approaches (cont’d)

<table>
<thead>
<tr>
<th></th>
<th>Generative</th>
<th>Discriminative</th>
</tr>
</thead>
</table>
| **Supervised**| Neural Networks     | Support Vector Machines  
|               |                     | Neural Networks  
|               |                     | K-Nearest Neighbors  
|               |                     | Decision Trees  
|               |                     | Random Forest  |
| **Unsupervised** | Mixture Models     | K-means               |
Supervised Machine Learning

- Find the following mapping, given the training set \( \{x_i, y_i\}_{i=1}^{N} \):

\[
y = f(x)
\]

\[f: \mathbb{X} \rightarrow \mathbb{Y}.
\]

- Targets can be for classification or regression

- Methods:
  - K Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forest, Neural Networks

<table>
<thead>
<tr>
<th>Input</th>
<th>Target Class</th>
<th>Target Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Happy</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>Happy</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>Happy</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>Happy</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Unsupervised Machine Learning

- Learning without labels
- Goal: Discover a better representation of data
- Uses:
  - Dimensionality reduction
  - Data visualization
  - Preparation for supervised learning
  - Clustering
- Methods:
  - Principle/Independent Component Analysis, k-Means/x-Means Clustering, Manifold learning methods, ...
OTHER FORMS OF LEARNING

Self-supervised learning
Weakly-supervised learning
Zero-shot/Few-shot learning
Reinforcement Learning
Meta-learning
Life-long learning
Model selection

\[ f(x) = Wx \]

\[ f(x) = W_1 x + W_2 x^2 \]

\[ f(x) = \sum_i W_i x_i^i \]

\[ f(x) = W f_1(x) \]

\[ f(x) = W_1 f_1(x) + W_2 f_2(x)^2 \]

\[ \ldots \]
Model Complexity

Models range in their flexibility to fit arbitrary data

- **Simple model**
  - low bias
  - high variance
  - small capacity may prevent it from representing all structure in data

- **Complex model**
  - high bias
  - low variance
  - large capacity may allow it to memorize data and fail to capture regularities

---

Slide Credit: Michael Mozer

Spring 2021

Previously on CENG501!
Bias-Variance Dilemma

Previously on CENG501!
How to evaluate performance

• Hold-out method:
Cross-validation

• K-fold cross validation

Fig: https://scikit-learn.org/stable/modules/cross_validation.html
Cross-validation

- Group/subject-based validation: Determine the folds based on group/subject information
- Leave one subject out (LOSO) cross validation
- Divide dataset into k subsets with respect to subject identifiers

Training and test sets contain the same subjects

Training and test sets do NOT contain the same subjects
Evaluation Metrics
Classification

• Always try to quantify the predictions that have not been made.
• Recall & F-measure are frequently used.

\[
\begin{align*}
TP \text{ Rate (Recall)} &= \frac{TP}{TP + FN} \\
FP \text{ Rate} &= \frac{FP}{FP + TN} \\
Precision &= \frac{TP}{TP + FP} \\
F - measure &= 2 \frac{precision \times recall}{precision + recall}
\end{align*}
\]

Credit: Oya Celiktutan
Evaluation Metrics

Regression

Let $y_k$ and $\hat{y}_k$ be the ground-truth and predicted labels ($N$: number of predictions).

- **Mean Squared Error:**
  \[
  \text{MSE} = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}_k)^2
  \]

- **Correlation:**
  \[
  \text{COR} = \frac{\sum_{k=1}^{N} (y_k - \mu_{y_k})(\hat{y}_k - \mu_{\hat{y}_k})}{\sqrt{\sum_{k=1}^{N} (y_k - \mu_{y_k})^2} \sqrt{\sum_{k=1}^{N} (\hat{y}_k - \mu_{\hat{y}_k})^2}}
  \]

  where $\mu_{y_k}$ and $\mu_{\hat{y}_k}$ are the sample means.

- **Coefficient of determination:**
  \[
  R^2 = 1 - \frac{\sum_{k=1}^{N} (y_k - \hat{y})^2}{\sum_{k=1}^{N} (y_k - \mu_{y_k})^2}
  \]

Credit: Oya Celiktutan
Today

• Before deep learning
  – History of deep learning
  – Biological neuron
  – Artificial neuron
  – Perceptron learning

• Towards deep learning
  – Linear classification/regression with gradient descent
  – Non-linear classification/regression
Administrative Issues

- Registrations
- Access to Moodle (https://odtuclass.metu.edu.tr/)
- First Take-Home Exam this week
- Project paper selection
  - https://docs.google.com/spreadsheets/d/1tzPHq_Vgu6gCwNyXJHGVqeA6pgU67H0nKYjqqisWfKc/edit?usp=sharing
  - Deadline: 29 March + 1-2 weeks
BEFORE DEEP LEARNING

History of deep learning
Biological neuron
Artificial neuron
Perceptron learning
Neuron

A neuron

• receives input signals generated by other neurons through its dendrites,

• integrates these signals in its body,

• then generates its own signal (a series of electric pulses) that travel along the axon which in turn makes contacts with dendrites of other neurons.

• The points of contact between neurons are called synapses.
Neuron

• The pulses generated by the neuron travels along the axon as an electrical wave.

• Once these pulses reach the synapses at the end of the axon open up chemical vesicles exciting the other neuron.

Slide credit: Erol Sahin
Neuron

(Carlson, 1992)

http://animatlab.com/Help/Documentation/Neural-Network-Editor/Neural-Simulation-Plug-ins/Firing-Rate-Neural-Plug-in/Neuron-Basics
The biological neuron - 2

(Carlson, 1992)
Face selectivity in IT

http://www.billconnelly.net/?p=291
Some facts about human brain

- ~ 86 billion neurons
- ~ $10^{15}$ synapses

Fig: I. Goodfellow
ARTIFICIAL NEURON
Alexander Bain (1818 –1903)

“Bain on Neural Networks”, Wilkes & Wade, 1997.

McCulloch-Pitts Neuron

\[ net = \sum_i w_i x_i + b \]

\[ f(net) = \begin{cases} 0, & net < 0 \\ 1, & net \geq 0 \end{cases} \]

Frank Rosenblatt (1928-1971)

Perceptron Learning


Sinan Kalkan

1873 1943 1958
Convolutional Weights
(Fukushima’s Neocognitron, 1979)

Backpropagation
(Werbos, 1982)
(Parker, 1985; LeCun, 1985)
(Rumelhart vd., 1986)

Convolutional Neural Networks
LeCun et al. (1989)

Object Classification
Traffic Sign Recognition
Cancer Detection

Alexander Bain
McCulloch-Pitts Neurons
Perceptron Learning

1873 '43 '58 '79 '82 '85 '86 '89 2012
<table>
<thead>
<tr>
<th>Year</th>
<th>Contributor</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>300 BC</td>
<td>Aristotle</td>
<td>introduced Associationism, started the history of human’s attempt to understand brain.</td>
</tr>
<tr>
<td>1873</td>
<td>Alexander Bain</td>
<td>introduced Neural Groupings as the earliest models of neural network, inspired Hebbian Learning Rule.</td>
</tr>
<tr>
<td>1943</td>
<td>McCulloch &amp; Pitts</td>
<td>introduced MCP Model, which is considered as the ancestor of Artificial Neural Model.</td>
</tr>
<tr>
<td>1949</td>
<td>Donald Hebb</td>
<td>considered as the father of neural networks, introduced Hebbian Learning Rule, which lays the foundation of modern neural network.</td>
</tr>
<tr>
<td>1958</td>
<td>Frank Rosenblatt</td>
<td>introduced the first perceptron, which highly resembles modern perceptron.</td>
</tr>
<tr>
<td>1974</td>
<td>Paul Werbos</td>
<td>introduced Backpropagation</td>
</tr>
<tr>
<td>1978</td>
<td>Teuvo Kohonen</td>
<td>introduced Self Organizing Map</td>
</tr>
<tr>
<td>1982</td>
<td>Kunihiko Fukushima</td>
<td>introduced Neocogitron, which inspired Convolutional Neural Network</td>
</tr>
<tr>
<td>1985</td>
<td>John Hopfield</td>
<td>introduced Hopfield Network</td>
</tr>
<tr>
<td>1986</td>
<td>Hilton &amp; Sejnowski</td>
<td>introduced Boltzmann Machine</td>
</tr>
<tr>
<td>1986</td>
<td>Paul Smolensky</td>
<td>introduced Harmonium, which is later known as Restricted Boltzmann Machine</td>
</tr>
<tr>
<td>1990</td>
<td>Yann LeCun</td>
<td>introduced LeNet, showed the possibility of deep neural networks in practice</td>
</tr>
<tr>
<td>1997</td>
<td>Schuster &amp; Paliwal</td>
<td>introduced Bidirectional Recurrent Neural Network</td>
</tr>
<tr>
<td>1997</td>
<td>Hochreiter &amp; Schmidhuber</td>
<td>introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks</td>
</tr>
<tr>
<td>2006</td>
<td>Geoffrey Hinton</td>
<td>introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.</td>
</tr>
<tr>
<td>2009</td>
<td>Salakhutdinov &amp; Hinton</td>
<td>introduced Deep Boltzmann Machines</td>
</tr>
<tr>
<td>2012</td>
<td>Geoffrey Hinton</td>
<td>introduced Dropout, an efficient way of training neural networks</td>
</tr>
</tbody>
</table>
Bain on Neural Networks

ALAN L. WILKES AND NICHOLAS J. WADE

University of Dundee, Dundee, Scotland

In his book *Mind and body* (1873), Bain set out an account in which he related the processes of associative memory to the distribution of activity in neural groupings—or neural networks as they are now termed. In the course of this account, Bain anticipated certain aspects of connectionist ideas that are normally attributed to 20th-century authors—most notably Hebb (1949). In this paper we reproduce Bain’s arguments relating neural activity to the workings of associative memory which include an early version of the principles enshrined in Hebb’s neurophysiological postulate. Nonetheless, despite their prescience, these specific contributions to the connectionist case have been almost entirely ignored. Eventually, Bain came to doubt the practicality of his own arguments and, in so doing, he seems to have ensured that his ideas concerning neural groupings exerted little or no influence on the subsequent course of theorizing in this area. © 1997 Academic Press

Alexander Bain (1818–1903), see Fig. 1, is best known for his textbooks *The senses and the intellect* (1855) and *The emotions and the will* (1859), in which he offered an interpretation of mental phenomena within an associationist framework (for further biographical detail, see Hearnshaw, 1964). Specifically, he tried to match quantitative estimates of the associations held in memory to the neural structure of the brain. It was this exercise that first drew Bain into confronting the potential properties of neural groupings or networks. In the course of thinking about these issues, he was led to speculate on how the internal structure of neural groupings could physically grow to reflect the contingencies of experience and how this same internal structure could come to support the variety of associative links typically found in memory.
Fig. 2. Bain's diagram illustrating the way in which the connections in a neural network can channel activation in different directions:

It requires us to assume, not merely fibres multiplying by ramification through the cell junctions, but also an extensive arrangement of cross connections. We can typify it in this way. Suppose \( a \) enters a cell junction, and is replaced by several branches, \( a', a'' \) etc; \( b \) in like manner, is multiplied into \( b', b'' \) etc. Let one of the branches of \( a \) or \( a' \), pass into some second cell, and a branch of \( b \), or \( b' \), pass into the same, and let one of the emerging branches be \( X \); we have then a means of connecting united \( a \) and \( b \) with \( X \); and in some other crossing, a branch of \( b \) may unite with a branch of \( c \), from which crossing \( Y \) emerges and so on. . . . By this plan we comply with the primary condition of assigning a separate outcome to every different combination of sensory impressions.

The diagram shows the arrangement. The fibre \( a \) branches into two \( a', a'' \); the fibre \( b \) into \( b', b'' \); \( c', c'' \). One of the branches of \( a \) unites with one of the branches of \( b \), or \( a', b' \) in a cell \( X \); \( b', c' \) unite in \( Y \), in \( Z \). (1873, pp. 110, 111)
McCulloch-Pitts Neuron  
(McCulloch & Pitts, 1943)

- Binary input-output
- Can represent Boolean functions.
- No training.

\[
\text{net} = \sum_i (w_{y,x_i}x_i) + w_{y,z_b} 
\]

\[
f(\text{net}) = \begin{cases} 
0, & \text{net} < 0 \\
1, & \text{net} \geq 0 
\end{cases}
\]

http://www.tau.ac.il/~tsirel/dump/Static/knowino.org/wiki/Artificial_neural_network.html
McCulloch-Pitts Neuron

• Implement \(\text{AND}(x, y)\):
  – Let \(w_x\) and \(w_y\) to be 1, and \(w_{+1}\) to be -2.
• When input is 1 & 1; net is 0.
• When one input is 0; net is -1.
• When input is 0 & 0; net is -2.

\[
f(\text{net}) = \begin{cases} 
0, & \text{net} < 0 \\
1, & \text{net} \geq 0 
\end{cases}
\]

http://www.tau.ac.il/~tsirel/dump/Static/knowino.org/wiki/Artificial_neural_network.html
McCulloch-Pitts Neuron

• Binary input-output is a big limitation

• Also called

  “ [...] caricature models since they are intended to reflect one or more neurophysiological observations, but without regard to realism [...]”

  -- Wikipedia

• No training! No learning!

• They were useful in inspiring research into connectionist models
THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN

F. ROSENBLATT

Cornell Aeronautical Laboratory

**Fig. 1.** Organization of a perceptron.
https://www.youtube.com/watch?v=cNxadbrN_aI
Let’s go back to a biological neuron

- A biological neuron has:
  - Dendrites
  - Soma
  - Axon

- Firing is continuous, unlike most artificial neurons
- Rather than the response value, the firing rate is critical
• Neurone vs. Node

• Very crude abstraction
• Many details overseen

“Spherical cow” problem!
Q: How does a physicist milk a cow?
A: Well, first let us consider a spherical cow...

Or

“Milk production at a dairy farm was low, so the farmer wrote to the local university, asking for help from academia. A multidisciplinary team of professors was assembled, headed by a theoretical physicist, and two weeks of intensive on-site investigation took place. The scholars then returned to the university, notebooks crammed with data, where the task of writing the report was left to the team leader. Shortly thereafter the physicist returned to the farm, saying to the farmer, "I have the solution, but it only works in the case of spherical cows in a vacuum".”
More on this

• https://medium.com/intuitionmachine/neurons-are-more-complex-than-what-we-have-imagined-b3dd00a1dcd3
PERCEPTRON LEARNING
Let us take a closer look at perceptrons

- Initial proposal of connectionist networks
- Rosenblatt, 50’s and 60’s
- Essentially a linear model composed of nodes and weights

\[ y(x) = \begin{cases} 
1, & w_0 + w_1 x_1 + \ldots + w_n x_n > 0 \\
0, & \text{otherwise}
\end{cases} \]

Or, simply
\[ y(x) = \text{sgn}(w \cdot x) \]
where \( \text{sgn}(x) = \begin{cases} 
0, & x \leq 0 \\
1, & x > 0
\end{cases} \)
Motivation for perceptron learning  

(No gradient descent yet)

• We have estimated an output $\hat{y} = \text{sgn}(w \cdot x)$
  – But the target was $y$
• Error (simply): $y - \hat{y}$
• Let us update each weight such that we “learn” from the error:
  – $w_i \leftarrow w_i + \Delta w_i$
  – where $\Delta w_i \propto (y - \hat{y})$
• We somehow need to distribute the error to the weights. How?
  – Distribute the error according to how much they contributed to the error: Bigger input contributes more to the error.
  – Therefore: $\Delta w_i \propto (y - \hat{y})x_i$
An example

• Consider \( x_i = 0.8, y = 1, \hat{y} = -1 \)
  – Then, \( (y - \hat{y})x_i = 1.6 \)
  – Which will increase weight \( w_i \) by 1.6.
  – Which makes sense considering the output and the target
Perceptron training rule

• Update weights
  \[ w_i \leftarrow w_i + \Delta w_i \]

• How to determine \( \Delta w_i \)?
  \[ \Delta w_i \leftarrow \eta (y - \hat{y}) x_i \]
  – \( \eta \): learning rate – can be slowly decreased
  – \( y \): target/desired output
  – \( \hat{y} \): current output, prediction
Perceptron - intuition

- A perceptron defines a hyperplane in N-1 space: a line in 2-D (two inputs), a plane in 3-D (three inputs),....
- The perceptron is a linear classifier: It’s output is -1 on one side of the plane, and 1 for the other.
- Given a linearly separable problem, the perceptron learning rule guarantees convergence.
Problems with perceptron

• Perceptron unit is non-linear

• However, it provides zero gradient (due to thresholding function), which makes it unsuitable to gradient descent in multi-layer networks.
Problems with perceptron learning

• Can only learn linearly separable classification.

linearly separable  

not linearly separable
Linear classification/regression
Non-linear classification/regression
Multi-layer perceptrons

TOWARDS DEEP LEARNING
LINEAR CLASSIFICATION AND REGRESSION
Linear Classification

• Goal: Find the following mapping, given the training set \( \{x_i, y_i\}_{i=1}^N \):

\[
y = f(x)
\]

\[
f: \mathbb{X} \rightarrow \mathbb{Y}.
\]

• Linear model:

\[
y = f(x) = f(x; W, b) = \text{sign}(Wx + b) = \text{sign}\left(\sum_{i=1}^{N} w_i x_i + b\right),
\]

where \( w_i \) (\( i = 1, \ldots, N \)) and \( b \) are parameters to be learned.
Linear Classification with Neurons

\[ t = \sum w_i x_i + b \]

apple
car
house
person
...
Linear Classification with Neurons

\[ t = \sum w_i x_i \]

\[ y = f(x) = f(x; \theta) = \sum_i w_i x_i + b = \mathbf{w} \cdot \mathbf{x} \]
Linear classification

\[ y = f(x; W, b) = Wx + b = \sum_{i=1}^{N} w_i x_i + b \]
Linear classification: One interpretation

\[ f(x_i; W, b) = Wx_i + b \]

**Interpretation:** Each row of \( W \) and \( b \) describes a line for a class

Figure: http://cs231n.github.io/linear-classify/
Linear classification:
Another interpretation

- Each row in $W$ can be interpreted as a template of that class.
  
  - $f(x_i; W, b) = Wx_i + b$ calculates the inner product to find which template best fits $x_i$.
  
  - Effectively, we are doing Nearest Neighbor with the “prototype” images of each class.

http://cs231n.github.io/linear-classify/
Loss function

• A function which measures how good our parameters (weights) are.
  – Other names: cost function, objective function
• Let $s_j = f(x_i; W)_j$
• An example loss function:
  $$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$
  Or equivalently:
  $$L_i = \sum_{j \neq y_i} \max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta)$$
• This forces the distances to other classes to be more than $\Delta$ (the margin)

http://cs231n.github.io/linear-classify/
Example

• Consider our scores for $\mathbf{x}_i$ to be $s = [13, -7, 11]$ and assume $\Delta$ as 10.

• Then,

$$L_i = \max(0, -7 - 13 + 10) + \max(0, 11 - 13 + 10)$$
Regularization

- In practice, there are many possible solutions leading to the same loss value.
  - Based on the requirements of the problem, we might want to penalize certain solutions.
- E.g.,

\[ R(W) = \sum_i \sum_j W_{i,j}^2 \]

- which penalizes large weights.
  - Why do we want to do that?

http://cs231n.github.io/linear-classify/
Why penalize large weights?

\[ R(W) = \sum_i \sum_j W_{i,j}^2 \]

- The solution is not unique: \( W \) is a solution, so is \( \propto W \).
- Large \( W \) has large variance:
  - Large and small weights can lead to abrupt changes in the boundary.
  - I.e. overfitting.
- Robustness to small changes in the input.
Combined Loss Function

• The loss function becomes:

\[ L = \frac{1}{N} \sum \sum L_i + \lambda R(W) \]

• If you expand it:

\[ L = \frac{1}{N} \sum \sum \left[ \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta) \right] + \lambda \sum \sum W_{i,j}^2 \]

Hyper parameters
(estimated using validation set)

http://cs231n.github.io/linear-classify/
Hinge Loss, or Max-Margin Loss

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[ \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta) \right] + \lambda \sum_i \sum_j W_{i,j}^2$$

http://cs231n.github.io/linear-classify/
Interactive Demo

http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/
Regression loss

\[ L = \frac{1}{N} \sum_{i} \sum_{j} (s_{ij} - y_{ij})^2 + \lambda \sum_{i} \sum_{j} w_{i,j}^2 \]

In general:
\[ \sum_{j} |s_j - y_j|^q \]

- \( q = 1 \): Absolute Value Loss
- \( q = 2 \): Square Error Loss.

Figure 1.29  Plots of the quantity \( L_q - |y - \tilde{y}|^q \) for various values of \( q \).  

Bishop
\[ L(x; \theta) = d(y, f(x; \theta)) \]

\[ \theta^* = \arg \min_{\theta} L(x; \theta) \]
\[ \theta^* = \arg \min_{\theta} L(x; \theta) \]

\[ \theta_{t+1} \leftarrow \theta_t - \eta \nabla_{\theta} L(x; \theta) \]
\[ \theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta L(x; \theta) \]

Learning Rate (Step Size)

\[ \nabla_\theta L(x; \theta) = \frac{\partial L(x; \theta)}{\partial \theta} \]
Gradient Descent

• True Gradient Descent
  – Calculate the loss & the gradient on the whole dataset
  – Then make the update
• Stochastic Gradient Descent
  – Calculate the loss & the gradient on examples one at a time
  – Update the weights after each example
• Batch Gradient Descent
  – Calculate the loss & the gradient on a set of examples (batch)
  – Update the weights after each bath
Gradient Descent

**Input**: Training set: \((x, y)_i, i = 1, \ldots, N\)

The network

**Output**: Network parameters, \(\theta\)

1. \(\theta_0 \leftarrow \) Random initial values
2. Until convergence:
   i. Take \(m\) samples from the dataset randomly
   ii. Calculate predictions, \(\hat{y}\), on \(m\) samples using the current parameters \(\theta_t\)
   iii. Calculate loss \(L()\) and \(\nabla_{\theta} L\)
   iv. Update the weights
      \[
      \theta_{t+1} \leftarrow \theta_t - \eta \nabla_{\theta} L
      \]
Gradient descent

https://en.wikipedia.org/wiki/Gradient_descent

(Goodfellow vd., 2016)
Derive the gradients of hinge loss

\[ \frac{\partial L_i}{\partial w_{jk}} = \begin{cases} \frac{\partial L_i}{\partial e_j} \frac{\partial e_j}{\partial s_j} \frac{\partial s_j}{\partial w_{jk}} & \text{if } s_j - s_{y_i} + \Delta > 0 \\ 1 & \text{otherwise} \end{cases} \]

\[ \frac{\partial L_i}{\partial w_{jk}} = \mathbb{I}(s_j - s_{y_i} + \Delta > 0) x_k \]

This assumed that \( j \neq y_i \). What happens if that's not the case? See the next page.
Derive the gradients of hinge loss

$$\frac{\partial L_i}{\partial w_{jk}} = ?$$

$$\frac{\partial L_i}{\partial w_{jk}} = \frac{\partial L_i}{\partial e_j} \frac{\partial e_j}{\partial s_j} \frac{\partial s_j}{\partial w_{jk}}$$

$$\sum_{j \neq y_i} \mathbb{I}(s_j - s_{y_i} + \Delta > 0)(-1)$$

$$\frac{\partial L_i}{\partial w_{jk}} = \sum_{j \neq y_i} \mathbb{I}(s_j - s_{y_i} + \Delta > 0)(-1)x_k$$