Lecture 10: Other forms of learning

Metric, Rank, Recommend, Embedding

Guangzong Chen

University of Pittsburgh

July 13, 2025

Table of Contents

Metric Learning

Most frequently used metrics of similarity (or dissimilarity) between two feature vectors are **Euclidean distance** and **cosine similarity**.

Euclidean distance

$$d^2(x, x') = \langle x - x', x - x' \rangle$$

We can modify Euclidean distance to make to parametrizable and learn metrics from data

Modified Euclidean distance

$$d_A^2(x, x') = \|x - x'\|_A = (x - x')^T A(x - x') =< x - x', x - x' >_A$$

Example

Euclidean distance A $\stackrel{\text{def}}{=} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Weighted A $\stackrel{\text{def}}{=} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Definition - Metric

1. $d(x, x') \geq 0$ nonnegativity
2. $d(x, x') \leq d(x, x') + d(x', z)$ triangle inequality
3. $d(x, x') = d(x', x)$ symmetry

In order to satisfy the definition of metric, A should be positive semi-definite. To satisfy the third condition, we can define a new metric which is $d'(x, x') = d(x, x') + d(x', x)$.

Training

We need to create two sets. The first set S is such that a pair of examples (x_i, x_k) belong to S if x_i and x_k are similar.
The second set D is such that a pair of examples (x_i, x_k) belongs to set D if x_i and x_k are dissimilar.

Optimization Problem

$$\min_{\mathrm{A}} \sum_{(\mathrm{x}_i,\mathrm{x}_k)\in\mathcal{S}} \left\|\mathrm{x}-\mathrm{x}'\right\|_{\mathrm{A}}^2 \text{ such that } \sum_{(\mathrm{x}_i,\mathrm{x}_k)\in\mathcal{D}} \left\|\mathrm{x}-\mathrm{x}'\right\|_{\mathrm{A}} \geq c$$

Riemannian Metric

Riemannian Metric is very related to this topic. They are almost defined in the same way. A Riemannian Metric defines the inner product of two vectors in the **tangent space**.

Riemannian Metric

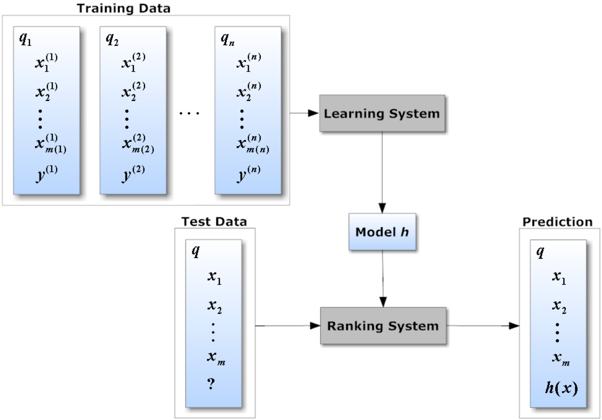
$$g(X, Y) = \sum g_{ij} x_i x_j$$

Also, a metric actually is a mapping from $R^n \times R^n$ to R, which defined as

$$g : R^n x R^n \rightarrow R.$$

This is a tensor. This may be the reason that modern machine learning frameworks use the word "tensor", such as TensorFlow and Pytorch.

Problem Settings



1. Query q: search keywords
2. Document: A paper, page, article
3. Collection of documents: A set of documents

Features of documents are different if the query is different.
The feature x_{ij} is calculated from document x_i and query q_j.
For a query q_j, we have a collection of documents X_j and the corresponding score of the documents Y_j.
Our training data contains a set of collection of documents. Problem:

Given a new query q_k and a set of documents, the task is to determine the rank of each document. (Sort the document)
The feature of document x_{ik} is known.

Methods

1. Pointwise
   Considered documents in the collection as unrelated. Then build a regression model to predict the score of each document.
2. Pairwise
   The idea is from sorting.
3. Listwise
   Build a metric, then minimize the metric.

Pairwise – Sorting

1. Bubble Sort
2. Selection Sort
3. Quick Sort

To sort an array, all we need is a function that can compare two elements in the array.
The idea here remains the same: We construct a model capable of determining which document should be given a higher rank in the list.
Training the model inside each query of documents.
What is the difference between the pointwise method and pairwise method?

Problem settings: We have user information and their preference. Give a new thing, will they like it? Youtube, Tiktok.

	user				movie					rated movies								
	Ed	Al	Zak	···	It	Up	Jaws	Her		It	Up	Jaws	Her					
	x_1	x_2	x_3	···	x_{21}	x_{22}	x_{23}	x_{24}	···	x_{40}	x_{41}	x_{42}	x_{43}	···	x_{99}	x_{100}	y	
$\mathbf{x}^{(1)}$	1	0	0	···	1	0	0	0	···	0.2	0.8	0.4	0	···	0.3	0.8	1	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	···	0	1	0	0	···	0.2	0.8	0.4	0	···	0.3	0.8	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	···	0	0	1	0	···	0.2	0.8	0.4	0.7	···	0.3	0.8	2	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	···	0	0	1	0	···	0	0	0.7	0.1	···	0.35	0.78	3	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	···	0	0	0	1	···	0	0	0.7	0.1	···	0.35	0.78	1	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	···	1	0	0	0	···	0.8	0	0	0.6	···	0.5	0.77	4	$y^{(6)}$
···	···	···	···	···	···	···	···	···	···	···	···	···	···	···	···	···	···	···
$\mathbf{x}^{(D)}$	0	0	0	···	0	0	1	0	···	0	0	1	0	···	0.95	0.85	5	$y^{(D)}$

Figure 1: Example for sparse feature vectors $\mathbf{x}$ and their respective labels y.

Factorization Machines

Model

The factorization machine model is defined as follows:

$$f(\mathrm{x}) \stackrel{\text{def}}{=} b + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} (\mathrm{v}_i \mathrm{v}_j)\, x_i x_j$$

where b and $w_i, i = 1, \ldots D$ are scalar parameters similar to those used in linear regression. Vectors $v_i, i = 1, \ldots D$, are k-dimensional vectors of factors. k is a hyperparameter and is usually much smaller than D.

Loss Function

The loss function can be squared error loss (for regression) or hinge loss. For classification with $y \in \{-1, +1\}$, with hinge loss or logistic loss the prediction is made as $y = sign(f(x))$.

Denoising Autoencoders (DAE)

1. A Denoising autoencoder is a neural network that reconstruct its input from bottleneck layer.
2. For recommander model, the denoising autoencoder is to reconstruct those removed item.
3. Imagine you possess a list encompassing the preferences of various users. Presently, you are faced with a situation where you only possess partial data related to a new user.

Word Embeddings

1. word2vec – skip-gram or CBOW
2. The dataset are generated from unlabeled text.

Example: "I almost finished reading the book on machine learning."
Let's denote words in a skip-gram like this: $[x_{-3}, x_{-2}, x_{-1}, x, x_{+1}, x_{+2}, x_{+3}]$. In our above example of the sentence, x_{-3} is the one-hot vector for "finished," x_{-2} corresponds to "reading," x is the skipped word $(\cdot)$, x_{+1} is "on" and so on. A skip-gram with window size 5 will look like this: $[x_{-2}, x_{-1}, x, x_{+1}, x_{+2}]$.

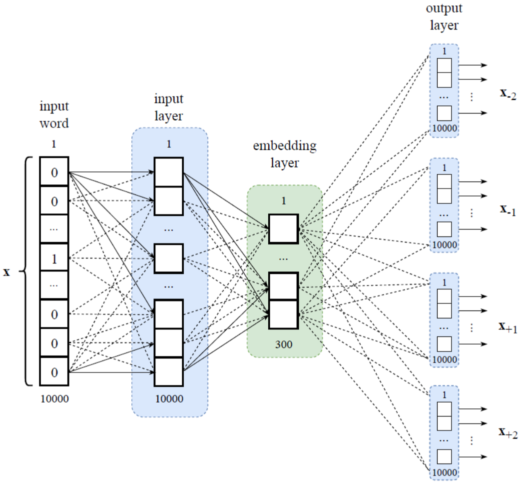
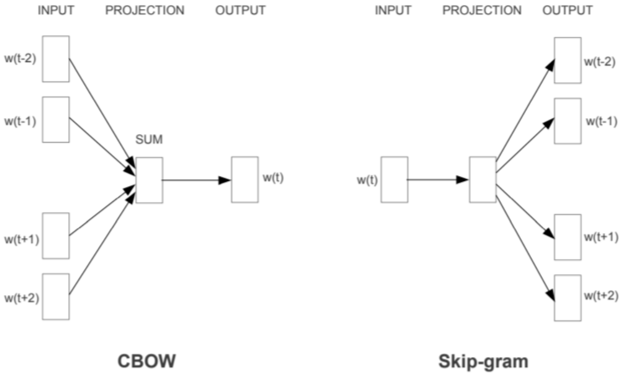


Figure 2: The skip-gram model with window size 5 and the embedding layer of 300 units.


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram