Deep Learning

Instructor: Tongyao Pang

Notes Taker: Zejin Lin

TSINGHUA UNIVERSITY.

linzj23@mails.tsinghua.edu.cn

lzjmaths.github.io

April 11, 2025

Contents

1 Regression

$$\min_{\omega\in\mathbb{R}^m} \frac{1}{2N}\|\Phi\omega - y\|^2 + \lambda C(\omega) \tag{1.1}$$

Lasso: $C = \|\omega\|_1$. **Ridge regression**: $C = \|\omega\|_2$.

subgradient of f:

$$\partial f(x_0) = \{g | f(x) \geqslant f(x_0) + g^T(x - x_0)\}$$

In particular,

$$\partial|x| = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \\ [-1, 1], & x = 0 \end{cases}$$

1.1 Binary classification problem

one-hot encoding for the output $\{\binom{1}{0}, \binom{0}{1}\}$. It can be understood as the probability for each class and can take continuous values.

A **linear hypothesis space** is $\{u(x) : u = \omega^T x, x \in \mathbb{R}^n, \omega \in \mathbb{R}^n\}$.

Softmax:Map the extracted feature u to the space of one-hot codes

$$\mu = \frac{1}{1 + e^{-u}}, \quad 1 - \mu = \frac{e^{-u}}{1 + e^{-u}} = \frac{1}{1 + e^{u}}$$

$$KL(p, q) = \int p(\log p - \log q) \tag{1.2}$$

For p real probability, to minimize (1.2), suffices to minimize

$$-\int p \log q_\theta \mathrm{d}x = -\sum_{x_i} \log q_\theta(x_i)$$

which is called **Maximum likelihood** (**cross entropy**)

$$-\sum \log p(y_i|x_i, \omega) = \sum -y_i \log \mu_i - (1 - y_i)\log(1 - \mu_i)$$

We reduce to minimize the thing above.

1.2 Gradient Descent

$$J(\theta) = \sum_{i=1}^{N} L(f_\theta(x_i), y_i), \quad \theta^{t+1} = \theta^t - \eta_t \frac{\partial J(\theta)}{\partial \theta}|_{\theta=\theta^t}$$

For **empirical loss**

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} J_i(\theta), J_i(\theta) L(f_\theta(y_i), x_i)$$

Stochastic Gradient Descent:only compute gradients over a mini batch for each epoch

$$\theta_{k+1} = \theta_k - \eta \frac{1}{B} \sum_{i \in I_B} \nabla J_i(\theta_k)$$

I_B, called **mini batch** are randomly sampled from the training data indexes.

The motivation to sample is that for distribution x, $\frac{x_1 + \cdots + x_N}{N}$ has the same mean μ but less variance $\frac{1}{N}\sigma^2$, so in order to have more randomness and greatly reduce the computational cost, we choose less ammount of data.

Randomness can help avoid getting stuck in local minimum.

SGD: $x_{t+1} = x_t - \alpha \nabla f(x_t)$.

SGD+**momentum**:

$$v_{t+1} = \rho v_t + \nabla f(X_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

v_t is the momentum which helps accelerate convergence by accumulating the gradients of past steps and smoothing out the oscillations, where $\rho = 0.9$ or 0.99.

1.3 Adaptive learning rate

$$r_t = r_{t-1} + \nabla f(x_t) \bigodot \nabla f(x_t)$$

$$x_{t+1} = x_t - \frac{\alpha}{\sqrt{r_t + \varepsilon}} \bigodot \nabla f(x_t)$$

For frequent features, the updates will be smaller, and for rare features, the updates will be larger.

Notation $\odot$ is the multiplication for each component, which means:

$$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \odot \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ \vdots \\ a_n b_n \end{pmatrix}$$

2 MLP

Exponential Moving Averaging(EMA)

$$r_t = \beta r_{t-1} + (1-\beta)\nabla f(x_t) \odot \nabla f(x_t)$$

$$x_{t+1} = x_t - \frac{\alpha}{\sqrt{r_t + \varepsilon}} \bigodot \nabla f(x_t)$$

RMSProp uses a moving average, avoiding overly aggresive decay complared with AdaGrad.

Adaptive Moment Estimation(Adam):RMAProp+Momentum

$$g_t = \nabla f(x_t)$$

$$v_t = \beta_1^t v_{t-1} + (1-\beta_1^t) g_t, r_t = \beta_2^t r_{t-1} + (1-\beta_2^t) g_t \odot g_t$$

$$v_t = \frac{v_t}{1-\beta_1^t}, E_t = \frac{E_t}{1-\beta_2^t}$$

$$x_{t+1} = x_t - \frac{\alpha}{\sqrt{r_t + \varepsilon}} \odot v_t$$

3 Vanishing or Exploding Gradient

The gradient of the Sigmoid function is very small most of the time, leading to vanishing gradients

We want to avoid exploding or vanishment in gradient.

Sigmoid: $\sigma(z) = \dfrac{1}{1+e^{-z}}$.

Tanh: $\sigma(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$.

ReLU: $\mathrm{ReLU}(z) = \begin{cases} z, & z > 0 \\ 0, & \text{otherwise} \end{cases}$.

LeakyReLU: $\mathrm{LeakyReLU}(z) = \begin{cases} z, & z > 0 \\ az, & \text{otherwise} \end{cases}$. The gradient neither vanishes nor explodes; it is computationally fast, but some neurons may not be activated.

LeakyReLU solves the issue with ReLU and is the most commonly used.

Consider the back propagation $\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial J}{\partial x}W$. Cumulate after multi-layers

$$\mathrm{Var}(\frac{\partial J}{\partial x}) = \prod_i n_l \mathrm{Var}(W_l)\mathrm{Var}(\frac{\partial J}{\partial x_l})$$

We want $n_l \mathrm{Val}(W_l) \sim 1$.

After normalizing their variances, the updating becomes more steady and efficient.

To avoid variance becoming too small or too large in deep layers, normalize features in the network

$$\widehat{x}_i = \frac{x_i - \mathbb{E}x_i}{\sqrt{\mathrm{var}(x_i)}}$$

Batch Normalization: normalize features across samples within each batch.

4 Convolution

4.1 Filtering for Image Processing

Mean filtering: replacing each pixel value in an image with the mean value of its neighbors

Median filtering: replacing each pixel value in an image with the median value of its neighbors.

Convolution of two functions f and h

$$g(x) = f(x) * h(x) = \int f(\tau)h(x-\tau)d\tau$$

In some sense, convolution is a way to do weight average to for f locally.

Discrete 1D convolution is a matrix convolution

$$g[i] = \sum_{n=1}^{N} f[n]h[i-n]$$

where the corresponding matrix is a multi-diagonal matrix.

Continuous 2D convolution

$$g(x) = \iint f(\tau)h(x-\tau)\mathrm{d}\tau$$

Discrete 2D convolution

$$g[i,j] = \sum_{n=1}^{N}\sum_{m=1}^{N} f[n,m]h[i-n,j-m]$$

If the input shape $n_h \times n_w$, the convolution kernel shape $k_h \times k_w$ and the output shape will be $(n_h - k_h + 1) \times (n_w - k_w + 1)$.

Different Filter can extract different features.

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

is a high-pass filter, which extract the edge in the image.

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

is a low-pass filter, which makes the image more smooth, *i.e.* fuzzy, describing a global feature.

4.2 Convolution neural network

In practice, we can use C_{out} filters of $C_{in} \times H \times M$ to obtain a feature of $C_{out} \times H \times M$ size, with input C_{in} extracted features of $H \times M$ size.

Receptive field of CNN expands along with the depth of NN.

We use the **stride** to expand receptive field in one layer.

Pooling Layer:

Max pooling:

$$y[i] = \max_{j \in N(i)} x[j]$$

Average Pooling

$$y[i] = \frac{1}{N(i)} \sum_{j \in N(i)} x[j]$$

Similar as convolution: local connection and shift invariance, but non-linear. usually set stride > 2.

$$\frac{\partial \mathcal{L}}{\partial W^l} = \text{input}_{,\text{valid}} \, , \frac{\partial \mathcal{L}}{\partial \text{output}}$$

4.3 Unet

U-Net is widely used in image-to-image tasks, e.g. denoising, super-resolution, and segmentation.

5 Recurrent Neural Network

5.1 Sequential Model

Partial Convolution Current states only depend on the previous state

Recurrent Neural Network Introduce hidden state h to stores previous information.

$$h_t = f(h_{t-1}, x_t)$$
$$y_t = g(h_t)$$

recurrent: same f, g for different t.

5.1.1 Vanila RNN

$$h_t = \tanh(W h_{t-1} + U x_t)$$
$$y_t = V h_t$$

5.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) is designed to avoid vanishing or exploding gradients, it is capable of learning long-term dependencies.

For each block, we store a history message.

Cell state: like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

Gates: optionally let information through.

$$\begin{pmatrix} i \\ f \\ o \\ \widetilde{C}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \begin{pmatrix} W_i \\ W_f \\ W_o \\ W_c \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

Input gate i, forget gate $f : f_t = 1$ and output gate o.

Gate i, f, o control the information flow.

It gives the information flow h and the history memory flow C. So it can go deeper.

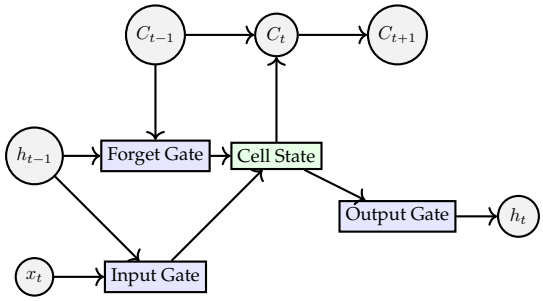

Figure 5.1: Single LSTM Cell Structure with Extended Cell State Flow

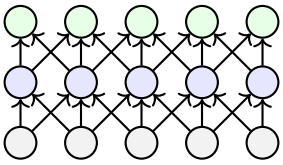

Figure 5.2: CNN Networks

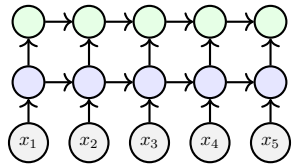

Figure 5.3: RNN Network

6 Transformer

6.1 Vision Attention

V1(Primary Visual Cortex) V1 is the first stage of visual processing in the brain. It is responsible for processing basic visual features (edges, orientation)

V1 Saliency Hypothesis V1 transforms the visual inputs into a saliency map of the visual field to guide visual attention of direction of gaze.

6.2 Context Attention

One word "attends" to other words in the same sentence differently.

Context information in different sentence is not of fixed length.

6.3 Attention

Attention in deep learning is a vector of importance weight.

Automatically search for parts of a source sentence that are relevant to predicting a target word.

Attention:

1. query-key matching

$$s_i = q_i \cdot k_i$$

2. compute weights(attention)

$$w_i = \text{softmax}(s_i)$$

3. weighted averaging

$$z \sum_i w_i v_i$$

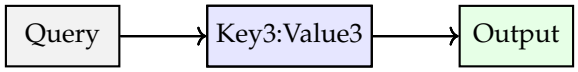

Figure 6.1: Query → Key:Value → Output

For multiple queries

1. query-key matching
$$S = QK^T$$

2. compute weights(attention)
$$W = \mathrm{softmax}(\frac{S}{\sqrt{c}})$$
For independent q_i, k_i, with mean 0 and variance q, scaled dot=product also has mean 0 and variance 1.

3. weighted averaging
$$Z = WV$$

where the queries are a $q \times c$ matrix and keys are a $k \times c$ matrix. And values are $k \times d$ matrix and output are $q \times d$ matrix.

Input $X = (x_t)_t$:

- Long Context: compute attention across the whole sequence.
- Weight W_q, Q_k, Q_v are shared by all X_t.

$$Q_t = X_t W_q \in \mathbb{R}^{n \times m}$$
$$K_t = X_t W_k \in \mathbb{R}^{n \times m}$$
$$V_t = X_t W_v \in \mathbb{R}^{n \times d}$$

It gives the outputs, which can be as the input of another attention block.

And we can define different heads (like channels in CNN) extract different information.

Index

List of Theorems