2022-11-19

Lecture 11. Kohonen maps, autoencoders, transfer learning, generative adversarial networks

Lecture 11. Kohonen maps, autoencoders, transfer learning, generative adversarial networks

Alex Avdyushenko

Kazakh-British Technical University

November 19, 2022

Good afternoon, dear students. Today we are going to talk about another interesting topic in deep learning, it is autoencoders and generative adversarial networks or simply GANs.

Lecture 11. Kohonen maps, autoencoders, transfer learning, generative adversarial networks

└Five-minutes block

Five-minutes block

- What are the main disadvantages of the encoder-decoder architecture?
- Write down the attention model formula $Attn(q, K, V)$
- Describe two criteria for BERT training

As always we start with five-minutes questions on the previous lecture. Please, write answers or send photos with them directly to me in private messages here in Teams or may be in Telegram, but choose only one option please :)

Formulation of the clustering problem

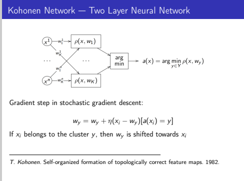
Given:
$X^\ell = \{x_1, \dots, x_\ell\}$ — training set of objects $x_i \in \mathbb{R}^n$
$\rho : X \times X \to [0, \infty)$ — distance function between objects
Find:
Y is the set of clusters, for example, given by their centers $w_y \in \mathbb{R}^n$
Let the clustering algorithm is Winner Takes All (WTA):

$$a(x) = \arg\min_{y \in Y} \rho(x, w_y)$$

Optimization criterion: average intracluster distance

$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)}) \to \min_{w_y : y \in Y}$$

But we start from afar with the formulation of the clustering problem. So you are given a training set of objects, essentially consisting of number features each.



Actually we can write the same operations in the form of special two-layer neural network. And of course you use stochastic gradient descent for optimization of our network.

Stochastic Gradient Descent

Input: sample X^ℓ, learning rate η, parameter λ
Output: cluster centers $w_1, \dots, w_K \in \mathbb{R}^n$

1. initialize centers $w_y,\ y \in Y$
2. functional estimate:

$$Q = \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)})$$

3. **repeat**
   - select object x_i from X^ℓ (e.g. random)
   - compute cluster: $y = \arg\min_{y \in Y} \rho(x_i, w_y)$
   - make a gradient step: $w_y = w_y + \eta(x_i - w_y)$
   - evaluate functional: $Q = \lambda\rho^2(x_i, w_y) + (1 - \lambda)Q$
4. **while** the value of Q and/or the weights of w_y do not converge

Ok, here we have written down all the steps of SGD in this case.

Hard and soft competition

WTA, Winner Takes All

$$w_y = w_y + \eta(x_i - w_y)[a(x_i) = y], y \in Y$$

WTA rule disadvantages

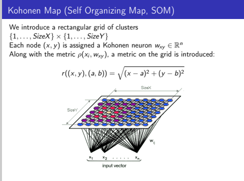
- slow convergence rate
- some cluster centers may never be selected

Soft competition rule (WTM, Winner Takes Most)

$$w_y = w_y + \eta(x_i - w_y)K(\rho(x_i, w_y)), y \in Y$$

where the kernel $K(\rho)$ is a nonnegative non-increasing function.
Now the centers of all clusters are shifted towards x_i, but the farther from x_i, the smaller the shift

Now let's compare two rules of competition. The first one is the Winner Takes All rule, we used it on previous slides. The second one is a little bit softer, the Winner Takes Most rule.



Finally, now we can formulate a new unusual approach to the clustering problem, which is called Kohonen Map (or Self Organizing Map). So we introduce a rectangular grid of clusters, where each node (x, y) is assigned to a Kohonen neuron $w_x y \in R^n$.

Kohonen Map Training

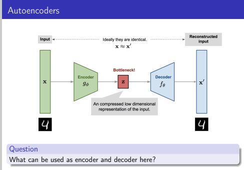
Input: sample X^ℓ, learning rate η
Output: $w_{xy} \in \mathbb{R}^n$ are weight vectors

1. initialize weights: $w_{xy} = \text{random}\left(-\frac{1}{2MH}, \frac{1}{2MH}\right)$
2. **repeat**
    - choose a random object x_i from X^ℓ
    - WTA: compute cluster coordinates:

      $$(a_i, b_i) = \arg\min_{(a,b)} \rho(x_i, w_{ab})$$

    - **for all** $(a, b) \in \text{Neighborhood}\ (a_i, b_i)$
      WTM: do gradient descent step:
      $w_{ab} = w_{ab} + \eta(x_i - w_{ab})K(r((a_i, b_i), (a, b)))$
3. **yet** clustering does not stabilize

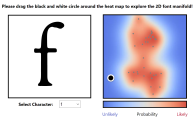
How do we train it?



Ok, great. Now let's move on to another interesting approach in machine learning. It is autoencoders and it's main idea is quite simple. So you have some input vector, for example an image. Then you have two parts of the model, which are the encoder part, where you get some representation of the input, and the decoder part, where you try to get back to the initial input vector.

2022-11-19

Lecture 11. Kohonen maps, autoencoders, transfer learning, generative adversarial networks



2d font manifold demonstration

Here let's enjoy a little demo where different fonts are nested in a 2D plane.

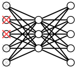
Denoising Auto Encoder

Stability of code vectors z_i with respect to noise in x_i:

$$\mathscr{L}_{DAE}(\alpha, \beta) = \sum_{i=1}^{\ell} E_{\tilde{x} \sim q(\tilde{x}|x_i)} \mathscr{L}(g(f(\tilde{x}, \alpha), \beta), x_i) \to \min_{\alpha, \beta}$$

Instead of calculating the expectation $E_{\tilde{x}}$ in the stochastic gradient method, x_i objects are sampled and noisy one at a time: $\tilde{x} \sim q(\tilde{x}|x_i)$

- Gaussian noise: $\tilde{x} \sim N(x_i, \sigma^2 I)$
- zeroing components of vector x_i with probability p_0



P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. ICML-2008.

Ok, now let's discuss briefly a few more autoencoders. For a example, the denoising one.

Variational Auto Encoder

A generative model is constructed capable of generating new objects x similar to the objects of the sample $X^\ell = \{x_1, \dots, x_\ell\}$
$q_\alpha(z|x)$ — probabilistic encoder with α parameter
$p_\beta(\hat{x}|z)$ — probabilistic decoder with β parameter

$$\mathscr{L}_{VAE}(\alpha, \beta) = \sum_{i=1}^{\ell} \log p(x_i) = \sum_{i=1}^{\ell} \log \int q_\alpha(z|x_i) \frac{p_\beta(x_i|z)p(z)}{q_\alpha(z|x_i)} dz \geq$$

$$\geq \sum_{i=1}^{\ell} \int q_\alpha(z|x_i) \log p_\beta(x_i|z) dz - KL(q_\alpha(z|x_i) \| p(z)) \to \max_{\alpha,\beta}$$

D.P.Kingma, M.Welling. Auto-encoding Variational Bayes. 2013.
C.Doersch. Tutorial on variational autoencoders. 2016.

The next one is Variational Auto Encoder and it may be difficult to understand from the first view, but it's main feature is to use lower variational bound of log likelihood of your model.

Lecture 11. Kohonen maps, autoencoders, transfer learning, generative adversarial networks

$$\sum_{i=1}^{\ell} \underbrace{E_{z\sim q_\alpha(z|x_i)} \log p_\beta(x_i|z)}_{\text{quality reconstruction}} - \underbrace{KL(q_\alpha(z|x_i)\|p(z))}_{\text{regularizer by } \alpha} \to \max_{\alpha,\beta}$$

where $p(z)$ is the prior distribution, usually $N(0, \sigma^2 I)$

Reparametrization $q_\alpha(z|x_i):\ z = f(x_i, \alpha, \varepsilon),\ \varepsilon \sim N(0, I)$

Stochastic gradient method:

- sample $x_i \sim X^\ell,\ \varepsilon \sim N(0, I),\ z = f(x_i, \alpha, \varepsilon)$
- gradient step $\alpha = \alpha + h\nabla_\alpha[\log p_\beta(x_i|f(x_i, \alpha, \varepsilon)) - KL(q_\alpha(z|x_i)\|p(z))]$
  $\beta = \beta + h\nabla_\beta[\log p_\beta(x_i|z)]$

Generation of similar objects:

$$x \sim p_\beta(x|f(x_i, \alpha, \varepsilon)), \varepsilon \sim N(0, I)$$

So we two parts of functional, and first one if responsible for quality reconstruction, second one is actually regularizer by α.

To evaluate the expactation of some function with z distribution you need to use reparametrization trick. What is it? On the first step you sample z-value as value of new function f with new parameter ϵ, which comes from some random distribution, for example normal. And then on the second step you get the expectation with your z-value.

Multi-task learning

- $f(x, \alpha)$ — universal part of the model (vectorization)
- $g(x, \beta)$ — specific parts of the model for problems $t \in T$

Simultaneous training of the model f on tasks $X_t, t \in T$:

$$\sum_{t \in T} \sum_{i \in X_t} \mathscr{L}_{ti}(f(x_{ti}, \alpha), g(x_{ti}, \beta_t)) \to \min_{\alpha, \{\beta_t\}}$$

Learnability: the quality of solving a particular problem $\langle X_t, \mathscr{L}_t, g_t \rangle$ improves with increasing sample size $\ell_t = |X_T|$.

Learning to learn: the quality of the solution of each of the problems $t \in T$ improves with the growth of ℓ_t and the total number of problems $|T|$.

Few-shot learning: a small number of examples, sometimes even one, is enough to solve the t problem.

M.Crawshaw. Multi-task learning with deep neural networks: a survey. 2020

Y.Wang et al. Generalizing from a few examples: a survey on few-shot learning. 2020

The hot topic of the last couple of years is exactly multitask and multi-domain training. So we want to have one AI model, which is good for all tasks =) Something like another one step on the way to general AI.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. A
was some unprocessed data that should have been added to the
this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page
away, because LaTeX now knows how many pages to expect for t
document.