Stanford CS231n: Convolutional Neural Networks

James Zhang[*]

July 2, 2023

*Email: jzhang72@terpmail.umd.edu

Contents

§1 Image Classification

An image classification problem can be solved with **K Nearest Neighborhood** classficiation algorithm but it can perform very poorly where the properties of KNN are

- k is the number of neighbors we compare
- the distance is either measured by Euclidean distance or Manhattan distance

Linear **support vector machines** is also an option for image classifcation, but it's difficult to scale. **Logistic regression** is also a solution, but image classficiation is not necesarily linear.

§2 Loss Functions

In the last section, we talk about **linear classifiers** but we don't discuss training the parameters of the models. We need a loss function to indicate how good or bad our current parameters are.

$$L_i = (f(X_i, W), Y_i)$$

such that the loss of the *ith* example is a function of our hypothesis - which itself is a function of our example and our weights - and the label. We can denote the average loss of the batch as

$$L_{batch} = \frac{1}{N} \times \sum_{i=1}^{N} L_i$$

where N is the number of examples in a batch. Minimizing this loss function is known as optimization.

Definition 2.1. The loss function for a linear SVM classifier is called the **hinge loss** (L1-SVM). Let $t = \pm 1$ be the intended output, and y be the prediction. The hinge loss of one example is

$$l(y) = \max(0, 1 - t \cdot y)$$

Note that if the model predicted the output correctly, the hinge loss value is 0.

There also exists the **squared hinge loss** (L2-SVM) that penalizes violated margins quadratically instead of linearly. Most of the time, unsquared hinge loss works better, though.

Definition 2.2. We add **regularization** such that the discovered model doesn't overfit the data.

$$L = \frac{1}{N} \times \sum_{i=1}^{N} L_i + \lambda \times R(W)$$

where R is the regularizer, and λ is the regularizaiton term.

- For L2, $R(W) = \sum_{i=1}^{N} W^2$
- For L1, $R(W) = \sum_{i=1}^{N} |W|$

Regularizations are also called weight decays.

Definition 2.3. We also have a softmax loss function for softmax regression, which recall is a generalization of logistic regression for more than two classes. Recall that the softmax function is the following

$$P(y = j|x) = \frac{e^{w_j^T x}}{\sum_{i=1}^{N} e^{w_i^T x}}$$

and we use the log loss or cross entropy loss function

$$L_i = -\log P(Y = y_i | X = x_i; w_i)$$

§3 Optimization

The key question in this section is how can we optimize the loss functions that we have discussed?

- Generate random parameters and try all of them on the loss and choose the best one in a Monte Carlo fashion.
- The better approach is to take the gradient and using calculus minimize our loss function.

$$w := w - \alpha * \nabla \frac{J(w)}{w}$$

§4 Convolutional Neural Networks

§4.1 History of Neural Networks

- The first perceptron machine was developed in 1957, and it was used to recognize the letters of the alphabet. Back propagation wasn't developed yet.
- The multilayer perceptron was develped in 1960. Back propagation wasn't developed yet.
- Back propagation was developed in 1986.
- For a while, no improvements were made to neural networks due to the limited computing resources and data.

§4.2 History of Convolutional Neural Networks

- Convolutional neural networks were introduced in 1998 in a paper called "Gradient-based learning appplied to document recognition."
- In 2012, AlexNet used the architecture with a larger data set and a GPU and won the image net challenge, and this propelled CNN architectures to become far more widely applied in tasks such as iamge classficiation, object detection, segmentation, face recognition, medical images, iamge captioning, and more.

§4.3 Traditional CNN Layers

CNN architectures make the explicit that the inputs are images, which allow us to encode certain properties into the architecture. There are a few distinct layers in CNNs, such as convolution, fully connected, rectified linear unit (ReLU), and pooling.

Definition 4.1. A **fully connected** layer is a layer in which all neurons are connected. Sometimes we call this a **dense** layer.

If the input shape is (X, M), then the weights shape for this will be (X, N), where N is the number of hidden neurons in the hidden layer.

Definition 4.2. A **convolution** layer is a layer in which we a apply filters of weights that go all through the image and maintain the structure of the input. We do this with the dot product $w^T x + b$ where we train and try to learn the optimal values for parameters w and b. The output of a convolution layer is known as an **activation map**.

Note 4.3. Note that we need to have multiple activation maps.

Example 4.4

Suppose we have 6 filters in our convolution layer and the input shapes are $(32, 32, 3)$. Suppose our filter sizes are $(5, 5, 3)$. Note that the depth of the filter must be 3 because the input depth is 3. Applying the filters across the entire image, note that our output shape will be $(28, 28, 6)$. If we only had one filter, then it would be $(28, 28, 1)$. After ReLU, the output shape is still $(28, 28, 6)$. Applying another convolution layer where the filters are now of shape $(5, 5, 6)$ to account for the new depth, the output shape would be $(24, 24, 10)$.

§5 Deep Learning Hardware and Software

§6 Training Neural Networks I

§7 CNN Architectures

§8 Generative Models

§9 Detection and Segmentation

§10 Visualizing and Understanding