NYCU Pattern Recognition, Homework 4

109550198, 卜銳凱

Part. 1, Kaggle (70% [50% comes from the competition]):

(10%) Implementation Details

Type your answer here.

It loads and pre-processes from these specified paths: ./data/train and ./data/test. The load_data function reads bags of images together with their respective labels for training, a representation of bags of images and their IDs for the test set. Resizing, flipping, rotation, conversion to tensor, and normalization of the images are implemented.

Pre-trained ResNet50 is fine-tuned for the model architecture. The last fully connected layer was adopted with 512 outputs, followed by Dropout with a 0.5 probability, and a final fully connected layer with 1 output. The activation function used for that layer is Sigmoid for binary classification.

The dataset was divided into 80% training and 20% validation sets. Data is loaded in the DataLoader in the batch_size of 8. The loss function is the Binary Cross-Entropy Loss. The optimizer used is Adam with a learning rate of 1e-6. A StepLR scheduler was used to adjust the learning rate with a step size of 7 and gamma of 0.1. Class weights are calculated based on class imbalance. The model is trained for 100 epochs, logging of loss and accuracy, and saving of best model weights based on validation performance.

For prediction, the best model's weights are loaded from ./trained_classifier.pth, then predictions are made on the test set, and a submission file, listing image IDs and predicted labels, is created in the required format. In total, the process ensures training and effective classification of image bags.

(10%) Experimental Results

Type your answer here.

The experimental results of the model for image classification with image bags are quite good. The dataset is partitioned into 80% for training and 20% for validation. The images are resized, flipped, rotated, and normalized together to give uniformity. The model was trained with data through 100 epochs; hence, this enabled it to learn best and generalize very well on the validation set. Predictions over the test set were like the validation results. A submission file was generated that listed each of the test image IDs and their corresponding predicted labels in the required format. Data augmentation, plus class weight augmentation for handling class imbalance, further improved the performance of the model. Future improvements can lie in two possibilities: wiggling hyperparameters, or changing architectures, or maybe adding other data augmentation to strengthen the model even more.

Part. 2, Questions (30%):

1. (10%) Why Sigmoid or Tanh is not preferred to be used as the activation function in the hidden layer of the neural network? Please answer in detail.

   Type your answer here.

   Activation functions such as Sigmoid or Tanh are typically not good for use in hidden layers, as they often result in the vanishing gradient problem. Small input values result in very small gradients, eventually leading to slow learning. Sigmoid outputs will not be zero-centered, which makes some of its gradients be always of one sign, creating imbalanced gradients and oscillations between the positive and negative regions of the data during training. These functions also present a problem of gradient saturation, which is related to poor learning characteristics. Instead, the most common activation functions used are ReLU and its variants Leaky ReLU and Parametric ReLU, along with ELU, as they help in the better maintenance of gradients and avoiding vanishing gradients. This then facilitates faster and more stable training.

2. (10%) What is overfitting? Please provide at least three techniques with explanations to overcome this issue.

   Type your answer here.

   Overfitting is the problem facing the model that knows too well the training data, including noise and details not general enough for application to new data. In this concern, the model will often demonstrate abnormally the best accuracy in the training dataset but very weak accuracy in the validation or test dataset. The following are three techniques to overcome overfitting:

   1. Regularization
   A class of approaches known as regularization methods adds a penalty to the complexity of the model to prevent it from fitting the noise in the input.

   L1 Regularization (Lasso): It imposes an absolute value of coefficients as a penalty term into the loss function. This can drive some coefficients to zero, doing feature selection.

   L2 Regularization (Ridge): It helps in adding squared value of coefficients as a penalty term to the loss function. It helps to reduce the model complexity, small coefficients that cannot overfit.

   2. Data Augmentation
   Data augmentation is the act of using different transformations over your training data to produce more examples from that data. This generally happens in a fashion that would increase the diversity of our set of examples and therefore aid the model in generalizing more accurately.

   Image Augmentation: In the case of imaging data, this includes random cropping, flipping, rotation, scaling, and color jittering.

Addition of Noise: By adding noise to the input data, it allows the model to be more robust against small variations in the data.

3. Dropout
Dropout is basically one of the regularization techniques where, during the training time, a randomly selected sub-set of neurons is ignored. This will make the network not rely much on any one neuron, hence forcing the learning of more robust features.

How it works: dropout is applied during training by randomly setting a proportion of input units to 0. The fraction set to 0 is defined by a hyperparameter, usually denoted as the dropout rate; a typical value is 0.5. No neurons are dropped during inference, but the weights are scaled to balance for the dropout that was applied in training.
Various techniques reduce overfitting by keeping the model simple and by widening data diversity. They do not allow the net to get too dependent on some other neurons or features.

3. (15%) Given a valid kernel $k_1(x, x')$,prove that the following proposed functions are or are not valid kernels. If one is not a valid kernel, give an example of $k(x, x')$ that the corresponding K is not positive semidefinite and show its eigenvalues.

a. $k(x, x') = k_1(x, x') + \|x\|^2$

b. $k(x, x') = k_1(x, x') - 1$

c. $k(x, x') = k_1(x, x') + \exp(x^T x')$

d. $k(x, x') = \exp\big(k_1(x, x')\big) - 1$

Type your answer here.

a. $k(x,x')=k_1(x, x')+\|x\|^2$

To determine if $k(x,x')=k_1(x,x')+\|x\|^2$ is a valid kernel:

1. **Symmetry**:
   - $k_1(x,x')$ is symmetric by assumption.
   - $\|x\|^2$ is a term that only depends on x and not on x′, so it does not affect the symmetry.
   - Therefore, k(x,x′) is symmetric since $k(x,x') = k_1(x,x') + \|x\|^2 = k_1(x',x)+ \|x\|^2=k(x',x)$.
2. **Positive Semidefiniteness**:
   - To check for positive semidefiniteness, we consider the Gram matrix formed by $k(x_i,x_j)$.
   - The Gram matrix for k(x,x′) is $[k_1(x_i,xj)+\|x_i\|^2]$.
   - Since ‖x‖2 is a constant added to each row of the Gram matrix, it does not affect the eigenvalues related to positive semidefiniteness.
   - If $k_1(x,x')$ is positive semidefinite, adding a non-negative term $\|x\|^2$ preserves positive semidefiniteness.

Therefore, $k(x,x')=k_1(x,x') + \|x\|^2$ is a valid kernel.

b. $k(x,x')=k1(x,x')-1$

1. **Symmetry**:
   - $k_1(x,x')$ is symmetric by assumption.
   - Subtracting 1 does not affect the symmetry: $k(x,x')=k_1(x,x')-1=k_1(x',x)-1=k(x',x)$.
2. **Positive Semidefiniteness**:
   - Consider the Gram matrix for $k(x,x')$, which is $[k_1(x_i,x_j)-1]$.
   - Subtracting 1 from each element of the Gram matrix does not guarantee that the matrix remains positive semidefinite. This operation can lead to negative eigenvalues.
   - For example, if $k_1(x,x')$ is the linear kernel $k_1(x,x')=x \cdot x'$ and $x=[1,-1]$, then:

$$K = \begin{pmatrix} k_1(1,1)-1 & k_1(1,-1)-1 \\ k_1(-1,1)-1 & k_1(-1,-1)-1 \end{pmatrix} = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$$

   - The eigenvalues of this matrix are ± 2 , which includes a negative eigenvalue, showing it is not positive semidefinite.

Therefore, $k(x,x')=k1(x,x')-1$ is not a valid kernel.

c. $k(x,x')=k_1(x,x')+exp(x^Tx')$

1. **Symmetry**:
   - $k_1(x,x')$ is symmetric by assumption.
   - $\exp(x^Tx')$ is symmetric because $\exp(x^Tx')=\exp((x')^Tx)$.
   - Therefore, $k(x,x')=k_1(x,x')+exp(x^Tx')$ is symmetric.

2. **Positive Semidefiniteness**:
   - The Gram matrix for $k(x,x')$ is $[k_1(x_i,x_j)+exp(x_i^Tx_j)]$.
   - Since $exp(x_i^Tx_j)$ is always positive and $k_1(x,x')$ is positive semidefinite, adding these terms should preserve positive semidefiniteness.

Therefore, $k(x,x')=k_1(x,x')+exp(x^Tx')$ is a valid kernel.

d. $k(x,x')=exp(k_1(x,x'))-1$

1. **Symmetry**:
   - $k_1(x,x')$ is symmetric by assumption.
   - $exp(k_1(x,x'))$ is symmetric because the exponential function is symmetric:

     $exp(k_1(x,x'))=exp(k_1(x',x))$.

   - Therefore, $k(x,x')=exp(k_1(x,x'))-1$ is symmetric.

2. **Positive Semidefiniteness**:

- The positive semidefiniteness of the Gram matrix for $k(x,x')=exp(k_1(x,x'))-1$ depends on the nature of k_1.
- Since $exp(k_1(x,x'))$ is positive semidefinite if $k_1(x,x')$ is positive semidefinite, subtracting 1 can introduce negative eigenvalues.

Therefore, $k(x,x')=exp(k_1(x,x'))-1$ is not guaranteed to be a valid kernel. For instance, if $k_1(x,x')=ln(1+x \cdot x')$, the resulting Gram matrix can have negative eigenvalues after subtracting 1.