Regularizations

- Regularizations
  - L1: Lasso
    - Function: $CostFunction = Loss + \frac{\lambda}{2m} * \sum ||w||$
    - penalize the absolute value of the weights
    - useful for feature engineering
      - Explainable: select the important features whose weights do not equal to 0, while deletes those unimportant features' weights are equal 0
    - Laplace distribution
  - L2:Ridge
    - Function: $CostFunction = Loss + \frac{\lambda}{2m} * \sum ||w||^2$
      - λ: regularization parameter
    - Also called *weight decay* 权重衰减, owing to it forces the weights to decay towards 0.
    - Gaussian distribution
  - Coding in Keras:
    - `from Keras import regularizers`
    - `model.add(Dense(64,input_dim=64,kernel_regularizer=regularizers.l2(0.01)`
  - Dropout(more useful in DL)
    - Concepts: At every iteration, it randomly selects some nodes, and removes them along with all of their incoming and outgoing connections.
      - use **backpropagation** to learn and update the parameters in neural network, while the nodes which are deleted not engage in the update progress.
    - the hyperparamter of dropout function: the probability(p) to choose how many nodes should be dropped.
    - Apply to hidden layers and input layers.
    - Ensemble learning
    - Coding:
      - `from keras.layers.core import Dropout`
      - `model = Sequential([ Dense(output_dim=hidden1_num_units, input_dim=input_num_units, activation='relu'),`
        - `Dropout(0.25),` # we define $p = 0.25$ as the probability of dropping
        - `Dense(output_dim=output_num_units, input_dim=hidden5_num_units, activation='softmax'), ])`

- Data Augmentation
  - increase the size of the training data
- Early Stopping
  - cross-validation strategy, keep one part of training dataset as validation set
  - If we see the performance of validation set is getting worse, we can stop the training on model
  - Coding
    - `from keras.callbacks import EarlyStopping`
    - `EarlyStopping(monitor='val_err', patience=5)`
      - **monitor** : the quantity that needs to be monitored
      - '**val_err**' denotes the validation error.
      - **Patience**: the number of epochs with no further improvement after which the training will be stopped.
- Concepts: to prevent the model too complex and overfitting, add penalization elements after the original loss function, improving the model's performance on the unseen data.→ `Cost Function=Loss + Regularization Term`
  - In deep learning, it penalizes the weight matrices of the nodes.
  - If the `regularization coefficient` is too high,
    - some weight matrices nearly equal zero
    - a simplier linear network
    - slight underfitting for the training data

以上内容整理于 幕布文档