COMP 5630/6630:Machine Learning

Lecture 8: MLP Demo, Project Guidelines

ML in Practice, Deep Learning

Adapted from: Drs. Dhruv Batra, Aakur

Today’s Class

- ML in Practice
- Regularization in Neural Network in Practice
- Deep Learning

ML Pipeline

ML Pipeline

- Data Preprocessing
- Feature Engineering
- Model Selection
- Model Training
- Model Evaluation
  - Model Optimization

ML Pipeline

- Data Preprocessing
- Feature Engineering
- Model Selection
- Model Training
- Model Evaluation
  - Model Optimization

- Independent variables
  - Variable do not affected by other variables in the model
- Dependent variable
  - Values are dependent on other variables
  - Example: prediction output

Data Preprocessing

- Data Cleaning
- Handling missing values and imbalanced data
- Feature Engineering
- Splitting
- Scaling and normalization

Data Preprocessing

- Data Cleaning
  - Check formatting
  - Example: strings to numeric categories, parsing dates

Data Preprocessing

- Handling missing values: identify missing values
  - dataframe.isnull().values.any()
    - returns True when there is at least one missing value occurring in the data
  - df[df['height'].notnull]
    - Returns whether the column has any rows with missing values

Data Preprocessing

- Handling missing values: handling missing values
  - Throw out the entire row if any value is missing
  - Replace missing values with mean/median

Data Preprocessing

- Handling imbalanced data
    - Most ML algorithms prefer balanced dataset. However, imbalanced data is common.
    - Approaches to handle imbalanced data
        - Balanced random sampling
            - To make the data balance, randomly throw out instances from the majority class
        - Synthetic data generation
            - Generate synthetic data from the minority class. Example algorithm: SMOTE
        - Accept the imbalance dataset distribution ☺

Data Preprocessing

- Feature Engineering
  - Feature Selection
    - Relevant features for the task at hand
    - Reliable features to make prediction
  - Feature Construction
    - Refactor features
  - Goal for feature selection
    - Eliminate spurious information
    - Enhancing efficiency
    - Reduce multicollinearity among features
      - multicollinearity : Feature variables are correlated in a model

Data Preprocessing

- Feature Selection
  - Statistical correlation
    - Identifying and removing features that are highly correlated with each other.
  - Information gain
    - Select features with high information gain
  - Automatic feature selection: scikit-learn library
    - **SelectKBest:** selects k best feature
    - **RFE (Recursive Feature Elimination):** starts will all features, recursively eliminates features until a specified number of features remain.

Data Preprocessing

- Data splitting
  - Training/test/validation
  - K-fold cross validation

Data Preprocessing

- Data Scaling and Normalization
  - Z-score: **sklearn.preprocessing.StandardScaler**
  - Min max normalization: **sklearn.preprocessing.MinMaxScaler**

- Apply normalization on training set (not on test set)
- Apply learned normalization to test set

from sklearn.preprocessing

- *fit_transform() on training data*
- *transform() on the test data*

Model Selection

- Depends on the task at hand
  - Supervised learning: classification, regression
    - Example: predicting presence of a bug in a software
    - Predicting temperature
  - Unsupervised: Clustering
    - Example: Identifying patterns of behavior between high and low performing students
- Typically assess a series of models
  - For binary classification, one can evaluate logistic regression, decision trees, MLP, ...

Model Evaluation

- Regression
  - Root Mean Squared Error (RMSE)
  - Mean Absolute Error (MAE)
  - The lower RMSE and MAE, the better is model
- Classification
  - Accuracy
  - Precision
  - Recall
  - F1 score
  - AUC-ROC
  - The higher are the classification metrices, the better model performance.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- https://www.aporia.com/learn/root-mean-square-error-rmse-the-cornerstone-for-evaluating-regression-models/

Model Evaluation: Optimization

- Hyperparameter Tuning: Selecting the optimal set of parameters for ML model.

- Approaches for hyperparameter tuning
  - Grid Search: Systematically explores a predefined set of parameter values, effectively creating a grid of possible configurations.

  - Random search: similar to grid search, but instead of using all the points in the grid, it tests only a randomly selected subset of these points.

  - Example parameters
    - Neural network: Hidden layers, learning rate, kernels etc.
    - LR: solver, penalty, regularization

Model Evaluation & Optimization: Hyperparameter Tuning

- Hyperparameter Tuning Example: Logistic regression
- solver= {***‘liblinear’, ‘newton-cg’***}
  - [depends on binary/multiclass LR. See documentation.]
- penalty = {L2, L1, None}

- Grid search= 2 x 3 = 6 parameter combinations
- = {***‘liblinear’, L1***} , {***‘newton-cg’, None***} ,
- = {***‘liblinear’, L2***} , { ***‘newton-cg’, None***}
- = {***‘liblinear’, None***} , { ***‘newton-cg’, None***}

Model Evaluation & Optimization: Hyperparameter Tuning

- Hyperparameter Tuning Example: Logistic regression

- Grid search
  - Train LR with each of these 6 parameter combination.
  - Test on the validation set
  - Select the parameter combination which gives the best result

- Advantages
  - Finds the best combination of parameters

- Disadvantages
  - Exhaustive search, search space grows with number of parameters and values

Model Evaluation & Optimization: Hyperparameter Tuning

- Hyperparameter Tuning Example: Logistic regression

- Random search
  - Randomly select a subset (say, 4) parameter combinations from the 6 parameter combinations
  - Train LR with each of these parameter combination.
  - Test on the validation set
  - Select the parameter combination which gives the best result

- Advantages
  - Faster than grid search
  - The larger this dataset, the more accurate the optimization but the closer to a grid search.

- Disadvantages
  - The smaller this subset, the faster but less accurate the optimization.

Regularization in Practice: Neural Network

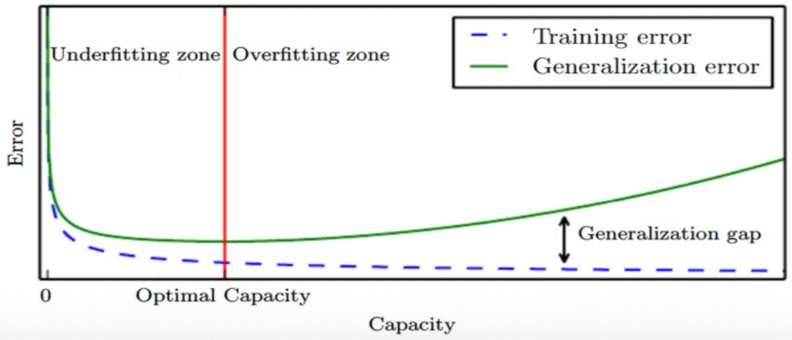
NN regularization?

Generalization error

- Performance on inputs not previously seen
  - Also called as *Test error*

Regularization is:

- "any modification to a learning algorithm to reduce its *generalization error* but not its *training error*"
- Reduce generalization error even at the expense of increasing training error
  - E.g., Limiting model capacity is a regularization method



Why regularization?

Generalization

- Prevent over-fitting

Occam's razor

Bayesian point of view

- Regularization corresponds to prior distributions on model parameters

Regularization: Limiting Number of Neurons

No. of input/output units determined by dimensions

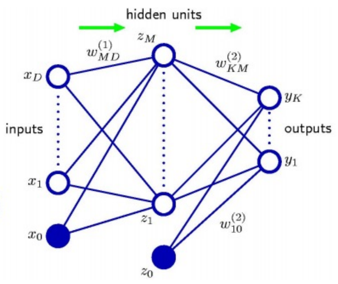
Number of hidden units M is a free parameter

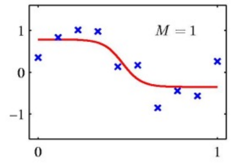
- Adjusted to get best predictive performance

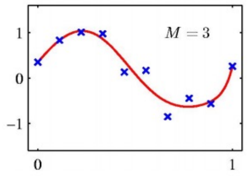
Possible approach is to get maximum likelihood estimate of M for balance between under-fitting and over-fitting

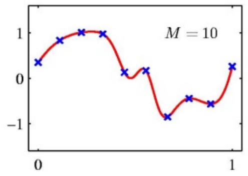
Regularization: Limiting Number of Neurons

Sinusoidal Regression Prob









Two layer network trained on 10 data points

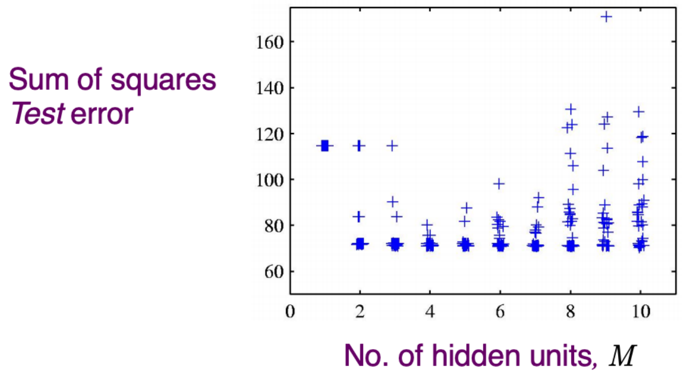
$M = 1, 3$ and 10 hidden units

Minimizing sum-of-squared error function
Using conjugate gradient descent

Generalization error is not a simple function of M due to presence of local minima in error function

Using Validation Set to fix no. of hidden units

Plot a graph choosing random starts and different numbers of hidden units M and choose the specific solution having smallest generalization error



- 30 random starts for each M
  30 points in each column of graph
- Overall best *validation* set performance happened at $M=8$

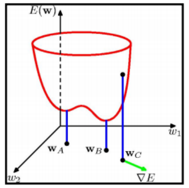
There are other ways to control the complexity of a neural network in order to avoid over-fitting

Alternative approach is to choose a relatively large value of M and then control complexity by adding a regularization term

Parameter Norm Penalty

Generalization error not a simple function of M

- Due to presence of local minima
- Need to control capacity to avoid over-fitting
  - Alternatively choose large M and control complexity by addition of regularization term



Simplest regularizer is *weight decay*

$$\boxed{\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}}$$

  - Effective model complexity determined by choice of λ
  - Regularizer is equivalent to a Gaussian prior over $\boldsymbol{w}$

In a 2-layer neural network

$$J_{regularized} = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right)\right)}_{\text{cross-entropy cost}} + \underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_l\sum_k\sum_j W_{k,j}^{[l]2}}_{\text{L2 regularization cost}}$$

Regularization: Weight Decay

The name weight decay is due to the following

$$\boxed{w_{t+1} = w_t - \alpha \nabla_w J - \lambda w_t}$$

To prevent overfitting, every time we update a weight $\boldsymbol{w}$ with the gradient ∇J in respect to $\boldsymbol{w}$, we also subtract from it $\lambda \boldsymbol{w}$.

This gives the weights a tendency to decay towards zero, hence the name.

Deep Neural Networks

What is Deep Learning?

1. Computational models composed of multiple processing layers
    - To learn representations of data with multiple levels of abstraction
2. Dramatically improved state-of-the-art in:
    - Speech recognition, Visual object recognition, Object detection
    - Other domains: Drug discovery, Genomics
3. Discovers intricate structure in large data sets
    - Using backpropagation to change parameters
    - Compute representation in each layer from previous layer
4. Deep convolutional nets: image proc, video, speech
5. Recurrent nets: sequential data, e,g., text, speech

Limitations of Conventional ML

- Limited in ability to process natural data in raw form
- Pattern Recognition and Machine Learning systems require careful engineering and domain expertise to transform raw data, e.g., pixel values, into a feature vector for a classifier

Automatic Representation Learning

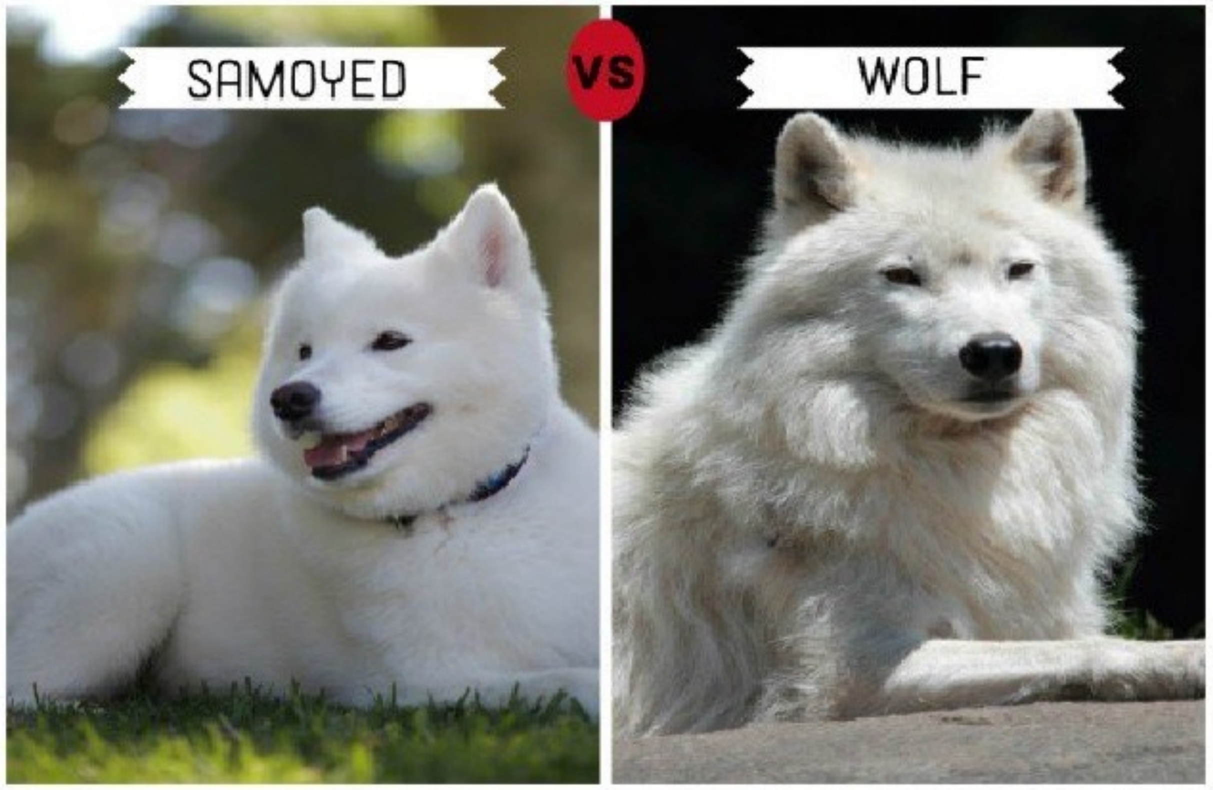
- Methods that allow a machine to be fed with raw data to automatically discover representations needed for detection or classification
- Deep Learning methods are Representation Learning Methods
- Use multiple levels of representation
  - Composing simple but non-linear modules that transform representation at one level (starting with raw input) into a representation at a higher slightly more abstract level
  - Complex functions can be learned
  - Higher layers of representation amplify aspects of input important for discrimination and suppress irrelevant variations

Example: Images

- Input is an array of pixel values
  - First stage is presence or absence of edges at particular locations and orientations of image
  - Second layer detects motifs by spotting particular arrangements of edges, regardless of small variations in edge positions
  - Third layer assembles motifs into larger combinations that corresponds to parts of familiar objects
  - Subsequent layers would detect objects as combinations of these parts
- Key aspect of deep learning:
  - These layers of features are not designed by human engineers
  - They are learned from data using a general purpose learning procedure

Deep versus Shallow Classifiers

- Linear classifiers can only carve the input space into very simple regions
- Image and speech recognition require input-output function to be insensitive to irrelevant variations of the input,
    - e.g., position, orientation and illumination of an object
    - Variations in pitch or accent of speech
    - While being sensitive to minute variations, e.g., white wolf and breed of wolf-like white dog called Samoyed
    - At pixel level two Samoyeds in different positions may be very different, whereas a Samoyed and a wolf in the same position and background may be very similar



Selectivity-Invariance dilemma

- Shallow classifiers need a good feature extractor
- One that produces representations that are:
  - selective to aspects of image important for discrimination
  - but invariant to irrelevant aspects such as pose of the animal
- Generic features (e.g., Gaussian kernel) do not generalize well far from training examples
- Hand-designing good feature extractors requires engineering skill and domain expertise
- Deep learning learns features automatically

- https://medium.com/@Tms43/understanding-padding-strides-in-convolutional-neural-networks-cnn-for-effective-image-feature-1b0756a52918