ECE 469/ECE 568 Machine Learning

Textbook:
Machine Learning: a Probabilistic Perspective by Kevin Patrick Murphy

Southern Illinois University

September 4, 2024

Data preprocessing in machine learning

- Preprocessing data in machine learning is an important first step.
- Data preprocessing ensures that the ML models are trained on high-quality and relevant data.
- Data preprocessing has several steps:
  - Data cleaning: Handling missing data entries and correcting outliers by filling in or omitting gaps in data and correcting outliers to prevent bias and improve model reliability.
  - Feature scaling: Normalizing/standardizing data to a uniform range to make model training more efficient.
  - Feature encoding: Transforming categorical data into a machine-readable formats.
- Accurate data preprocessing increases the accuracy of ML model training, while reducing the run-time and computational resources needed for training.

Data preprocessing in machine learning: Data cleaning

- Data cleaning is crucial in ML model training as it ensures data-set is free of errors and inconsistencies that could potentially skew results.
- Data cleaning involves:
  - Handling missing data entries:
    - Assign missing data entries by using statistical methods (mean, median) for numerical data
    - Impute missing data entries by mode for categorical data
    - Remove rows and columns with excessive missing values to maintain data integrity.
  - Format standardizing:
    - Make the data entries consistent across the data-set such as date formats, capitalization, and units of measurement
    - This helps avoiding discrepancies during training, testing and predicting stages.
  - Correcting outliers:
    - Outliers must be identified through statistical tests or visualization techniques
    - Outliers must be corrected using transformation, capping, and removal to prevent distortion of the data entries.

Data preprocessing in machine learning: Feature scaling

- Feature scaling is an important step prior to training a machine learning model.
- This ensures that features are within the same scale.
- There are two widely used feature scaling techniques.
  - Feature normalization
  - Feature standardization
- Feature scaling typically is required when the machine learning algorithm requires computing the gradients.
- For example, in training linear/logistic regression models and artificial neural networks, we will compute gradients.
- Having different scales for each feature may jeopardize the convergence of gradient descent algorithms.
- Scaling may not be so much important for distance-based and tree-based machine learning algorithms such as K-means clustering, K-nearest neighbor, support vector machine classifiers, decision trees, and random forests.

Data preprocessing in machine learning: Feature normalization

- Feature normalization is a scaling technique in which values are shifted and rescaled such that they end up ranging between 0 and 1.
- Let us consider a feature value, x. Its normalized value can be computed via Min-Max scaling as

$$x_{\mathrm{n}} = \frac{x - x_{\mathrm{min}}}{x_{\mathrm{max}} - x_{\mathrm{min}}},$$

  where x_{min} and x_{max}, respectively, are the minimum and maximum values of the particular feature x in the training data set.
- If $x = x_{\mathrm{min}}$, then $x_n = 0$.
- If $x = x_{max}$, then $x_n = 1$.
- Thus, the normalized feature will have a range:

$$0 \leq x_n \leq 1$$

Data preprocessing in machine learning: Feature standardization

- Feature standardization another scaling technique in which the values are centered around the mean with a unit standard deviation.
- Thus, after standardization, a feature will have a zero mean and a unit standard deviation.
- Let us consider a feature value, x. The standardized value of x can be computed as

$$x_s = \frac{x - m_x}{\sigma_x}$$

where m_x is the sample mean of the feature x, while σ_x is its unbiased sample standard deviation.

$$m_x = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{and} \qquad \sigma_x = \sqrt{\frac{1}{(N-1)}\sum_{i=1}^{N}(x_i - m_x)^2}$$

where N is size of the training data set.

- Note that unlike feature normalization, the standardization does not restrict the scaled feature to lie in a particular range.

Data preprocessing in machine learning: Feature normalization vs. standardization

- A normalized data-set will always range from 0 to 1.
- A standardized data-set will always have a zero mean and unit standard deviation. However, it can have any upper and lower bounds.
- Normalization versus standardization: There is no definite answer and the choice of scaling technique depends on the nature of the data sets, algorithm, or applications.
- Normalization is used when we do not assume a distribution for the data set, for example in training an artificial neural networks.
- Standardization is typically preferred when you know that the data set would follow a certain probability distribution, e.g. Gaussian.
- However, standardization is preferred over normalization when there are a lot of outliers in the data set.
- This is because unlike normalization, standardization does not have a bounding range. Thus, even if we have outliers in the data set, they will not be affected by standardization.

Data preprocessing in machine learning: Feature normalization vs. standardization

- If you do not have a complete understanding on a given data-set, you may start by fitting your machine learning model to raw, normalized and standardized data and compare the performance for the best performance.
- Scaling of target/output values is generally not required.
- It is a good practice to fit the scaling technique on the training data and then use it to transform the testing data.
- This practice will avoid any data leakage during the model testing phase.

Data preprocessing in machine learning: Feature encoding

- The features in a data-set may have both numerical and categorical data entries.
- Typically, machine learning models may only work with numeric.
- It is necessary to transform categorical features into numerical formats.
- Feature encoding deals with converting categorical data into numeric.
- Feature encoding can be done via the follwing techniques:
  - One-hot encoding
  - Ordinal encoding
  - Target-mean encoding
  - Frequency encoding

Data preprocessing in machine learning: Feature encoding

- Feature encoding can be done via One-hot encoding.
  - One-hot encoding: This technique assigns vectors to each categorical feature columns in a dataset. The vector represent whether corresponding feature is present/true (1) or not/false (0).

Numerical value	Animal
1.5	cat
3.6	cat
42	dog
7.1	crocodile

One hot encoding →

Numerical value	Cat	Dog	Tiger	Crocodile
1.5	1	0	0	0
3.6	1	0	0	0
42	0	1	0	0
7.1	0	0	0	1

Data preprocessing in machine learning: Feature encoding

- Feature encoding can be done also via ordinal encoding.
  - If the categorical feature is in ordinal format, then it has a ranked order for its values. Hence, such ordinal features can be converted to numerical values through ordinal encoding.
  - In ordinal encoding, each unique category/ranked value is assigned an integer value.

Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

Data preprocessing in machine learning: Feature encoding

- Feature encoding can be done also via target-mean encoding:
  - Target-mean encoding replaces categorical values with the mean value of the target variable.

Numerical value	Animal	Good
1.5	cat	0
3.6	cat	1
42	dog	1
7.1	crocodile	1

Target mean encoding →

Numerical value	Animal_target_mean	Good
1.5	0.5	0
3.6	0.5	1
42	1	1
7.1	1	1

Data preprocessing in machine learning: Frequency encoding:

- Feature encoding can also be done via frequency encoding:
  - Frequency encoding takes into account how many times a given categorical value is present in relation with a feature.

Numerical value	Animal
1.5	cat
3.6	cat
42	dog
7.1	crocodile

Frequency encoding →

Numerical value	Animal_freq
1.5	0.5
3.6	0.5
42	0.25
7.1	0.25

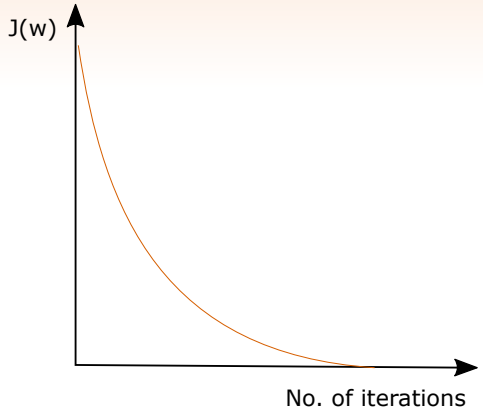
Gradient descent algorithm - practical considerations

- If the attribute/feature set is on the similar scale, then the gradient descent can converge quickly.
- For example, the square footage and the number of bedrooms in a feature list have drastically different scales.
- The contour curves may become skewed when several variables are on different scales –> gradient descent can be very slow.
- This can be avoided by using feature scaling.
- For example, we may scale the features to make them in the same scale, at least approximately.

$$x_1 = \frac{\text{size (sq. ft.)} - \min[\text{size (sq. ft.)}]}{\max[\text{size (sq. ft.)}] - \min[\text{size (sq. ft.)}]}$$

- We have already learned this technique as feature scaling through max-min normalization in machine learning

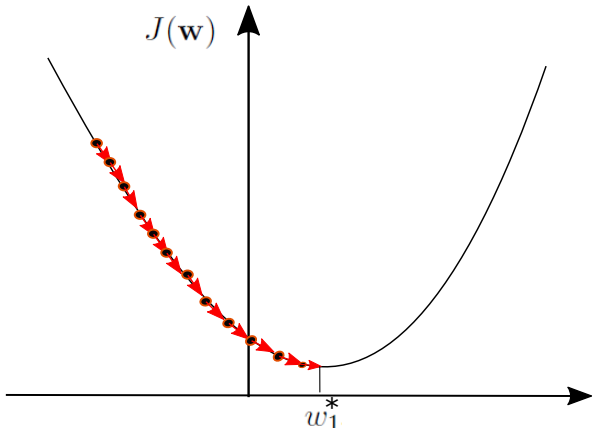
Gradient descent algorithm - Practical considerations



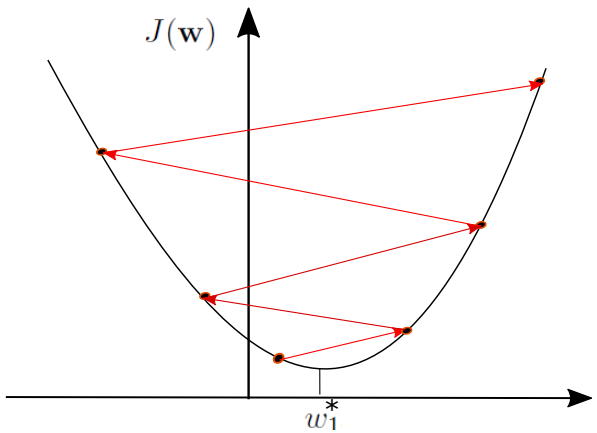
- To check the convergence of the algorithm, we can plot the cost function $J(\mathbf{w})$ (evaluated at the corresponding values of $\mathbf{w}$) against the number of iterations.
- If your gradient descent algorithm works properly, then in each iteration, the cost function must decrease.
- We may declare that the algorithm has converged when cost function value decreases less than a predefined cut-off threshold, e.g., 10^{-3} in one iteration.
- We need to use a smaller value for the learning rate (α) to avoid overshoot.
- If the learning rate α is too small, a large number of iterations (hence, a long-time) will be needed for convergence.

Gradient descent algorithm- Practical considerations

- For a sufficiently small learning rate α, the cost function should decrease in every iteration.
- But, if α is too small, then the gradient descent algorithm converges very slowly.



- If α is too large, then the cost function value may not decrease on every iteration. In this context, the algorithm may not converge.



Linear regression: Summary

- A linear regression model can be trained/computed through either gradient descent or Normal equations:

Gradient descent with multiple variables

Repeat until concergence{

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}) \quad \text{for} \quad j = 0, 1, 2, \cdots, n$$

}

- For gradient descent algorithm,
  - needs to choose the learning rate, α.
  - needs many iterations.
  - works well even when the size of the training date set is large.
- This gradient descent algorithm looks at every example in the entire training set on every step, and is called batch gradient descent.
- The mean-square cost function $J(\mathbf{w})$ is a convex quadratic function, and hence, the gradient descent algorithm always converge into the global minima.

Linear regression: Summary

- A linear regression model can be trained/computed through either gradient descent or Normal equations:

Normal equations

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

x_0	x_1	x_2	y
1	1.1	2.2	3.6
1	1.4	1.9	2.4
1	1.0	2.1	1.4

- For example,

$$\mathbf{X} = \begin{bmatrix} 1 & 1.1 & 2.2 \\ 1 & 1.4 & 1.9 \\ 1 & 1.0 & 1.4 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 3.6 \\ 2.4 \\ 1.4 \end{bmatrix}$$

- For gradient descent algorithm,
  - no need to choose a learning rate, α.
  - do not need many iterations.
  - becomes slow when the training data set is very large because it needs to compute the inverse $(\mathbf{X}^T\mathbf{X})^{-1}$, which scales as $O(N^2)$ for $N \times N$ matrix $\mathbf{X}$.

Gradient descent algorithm for large data sets

- This gradient descent algorithm looks at every example in the entire training set on every step, and is called batch gradient descent.

Batch gradient descent

Repeat until concergence{

$$\text{temp}_j := w_j - \alpha \sum_{i=1}^{m} \left(\sum_{l=1}^{n} w_l x_l^{(i)} - y^{(i)} \right) x_j^{(i)}$$

continue for all $j = 0, 1, \cdots, n$

$w_j := \text{temp}_j \quad \forall j$

}

- Thus, the batch gradient descent algorithm has to scan through the entire training set before taking a single step.
- This is a costly operation if the number of examples in the training data set (m) is very large.

Gradient descent algorithm for large data sets

- Useful terminology for Gradient descent algorithm
  - The batch size is a hyperparameter that defines the number of samples (rows in the data-set) to work through before updating the internal model parameters.
  - In machine learning, hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning.
  - An epoch in machine learning means one complete pass of the training data-set through the algorithm.
  - Thus, the epoch's number is an important hyperparameter for the learning algorithm.
  - It specifies the number of epochs or complete passes of the entire training dataset passing through the training or learning process of the algorithm.

Gradient descent algorithm for large data sets

- The convergence rate of batch gradient descent algorithm:
  - When the cost function is convex and its slope does not change abruptly (e.g., mean square error as the cost function), batch gradient descent with a fixed learning rate will eventually converge to the optimal solution.
  - But it can take $O(1/\epsilon)$ iterations to reach the optimum within a range of ϵ, depending on the shape of the cost function. Here, ϵ is a tolerance which can be used to bound the norm of the gradient vector.
  - The gradient descent has (almost) reached the minimum, when the norm of gradient vector becomes smaller than this tolerance (ϵ).
  - Meaning of $O(1/\epsilon)$: For example, if we divide the tolerance (ϵ) by 10 to have a more precise solution, then the algorithm may have to run about 10 times longer.
- The main problem of batch gradient descent algorithm is the fact that it uses the whole training set to compute the gradients at every step, which makes it very slow when the training set is very large.