Machine Learning and Data Mining I: Lecture 3
Polynomial Regression and Regularization

Morgan McCarty

06 July 2023

1 Polynomial Regression

1.1 Linear Regression Review

- Estimate the parameters of a regression function:
- $\hat{y}_i = w_0 + \Sigma_{j=1}^{N} w_j x_{ij}$ where:
  - $w = (X^T X)^{-1} X^T y$ where:
    - w is the vector of weights
    - X is the matrix of features
    - y is the vector of observations
  - y_i is the ith observation (of the vector of observations)
  - x_{ij} is the jth feature of the ith observation
  - N is the number of features
- We find the solution by minimizing the squared error on the training data.
- We want the model to $\underline{\text{generalize}}$ to new data (i.e. have low error on unseen [test] data).
- The parametric form of the regression function:
  - Linear in parameters
  - Can transform input features
  - Do we learn a straight line? Polynomial?

1.2 Polynomial Regression

- Transform the input features to higher order polynomials.
- i.e. $f_w(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_m x^m$
- Here m is the degree of the polynomial (the highest power of x).
- We can choose m, it is a hyperparameter.
- Does our optimization problem change? No, we can still use the same linear regression model, but we have to transform the input features.

- Our previous design matrix X was:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix} \tag{1}$$

- Where N is the number of observations and p is the number of features.
- All terms are linear in the parameters (i.e. $w_0, w_1, \ldots, w_N$ i.e. of order 1).
- We can transform the input features to higher order polynomials:

$$X = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \cdots & x_{11}^m \\ 1 & x_{21} & x_{21}^2 & \cdots & x_{21}^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N1}^2 & \cdots & x_{N1}^m \end{bmatrix} \tag{2}$$

- Where m is the degree of the polynomial and N is the number of observations.
- The weight vector w was previously:

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \tag{3}$$

- Now it is:

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \tag{4}$$

- The difference is that we have more features, but they are all linear in the parameters. That is that the parameters are still of order 1 so we can still use linear regression.
- The observation vector y remains the same:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{5}$$

- We can still use the same linear regression model:
- We will minimize the mean squared error:

$$\underset{w}{\operatorname{argmin}} \frac{1}{N} \Sigma_{i=1}^{N} (y_i - f_w(x_i))^2 \tag{6}$$

1.2.1 An Example

We have a training dataset sampled from the unit interval $[0, 1]$ with a target function $f(x) = 7.5 \sin(2.5\pi x)$ and gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

$$y_i = f(x_i) + \epsilon_i \tag{7}$$

The training dataset has N observations. (x_i, y_i) for $i = 1, \ldots, N$.

1. If we use Polynomial Regression with $m = 1$ (i.e. a straight line) the model fails to capture the underlying function.
   It has a massive training error and underfits the data. Meaning that the model is too simple to capture the underlying function.

2. If we use Polynomial Regression with $m = 6$ (i.e. a polynomial of degree 6) the model captures underlying function and the noise in the data.
   It has a negligible training error and overfits the data. Meaning that the model is too complex and fits to the noise, not just the function.

3. If we use Polynomial Regression with $m = 3$ (i.e. a polynomial of degree 3) the model captures underlying function without fitting to the noise in the data.
   It has a small training error and generalizes well to unseen data.

1.3 Overfitting

- The model learns idiosyncrasies (noise) in the training data resulting in poor generalization.
- When does this happen?
  - There are more model parameters than training data points.
  - Model is too complex and fits to the noise in the training data.
  - E.g. a degree N polynomial exactly fits $N + 1$ data points.

1.4 Avoiding Overfitting

- More training data: always works, but is not always possible in practice.
- Cross Validation: split the training data into a training set and a test set. Train the model on the training set and evaluate it on the test set.
- Regularization: Mathematical framework for controlling the complexity of the model.

2 Regularization

2.1 Ex. Polynomial Regression

More complex models (i.e. higher order or with larger m) lead to larger magnitude of model parameters. Slight perturbations in input features lead to large changes in the output. We can avoid overfitting by discouraging large model parameter values.

2.2 Occam's Razor

"Among competing hypotheses, the one with the fewest assumptions should be selected".
In order to avoid overfitting, we should choose the simplest model that fits the data.

2.3 Regularized Linear Regression

- Our original loss function:

$$E(w) = \frac{1}{2}\Sigma_{i=1}^{N}(y_i - w^T x_i)^2 \tag{8}$$

- Where x_i is the ith observation and y_i is the ith target value.
- We can penalize the weights to augment the error function (i.e. add a penalty/regularization term):

$$E^{ridge}(w) = \frac{1}{2}\Sigma_{i=1}^{N}(y_i - w^T x_i)^2 + \frac{\lambda}{2}||w||^2, \lambda \geq 0 \tag{9}$$

- $||w||^2$ is the squared Euclidean norm (L2 norm) of the weight vector.
- E.g. for weight vector w:

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \tag{10}$$

$$||w||^2 = w_0^2 + w_1^2 + \cdots + w_m^2 \tag{11}$$

- λ is the regularization parameter. It controls the strength of the regularization.
- $\lambda = 0$ is equivalent to the original loss function.
- $\lambda \rightarrow \infty$ is equivalent to setting all weights to 0.
- We can also use the L1 norm:

$$||w||_1 = |w_0| + |w_1| + \cdots + |w_m| \tag{12}$$

- This is called Lasso Regression. Other norms are possible.
- This would require swapping the L2 norm for the L1 norm in the loss function.
- The current model of Regularized Linear Regression is also called Ridge Regression.
- It is important to note that the regularization term is only applied to the weights, not the bias term. This is because the bias term is not a function of the input features.
- As such the weight vector has $m + 1$ elements, but the regularization term only has m elements.
- The training error will increase with the regularization parameter λ. This is because the model is penalized for having large weights.

2.4 Gradient Descent with L2 Regularization

- Once again we will use gradient descent to minimize the loss function.
- We will update the weights in two steps.
- First we update the intercept/bias term:

$$w_0^{i+1} = w_0^i - \eta\Sigma_{j=1}^N (w_0^i + w^{iT} x_j - y_j) \tag{13}$$

- Notice how the regularization term is not included in the update rule for the bias term.
- Next we update the weights:

$$w^{i+1} = w^i - \eta[\Sigma_{j=1}^N (w_0^i + w^{iT} x_j - y_j)(x_j) + \lambda w^i] w^{i+1} = (1-\eta\lambda)w^i - \eta\Sigma_{j=1}^N (w_0^i + w^{iT} x_j - y_j)(x_j), \text{ where } \lambda \geq 0 \tag{14}$$

- The regularization term is included for the rest of the weights.
- The second equation corresponds to "weight decay". i.e. each weight is decayed by a factor of $(1-\eta\lambda)$.
- "Weight decay" is a common term for regularization in neural networks. It is used because it is equivalent to L2 regularization. Both equations are not necessary to use together (i.e. you can use one or the other).
- As mentioned before, there are other norms that can be used for regularization.
- From a technical viewpoint: λ is a rank conditioner for the data covarience matrix, which helps make sure that the matrix is invertible.

2.5 Hyperparamter Learning

2.6 Connection to What We've Learned

- So far we have seen two hyperparameters:
  1. Learning Rate: η
  2. Regularization Parameter: λ
- There are several ways to choose these hyperparameters:
  1. Grid Search
  2. Random Search
  3. Genetic Algorithms
  4. Cross Validation
- These methods each have their own advantages and disadvantages.

2.7 Methods

1. Grid Search
   - Define a search space: i.e. a range of values for all hyperparameters of the model.
   - For Ridge Regression we have two hyperparameters: η and λ.
   - We can set a range of values in $[0, 1]$ for our parameters and evenly divide the range into 100 grid points.
   - At each value in the grid we train the model and estimate the performance metric (e.g. MSE, MAE, etc.).
   - Once we have evaluated the model at all grid points we choose the model with the best performance.
   - With many hyperparameters and a large range of values, this method can be very computationally expensive and time consuming. It may not be feasible to evaluate all grid points.
2. Random Search
   - Define a search space: i.e. a range of values for all hyperparameters of the model.
   - This functions similar to Grid Search, but instead of evaluating all grid points we randomly sample from the search space.
   - This is less computationally expensive than Grid Search, but it is unlikely to find the optimal hyperparameters.
   - It is possible to combine Grid Search and Random Search.
   - We can use Random Search to find a region of the search space that contains the optimal hyperparameters.
   - Then we can use Grid Search to find the optimal hyperparameters within that region.
3. Genetic Algorithms
   (a) Define a search space: i.e. a range of values for all hyperparameters of the model.
   (b) This is similar to Random Search, but instead of randomly sampling from the search space we use a genetic algorithm to find the optimal hyperparameters.
   (c) This is less computationally expensive than Grid Search, but if the search space is large it will only approach the optimal hyperparameters.
   (d) It is possible to combine Grid Search and Genetic Algorithms (similar to Random Search).
4. Cross Validation
   (a) This is the most common method for choosing hyperparameters.
   (b) We will take our training data and split it into k folds (a <u>fold</u> is a subset of the training data).
   (c) k is also a hyperparameter, but it is not the same as the hyperparameters of the model. Usually $k = 5$ or $k = 10$.
   (d) We will train the model on $k - 1$ folds and evaluate it on the remaining fold.
   (e) We will repeat this process k times, each time using a different fold as the test set.
   (f) We will then average the performance metric (e.g. MSE, MAE, etc.) over the k folds.
   (g) This is done at each point in the search space, so it is computationally expensive, but it is the most accurate method for choosing hyperparameters.

(h) It is important to ensure that there is no "leakage" between the training and test sets as this will result in an overly optimistic performance metric (i.e. the model will overfit the data).

(i) Over all the folds we will have k performance metrics and then we will average them.

2.8 Final Notes

- It is useful to standardize and normalize the input features before training the model.
- We can standardize the input features by subtracting the mean and dividing by the standard deviation.
- We can normalize the input features by subtracting the minimum and dividing by the range.