Tufts

Class #04: Regression with Regularization

Machine Learning (COMP 135)

1

Review: Squared Error in Linear Regression

- For a chosen set of weights, $\mathbf{w}$, we can define an error function as the ***squared residual*** between what the hypothesis function predicts and the actual output, summed over all N test-cases:

$$Loss(\mathbf{w}) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2$$

- Learning is then the process of finding a weight-sequence that ***minimizes*** this loss:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} Loss(\mathbf{w})$$

- ***Note***: Other loss-functions are commonly used (but the basic learning problem remains the same)

2

Review: Finding Weights Analytically

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} Loss(\mathbf{w})$$

- We can ***in principle*** solve for the weight with least error ***analytically***

1. Create data matrix with one training input example per row, one feature per column, and output vector of all training outputs

$$\mathbf{X} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N1} & f_{N2} & \cdots & f_{Nn} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

2. ***Solve*** for the minimal weights using linear algebra (for large data, requires optimized routines for finding matrix inverses, doing multiplications, etc., as well as for certain matrix properties to hold, which are not universal):

$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

3

When is the Analytic Solution Possible?

$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- Linear regression using squared error is ***special***: in many cases, we can generate an exact closed-form solution to the minimal-weights problem
  - Other ML techniques will use ***different*** error functions
  - For many of these functions, we ***do not*** have a closed form solution (or it is at least very hard to solve)
  - Gradient descent is thus a general-purpose method for finding the required weights iteratively

4

When is the Analytic Solution Possible?

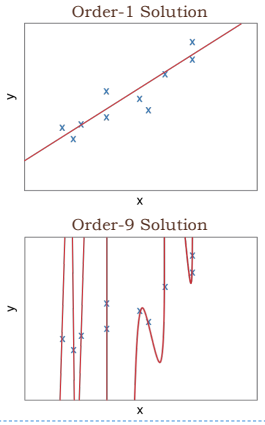
$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

- Even for linear regression with squared error, it is ***not*** always possible to use the linear algebra method directly
  - Depends upon the relationship between number of data-points (N) and number of features (F)
- N > F: More examples than features (over-determined)
  - In many cases, ***can*** be solved (***unless*** X^TX is rank-deficient)
  - Resulting solution likely still has error on training set
- N = F: Same number of examples and features
  - In many cases, ***can*** be solved (***unless*** X^TX is rank-deficient)
  - Resulting solution optimal, with 0 error on training set
- N < F: More features than examples (or rank-deficient matrix)
  - Infinitely many 0-error solutions exist
  - X^TX will ***not*** be invertible, and analytic methods will generally ***fail***

5

Review: Polynomial Regression & Overfitting

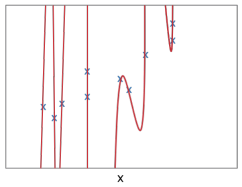
- When training a model, we sometimes need to make it more complex to fit data
  - Here, we can see that a simple linear regression doesn't fit data very well
  - At the same time, the higher-order polynomial solution hits all our training data perfectly, but appears very overfit



6

Overfitting, Weights, and Possible Solutions

- Overfitting often comes from complex models with high magnitude weights
  - Large positive/negative coefficients
  - Some features treated as far more important than others (and perhaps more important than they ***should*** be)



- One solution: work on making the models ***simpler***
- Another: ***keep*** the model, but "discourage" large weights
- Instead of working to minimize model error alone, we seek to minimize a ***combination*** of error and complexity

$$Loss(\mathbf{w}) = Error(\mathbf{w}) + Complexity(\mathbf{w})$$

7

Regularizing Model Weights

$$Loss(\mathbf{w}) = Error(\mathbf{w}) + Complexity(\mathbf{w})$$

- We often want to be able to ***control*** the amount of emphasis we place upon model complexity
- We can thus add a weighting parameter to the model complexity measure itself:

$$Loss(\mathbf{w}) = Error(\mathbf{w}) + \lambda\, Complexity(\mathbf{w})$$

- If complexity is proportional to the magnitudes of the weights, this is known as regularization
  - Minimizing loss will involve reducing these magnitudes towards 0
  - The control λ controls how much we stress this: the higher λ is, the more we emphasize reducing weights over reducing error
  - At high enough values of λ, we can sacrifice accuracy for simplicity

8

Measuring Complexity of Weights

- Regularization can use the concept of a vector norm
- Given our weight vector:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

NB: omits bias term w_0, as in general we seek to minimize ***feature***-weights

- The p-norm (or Lp norm), for p = 1, 2, …, is:

$$|\mathbf{w}|_p = \left(\sum_i |w_i|^p \right)^{1/p}$$

$$= \sqrt[p]{\sum_i |w_i|^p}$$

9

Lp Norms of Weights

- The basic L1 norm of the weights is simply a sum of their magnitudes (absolute values):

$$|\mathbf{w}|_1 = |w_1| + |w_2| + \cdots + |w_n|$$

- The L2 norm is probably most common in many domains, and is sometimes just called "the norm" or "magnitude":

$$|\mathbf{w}|_2 = (|w_1|^2 + |w_2|^2 + \cdots + |w_n|^2)^{\frac{1}{2}}$$

- The definition can be extended to any positive value p:

$$|\mathbf{w}|_p = (|w_1|^p + |w_2|^p + \cdots + |w_n|^p)^{\frac{1}{p}}$$

10

Regression with Regularization

- To use Lp norms directly in linear regression, we can add them to the loss, and have our model minimize the combination of model error and the weight norm:

$$Loss(\mathbf{w}) = Error(\mathbf{w}) + \lambda\, Complexity(\mathbf{w})$$

$$= \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda\, |\mathbf{w}|_p$$

- In practice, we sometimes use modifications of the norm
- Depending upon the version of the norm we use, we can get different results

11

Basic Effects of Regularization

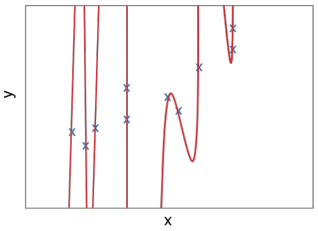
$$Loss(\mathbf{w}) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda\, |\mathbf{w}|_p$$

- We can use the same analytical (closed-form) or iterative (gradient descent) techniques to find the weights that minimize our combined loss function
  - We can experiment with ***both*** the value of p we use for the norm, and the weight factor λ that is applied to it
  - Setting (λ = 0) means we are just doing linear regression without any regularization
  - Setting (λ > 0) can cause final weights to be closer to 0
  - As usual, we would explore both, using validation methods, to try to find most robust settings, avoiding likely overfitting

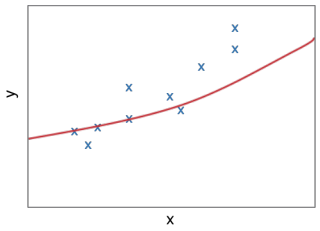
12

Levels of Regularization

(λ = 0): No regularization



(λ >> 0): Heavy regularization

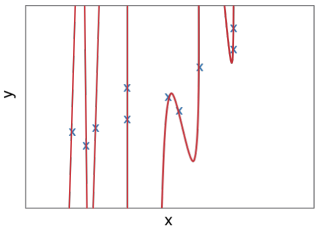


- For a given value of p, we can explore higher or lower values of control parameter λ
  - Very low values ***may*** generate overfitting
  - Very high values ***may*** generate underfitting

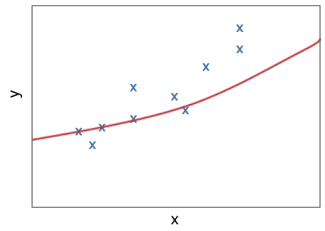
13

Regularization and Bias/Variance

(λ = 0): No regularization



(λ >> 0): Heavy regularization

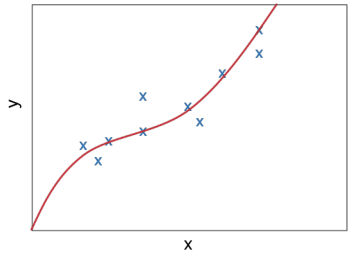


- An overfit model often has low bias but high variance
  - Training data is captured precisely, but so is noise, and result fits other data poorly
- An underfit model often has high bias but low variance
  - Simple, less-noisy models that do not capture our data very well

14

Basic Effects of Regularization

(λ = "just right"): Moderate regularization



- An ideal solution seeks values of λ that lead to overall best performance across training/testing data, ***balancing*** bias and variance trade-offs
  - Allows for more complex models, responding to subtle combinations of features
  - Helps avoid models that perform highly in training but are awful on new data

15

L1 & L2: Common Types of Regularization

- Most common regularized models use versions of the L1 and L2 norms
- L1 regression (aka the Lasso): seeks to minimize the summed magnitudes of the weights in the loss:

$$Loss_1(\mathbf{w}) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda |\mathbf{w}|_1$$

$$= \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda \sum_i |w_i|$$

- L2 regression (aka ridge regression): seeks to minimize the summed ***squared*** magnitudes of the weights:

$$Loss_2(\mathbf{w}) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda (|\mathbf{w}|_2)^2$$

$$= \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda \sum_i w_i^2$$

NB: uses square of the L2 norm (omitting square root operation)

16

L2: Ridge Regression

$$Loss_2(\mathbf{w}) = \sum_{j=1}^{N}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda \sum_i w_i^2$$

- L2 regression was proposed by Hoerl & Kennard (1970)
  - Has the effect of adding constant parameter λ to the diagonal entries in the matrix $X^T X$ (hence the name "ridge")
- By adding in a penalty in terms of the summed squares of weights, minimizing loss discourages large magnitudes
  - Weights are driven ***towards*** zero
  - In general, most will tend to be smaller non-zero values (for reasonable settings of λ > 0)

17

L1 Regression: The Lasso

$$Loss_1(\mathbf{w}) = \sum_{j=1}^{N}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2 + \lambda \sum_i |w_i|$$

- L1 regression first examined deeply by Tibshirani (1996)
  - The "**l**east **a**bsolute **s**hrinkage and **s**election **o**perator" or *lasso*
- Using summed absolute weights also drives them smaller
  - Weights are driven ***towards*** zero
  - Unlike how ridge tends to perform, large enough values of λ often means that many weights ***are*** zero exactly
  - Lasso thus serves as a way of doing feature selection: since zero-weight features play no part in the hypothesis, they are effectively pruned out and ignored

18

L1 vs. L2: Which to Choose?

- Neither form of regression/regularization is always best
  - Sometimes, if we know something about our data, we may have reason to prefer one over the other
  - Other times, we may just want to explore both
  - There are other techniques, too, like elastic net regularization, which combines the L1 and L2 ideas
- If you know that the data has a lot of features, only some of which are important, but you're not sure which: you might prefer L1, since it can "prune out" unimportant ones by setting their weights to zero
- If you know that the patterns in your data are affected by subtle combinations of many small features: you might prefer L2, since it usually won't prune out any

19