UNC CHARLOTTE

The WILLIAM STATES LEE COLLEGE *of* ENGINEERING

Introduction to ML
Lecture 3: Regularization

Hamed Tabkhi
Department of Electrical and Computer Engineering,
University of North Carolina Charlotte (UNCC)
htabkhiv@uncc.edu

UNC CHARLOTTE

Improving learning via data pre-processing

- Feature scaling (or normalization) and standardization
- Training and Validation
- Training regularization

Improving learning via data pre-processing

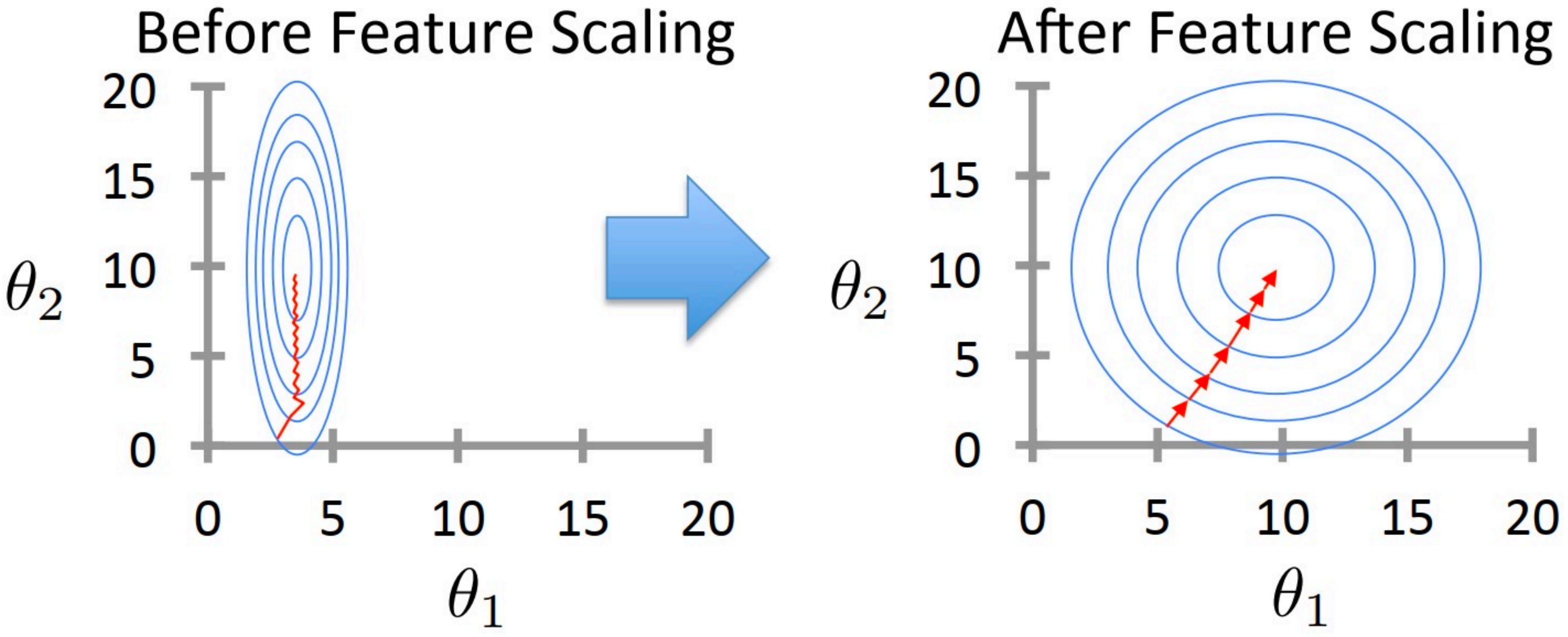
- Gradient descent based algorithms

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

  - The presence of feature value *x* in the formula will affect the step size of the gradient descent.
  - The difference in ranges of features (inputs) will cause different step sizes for each feature.
  - Data (feature) scaling
    - ensure the gradient descent moves smoothly towards the minima
    - the steps for gradient descent are updated at the same rate for all the features

Improving learning – feature scaling

- Idea: Ensure that features have similar scales



- Makes gradient descent converge much faster

The following slides are adapted from slides by Joseph Bradley and Andrew Ng

UNC CHARLOTTE

MIN MAX Normalization

- Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.
- Here’s the formula for normalization:

$$\text{Normalization} \quad \overset{\text{new value}}{X'} = \frac{\overset{\text{original value}}{x} - \min(x)}{\max(x) - \min(x)}$$

- Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.
- When the value of X is the minimum value in the column, the numerator will be 0, and hence X’ is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X’ is 1
- If the value of X is between the minimum and the maximum value, then the value of X’ is between 0 and 1

Standardization

- Rescales features to have zero mean and unit variance
  - Let μ_j be the mean of feature j: $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}$
  - Replace each value with:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

for j = 1...d
(not x_0!)

n

    - s_j is the standard deviation of feature j
    - Could also use the range of feature j (max$_j$ – min$_j$) for s_j

- Must apply the same transformation to instances for both training and prediction

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

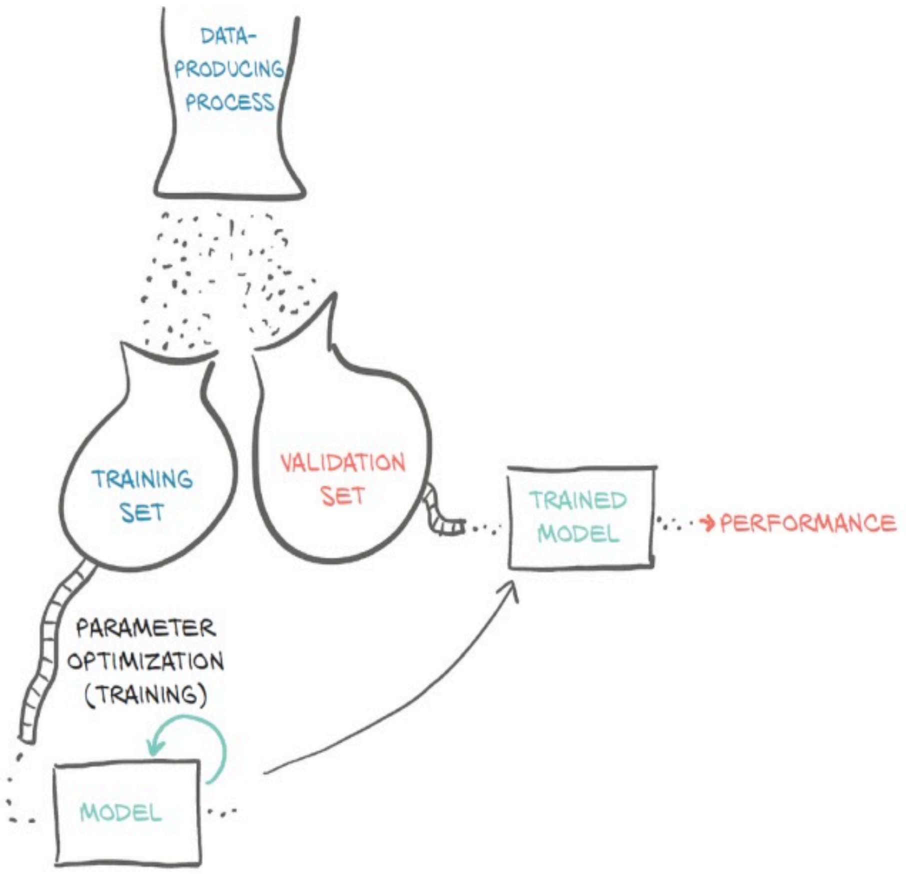
σ : Population standard deviation
x : Datapoint value
μ : Population mean
N : Population size

Improving learning via data pre-processing

- Training and Validation
- Feature scaling (or normalization) and standardization
- Training regularization

Evaluating the Training Loss

- During the training, we make sure that the loss is minimal *across training* data points, but we'll have no guarantee that the model behaves well *away from* or *in between* the data points.
- Evaluating the loss at those independent data points would yiel higher-than-expected loss.
- We must take a few data points out of our dataset (the *validation set*) and only fit our model on the remaining data points (the *training set*).
- Then, while we're fitting the model, we can evaluate the loss once on th training set and once on the validation set.

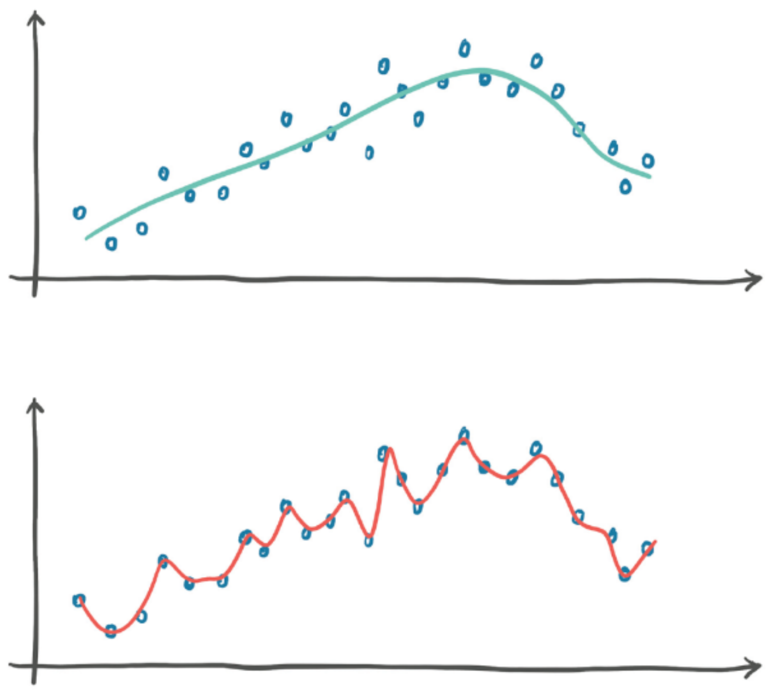


Training Rules

- Rule 1: if the training loss is not decreasing, chances are the model is too simple for the data. The other possibility is that our data just doesn’t contain meaningful information that lets it explain the output

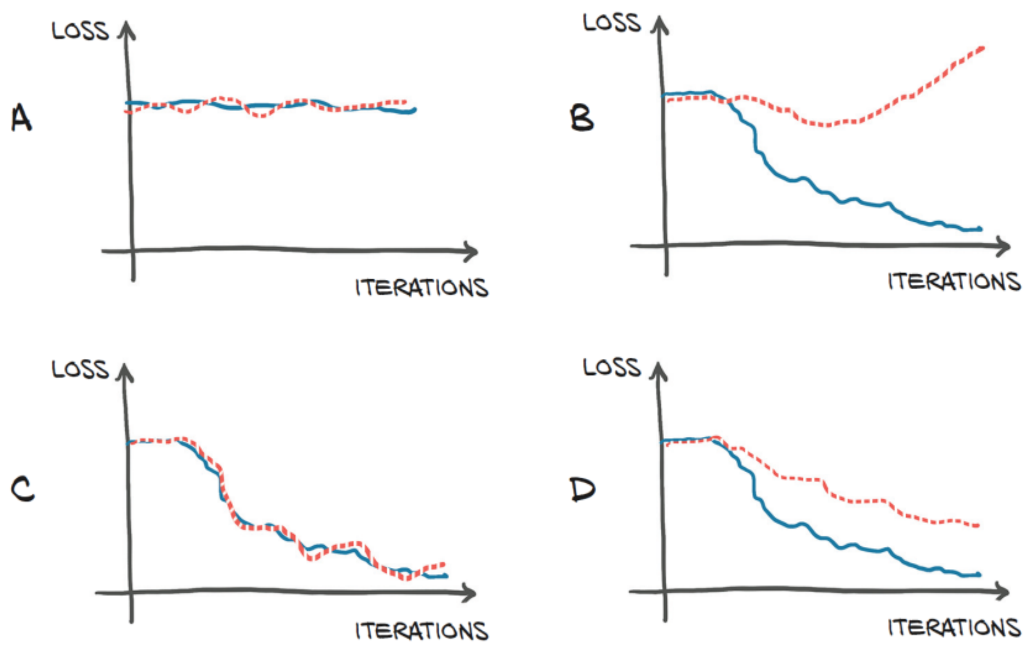
- Rule 2: if the training loss and the validation loss diverge, we’re overfitting.
    - If the loss evaluated in the validation set doesn’t decrease along with the training set, it means our model is improving its fit of the samples it is seeing during training, but it is not *generalizing* to samples outside this precise set.

Overfitting



- Over fitting: evaluating the loss at those independent data points would yield
- Both a piecewise polynomial or a really large neural could generate a model meandering its way through the data points.
  - There is nothing to keep the model in check for inputs away from the training data points.
- To avoid overfitting, While training, we need to keep the model in check for inputs away from the training data points.
- We've got some nice trade-offs here. On the one hand, we need the model to have enough capacity for it to fit the training set. On the other, we need the model to avoid overfitting.

Generalization



(A) Training and validation losses do not decrease; the model is not learning due to no information in the data or insufficient capacity of the model.

(B) Overfitting: training loss decreases while validation loss increases.

(C) Training and validation losses decrease exactly in tandem. Performance may be improved further as the model is not at the limit of overfitting.

(D) Overfitting is under control: training and validation losses have different absolute values but similar trend.

UNC CHARLOTTE

Improving learning via data pre-processing

- Training and Validation
- Feature scaling (or normalization) and standardization
- Training regularization

Regularization

- Training a model involves two critical steps:
- **Optimization:** when we need the loss to decrease on the training set
- **Generalization:** when the model has to work not only on the training set but also on data it has not seen before, like the validation set.

- The mathematical tools aimed at easing these two steps are sometimes subsumed under the label *regularization*.

Regularization: Parameters Penalties

- A major way to stabilize generalization is to add a regularization term to the loss.
- This term is crafted so that the parameters (weights) of the model tend to be small on their own, limiting how much training makes them grow.
- Overall, we want to keep the value of the parameters stay relatively small with respect to our training loss

- **In other words, it is a penalty on larger parameters values.**
- This makes the loss have a smoother topography, and there's relatively less to gain from fitting individual samples.

Regularization: Parameters Penalties

- Regularization
  - A method for automatically controlling the complexity of the learned hypothesis
  - **Idea**: penalize for large values of θ_j
    - Can incorporate into the cost function
    - Works well when we have a lot of features, each that contributes a bit to predicting the label
  - Can also address overfitting by eliminating features (either manually or via model selection)

Regularization: Parameters Penalties

- Linear regression objective function

Input variables (features)
$x_1, \ldots, x_n$

m samples

$$J(\theta) = \frac{1}{2m}\left[\underbrace{\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}_{\text{model fit to data}} + \lambda \underbrace{\sum_{j=1}^{n}\theta_j^2}_{\text{regularization}}\right]$$

Measures the magnitude of the parameter vector

$$\min_\theta J(\theta)$$

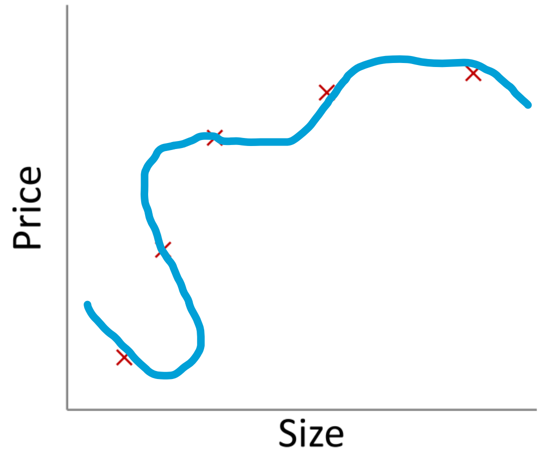
  - λ is the regularization parameter ($\lambda \geq 0$)
  - No regularization on θ_0!

Regularization: Parameters Penalties

Understanding Regularization

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \underline{\lambda \sum_{j=1}^{n} \theta_j^2}\right]$$

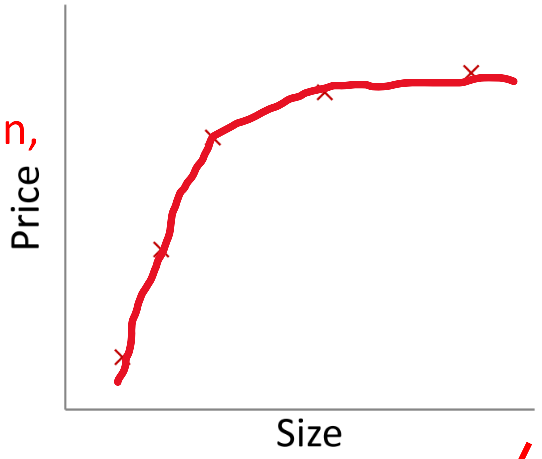
Without regularization



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

With regularization, what we get?

penalize θ_3 and θ_4



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cancel{\theta_3} x^3 + \cancel{\theta_4} x^4$$

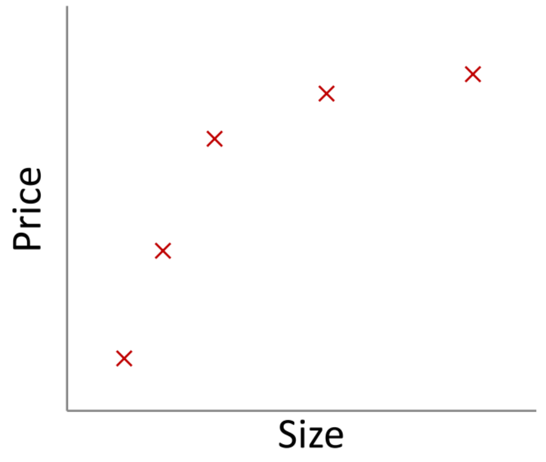
$\approx$

Regularization: Parameters Penalties

Understanding Regularization

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

- What happens if we set λ to be huge (e.g., 10^{10})?

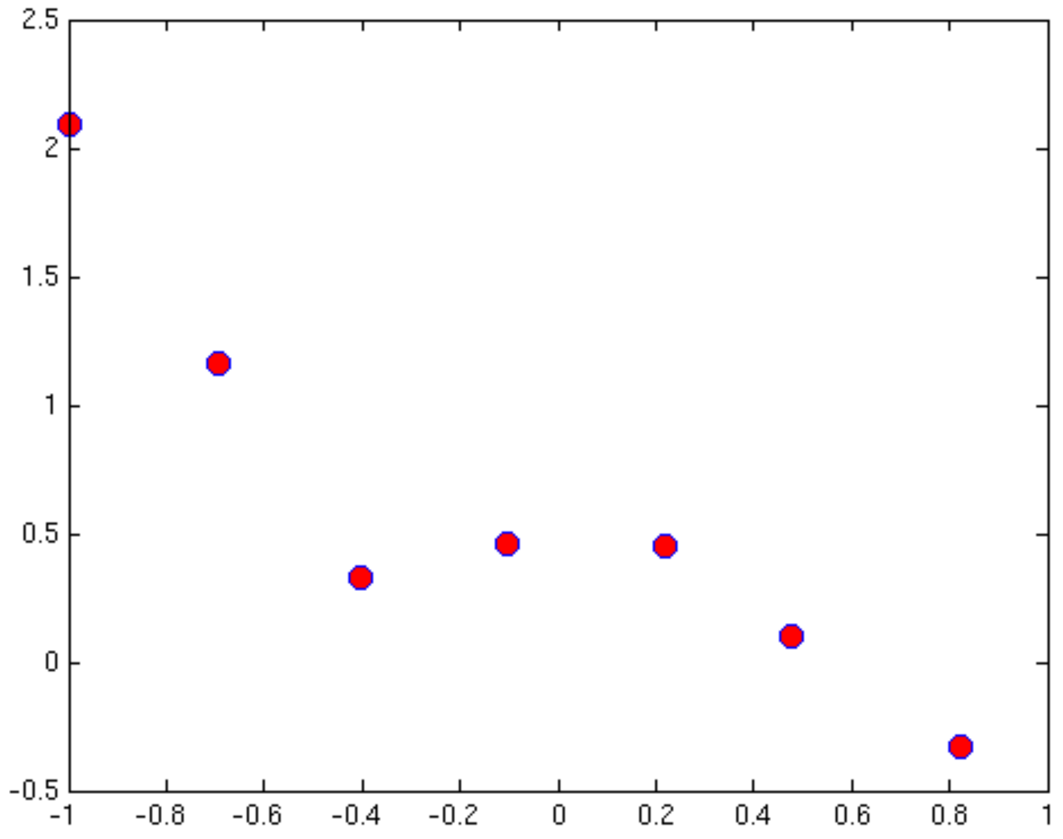


$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Regularization: Parameters Penalties

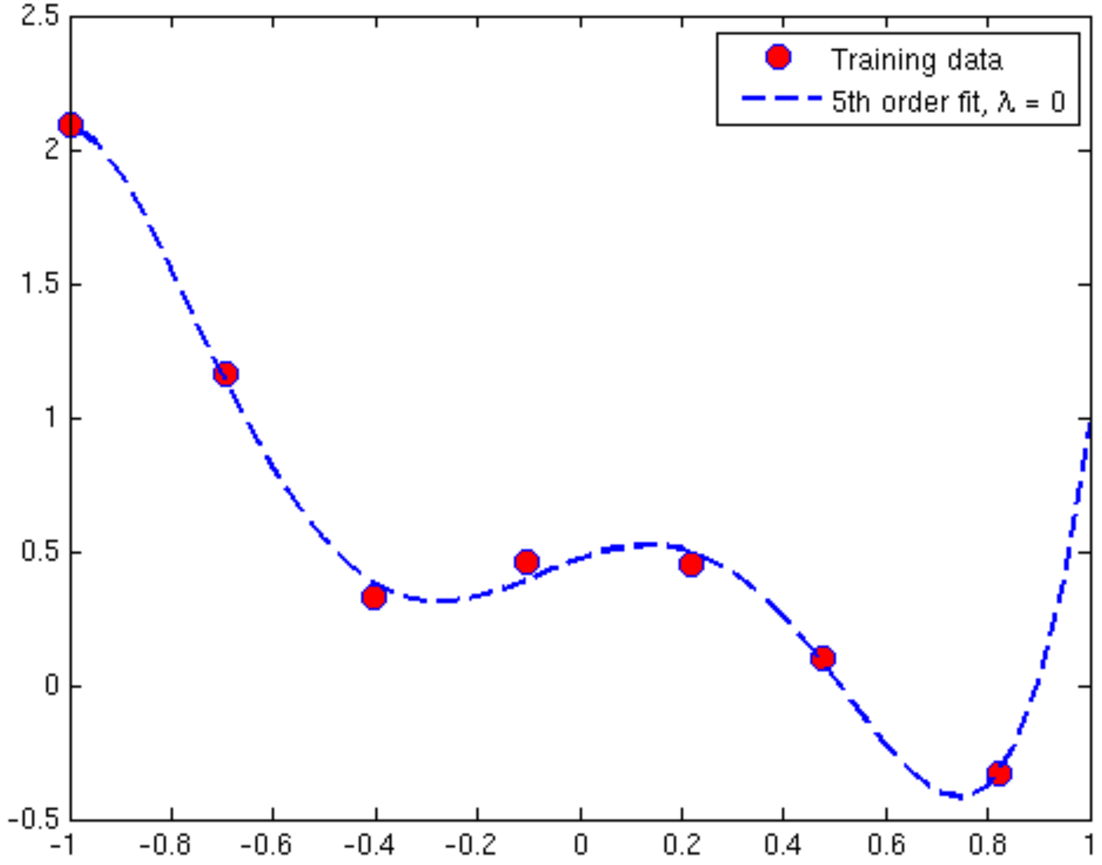
- Example

Input data scatter plot



C CHARLOTTE

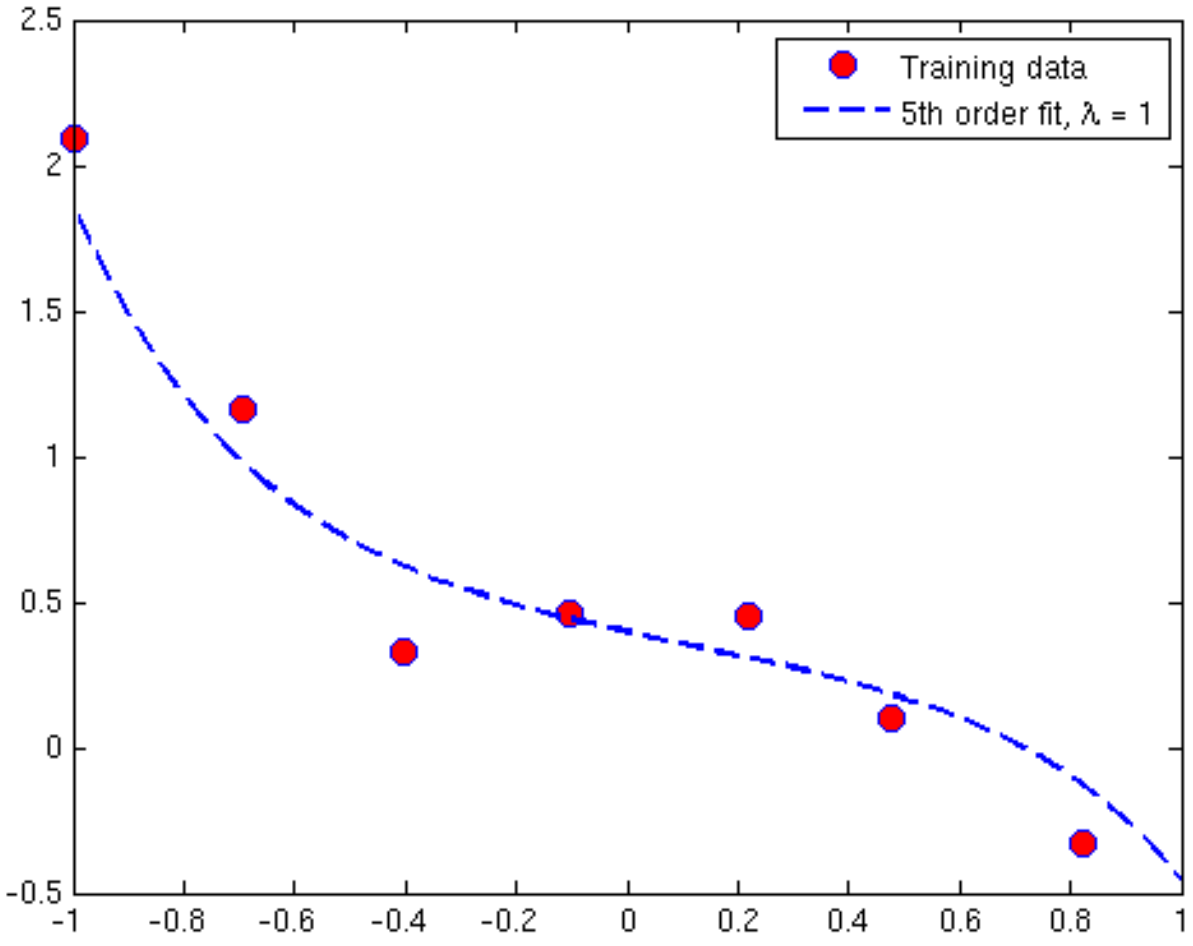
Regularization: Parameters Penalties



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

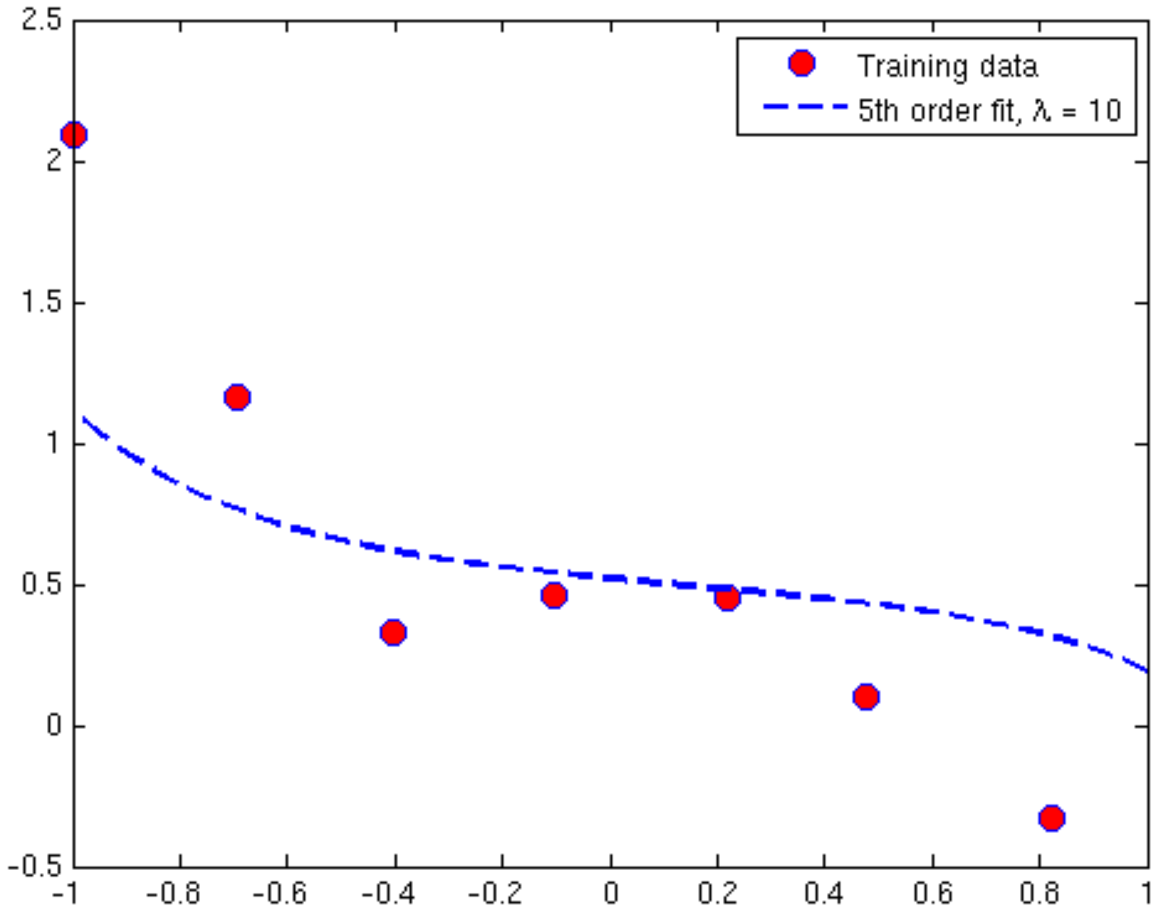
UNC CHARLOTTE

Regularization: Parameters Penalties



UNC CHARLOTTE

Regularization: Parameters Penalties



Regularization: Parameters Penalties

- Regularized Linear Regression

Gradient decent

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \underbrace{x_0^{(i)}}_{1}$$

Add for regularization

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m}\theta_j \right]$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n) \}$$

$$\Downarrow$$

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Useful resources

- Regularized regression (implementation)
  - https://dafriedman97.github.io/mlbook/content/c2/s2/regularized.html
- Machine learning in NumPy
  - https://numpy-ml.readthedocs.io/en/latest/numpy_ml.linear_models.html