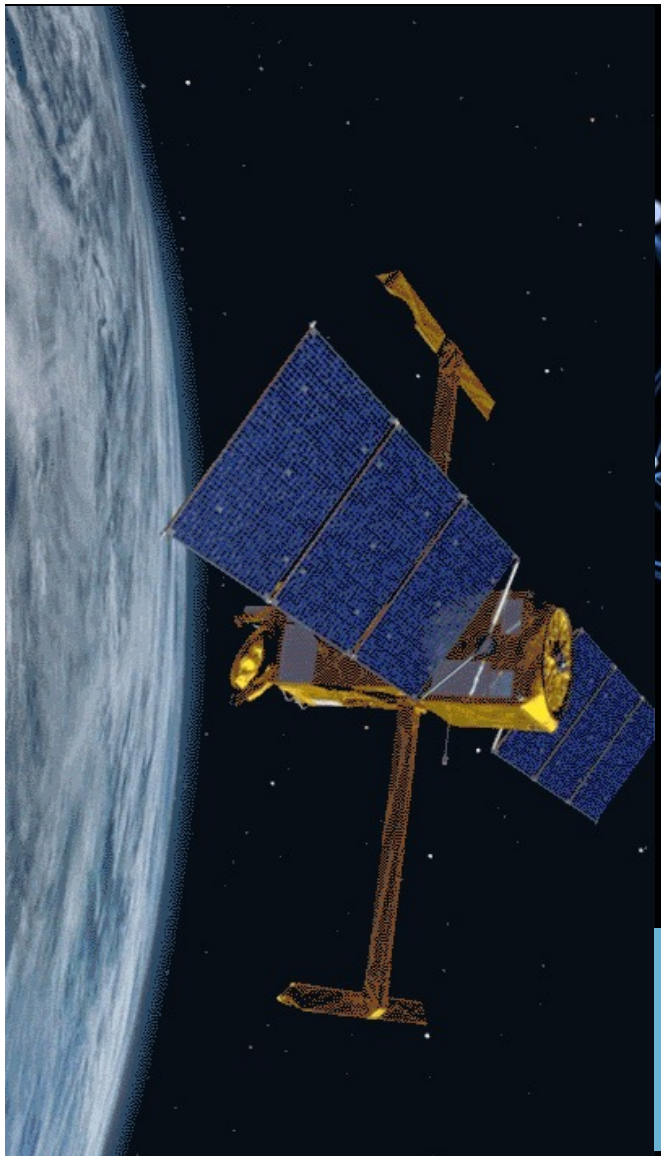
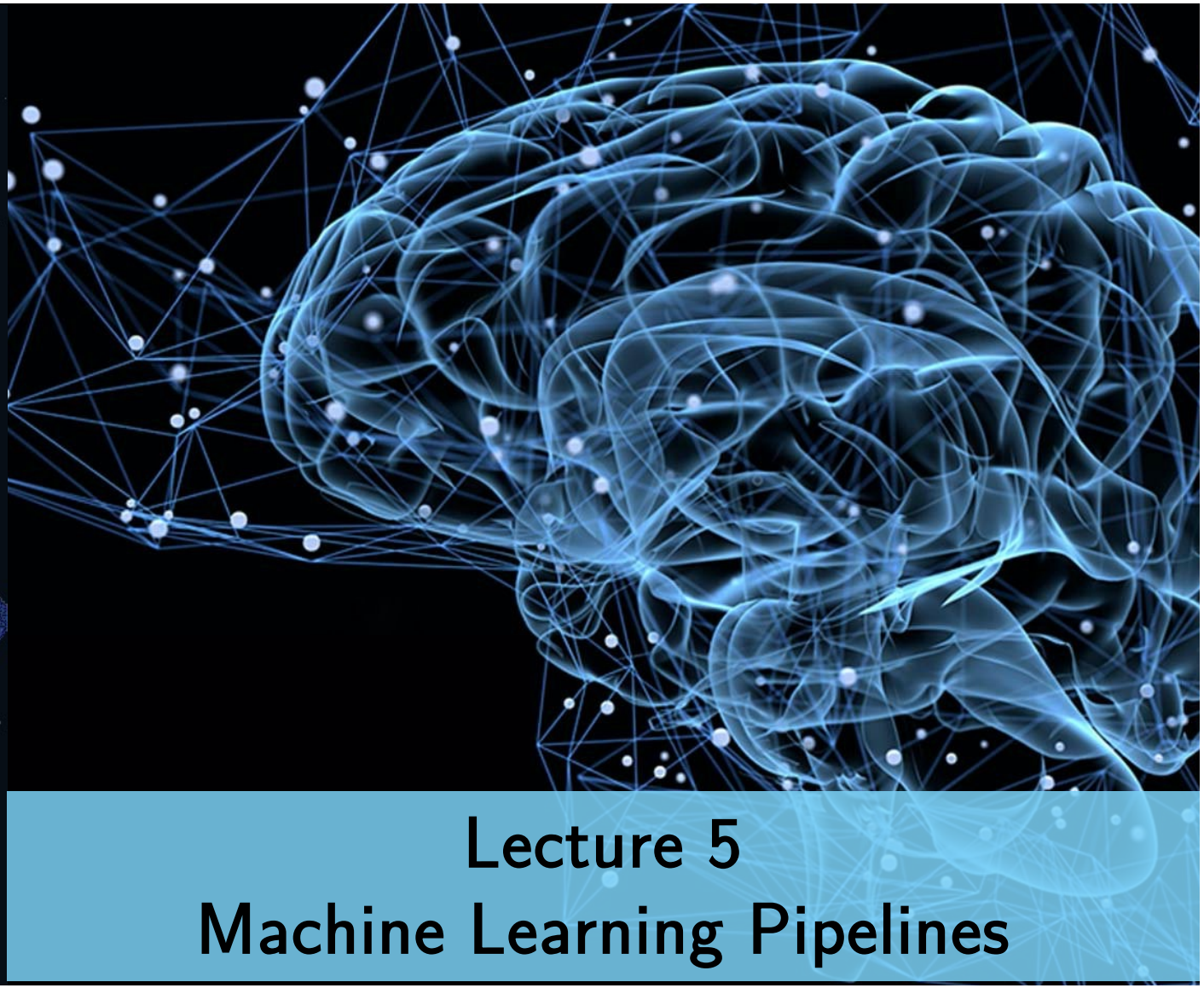




Lecture 5
Machine Learning Pipelines

Overview

- Data preprocessing **pipelines**
  - Categorical features
  - Missing feature values
  - Feature scaling
- How to handle underfitting/overfitting?
  - Lasso & Ridge Regularization
  - Feature selection

Categorical features

Example

Course Number	Credit hrs	Days	Times	Instructor	Average grade	SEI score	Topic		Course popularity
5194	2	Wed	9.20am – 10.15am	Joachim Moortgat	NaN	NaN	Programming	…	8.1
5751	4	Tue-Thur	9.35am - 11.35am	Joachim Moortgat	87%	4.9	Programming	…	7.1
5663	3	Tue-Thur	11.10am - 12.30pm	Franklin Schwartz	66%	5	Sustainability	…	6.3
5780	4	Mon-Wed-Fri	9.10am - 10.05am 2.40pm - 5.30pm	Derek Sawyer	90%	4.7	Seismology	…	6.8
5641	2	Tue-Thur	12.45pm - 2.35pm	Yanlan Liu	80%	5	Programming	…	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
5651	3	Tue-Wed-Thur	9.35 - 10.55am and 2.20pm - 4.10pm	Audrey Sawyer	85%	5	Hydrogeology	…	9

Categorical features

- Features like ‘instructor’ contain strings: called **categorical features**
- ML algorithms are purely algebra on arrays, need only numbers
- How do we turn categorical features into numbers?

One-Hot-Encoding

SEI/5	Hrs per week/5	Morning class	Moortgat	Schwartz	D Sawyer	Y. Liu	A Sawyer		Course popularity
0.98	0.4	1	1	0	0	0	0	…	?
0.98	0.8	1	1	0	0	0	0	…	7.1
1	0.6	0	0	1	0	0	0	…	6.3
0.94	0.8	0	0	0	1	0	0	…	6.8
1	0.4	1	0	0	0	1	0	…	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0.6	1	0	0	0	0	1	…	9

Impute missing values

Example

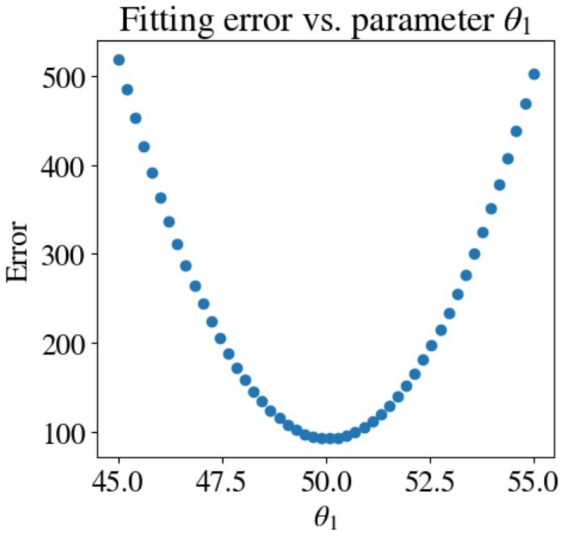
Course Number	Credit hrs	Days	Times	Instructor	Average grade	SEI score	Topic		Course popularity
5194	2	Wed	9.20am – 10.15am	Joachim Moortgat	NaN	NaN	Programming	…	8.1
5751	4	Tue-Thur	9.35am - 11.35am	Joachim Moortgat	87%	4.9	Programming	…	7.1

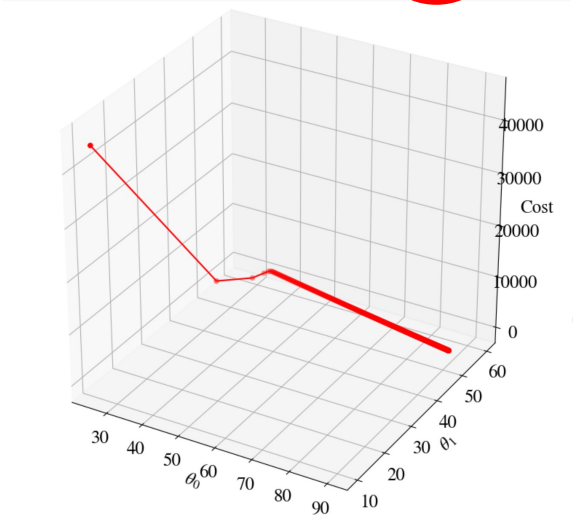
- Options: replace missing numerical feature values with, e.g., mean/median
- Replace categorical feature values with, e.g. most common, or separate class
- **from** sklearn.impute **import** SimpleImputer
- Or..: use ML to estimate missing values

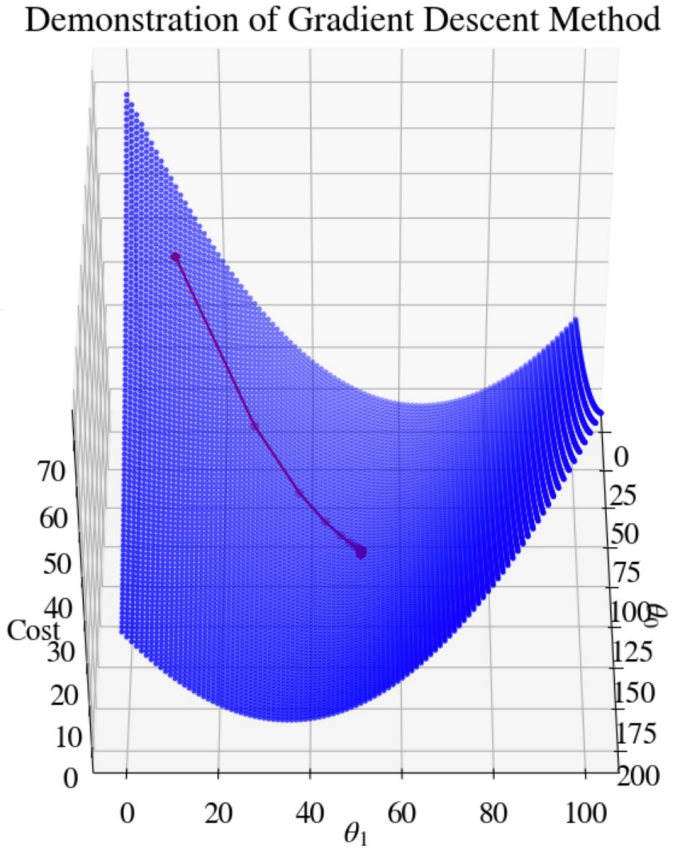
Feature Scaling

Machine Learning as Minimization Schemes

- Find minimum of mean-**squared** or logistic errors
  - $\theta_0^{\text{new}} = \theta_0 - \alpha\frac{\partial J}{\partial \theta_0} = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(\quad \hat{y}_i \quad - y_i).$
  - $\theta_1^{\text{new}} = \theta_1 - \alpha\frac{\partial J}{\partial \theta_1} = \theta_1 - \frac{\alpha}{m}\sum_{i=1}^{m}(\quad \hat{y}_i \quad - y_i)x_i.$







Challenge & Solution

- Gradient descent methods (and others) do not converge if error gradient in one feature is much larger/smaller than in other feature(s)

- Also affects *sensitivity* of fit to each feature

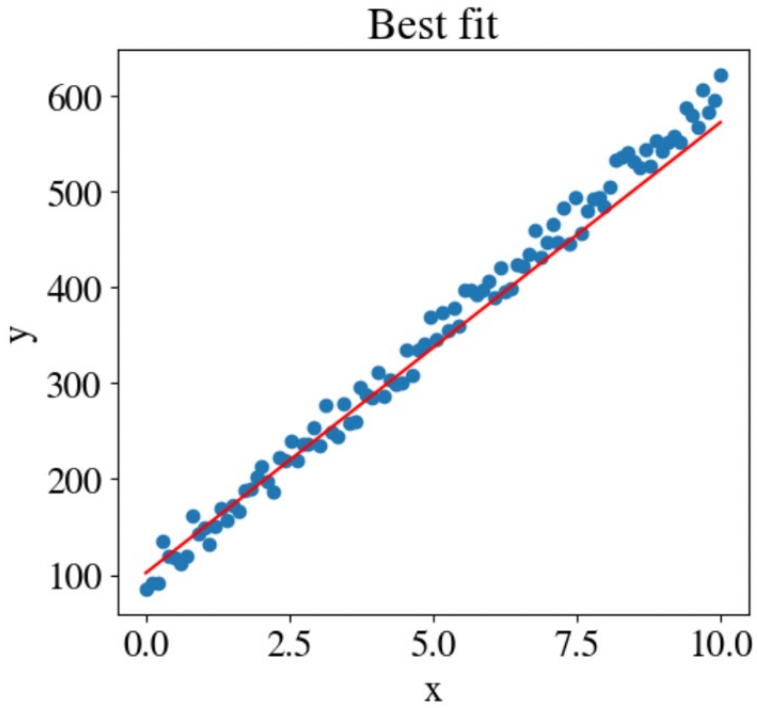
- Solution: `min-max' feature scaling: $\tilde{x}_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$.

- (e.g., `sklearn.preprocessing.`**`MinMaxScaler`**)

Example

- Original feature $0 < x_1 < 10$ (and forget about intercept)
- Best fit slope is, say, $\theta_1 = 47$, so:
- $y = \theta_1 x_1 = 47\ x_1$

- Define $\tilde{x}_1 = x_1 / x_{max}$ with $x_{max} = 10$
- So $0 < \tilde{x}_1 < 1$
- Best fit slope will be $\tilde{\theta}_1 = 470$
- $y = \tilde{\theta}_1 \tilde{x}_1 = 470\ \tilde{x}_1$
- Or with $\theta_1 = \tilde{\theta}_1 / x_{max}$, $y = \theta_1 x_1 = 47\ x_1$



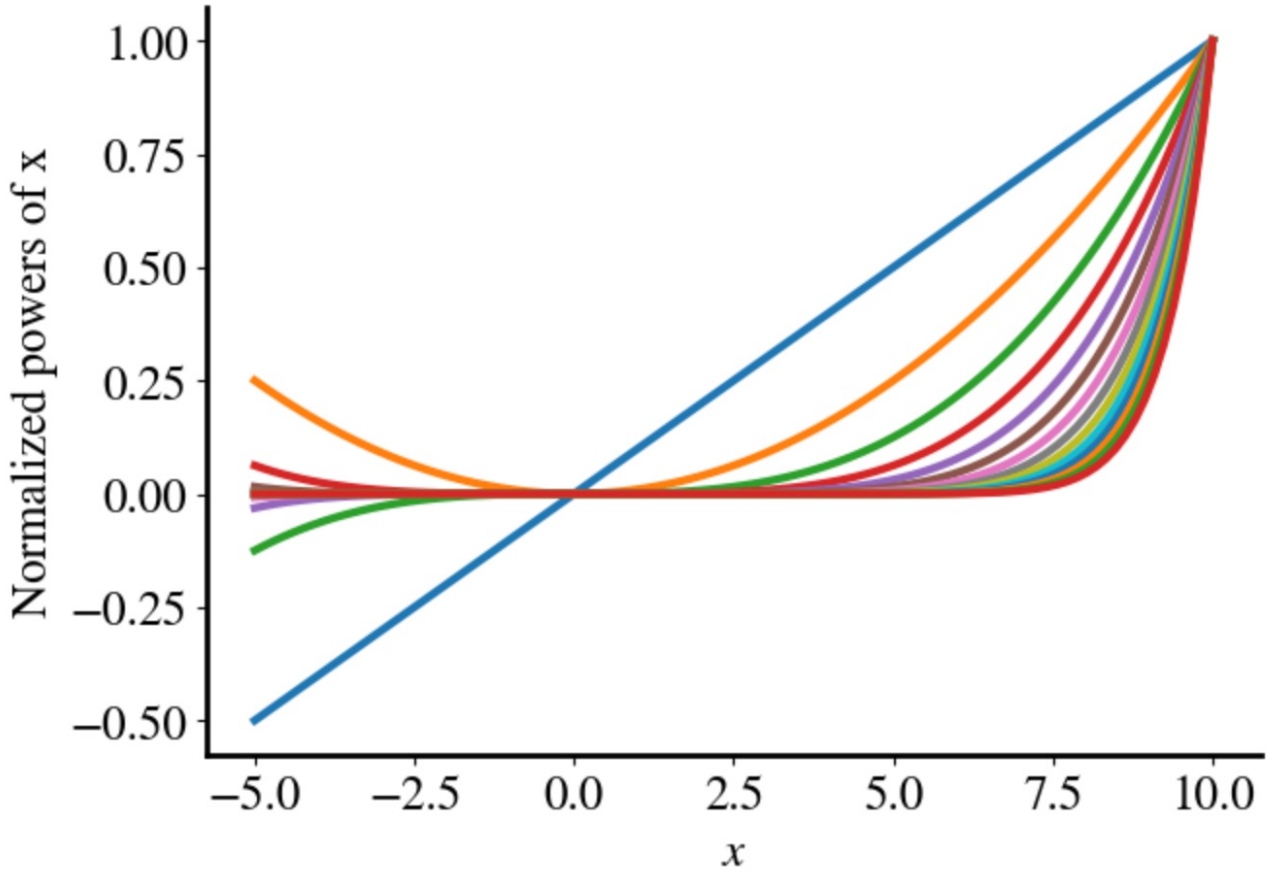
Generalization

- $i = 1, \ldots, m$ measurements, $j = 1, \ldots, n$ features (e.g. polynomials), $x_j^{(i)}$
- For each feature, find $x_{j,max}$ across all measurements **in *training* data**
- Normalize each feature $\tilde{x}_j^{(i)} = x_j^{(i)} / x_{j,max}$, then perform fitting
- Optional: divide fitting parameters $\widetilde{\theta_j}$ with $x_{j,max}$ to obtain model i.t.o. original / un-normalized features.

- Easy python code. Assume a feature array *X* (with or without bias), then

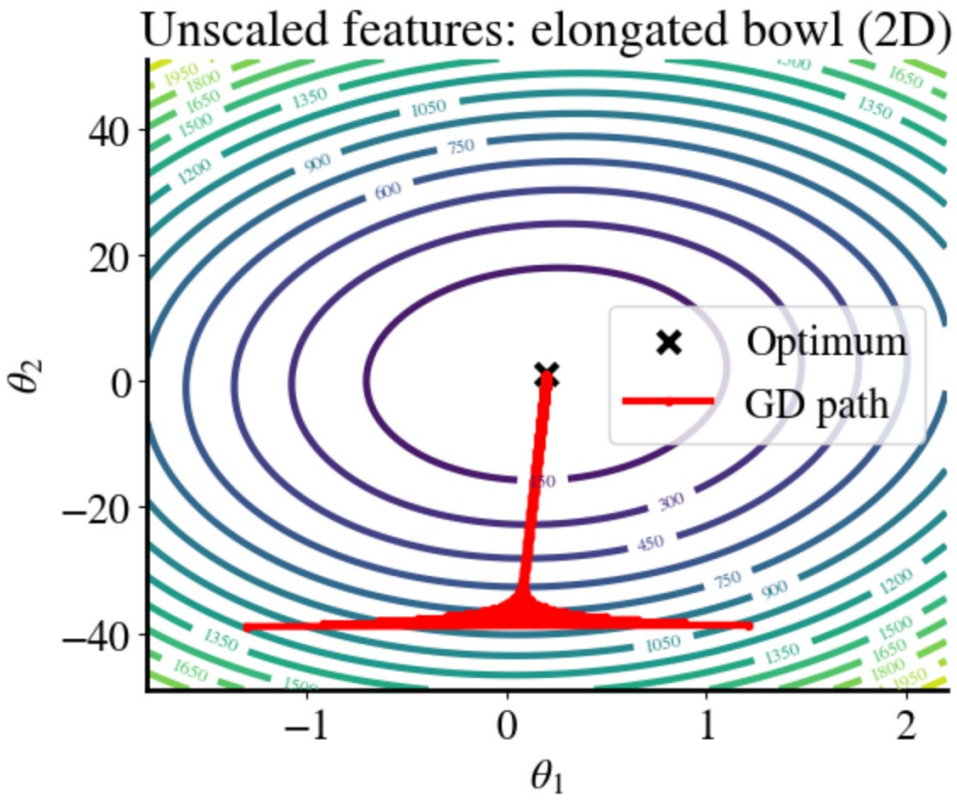
```
[ ]  1  # Feature array X
     2  max_of_each_feature = np.amax(X, axis=0) # axis=0 gives max along columns
     3  X = X/max_of_each_feature
```

Rescaled Polynomial Features

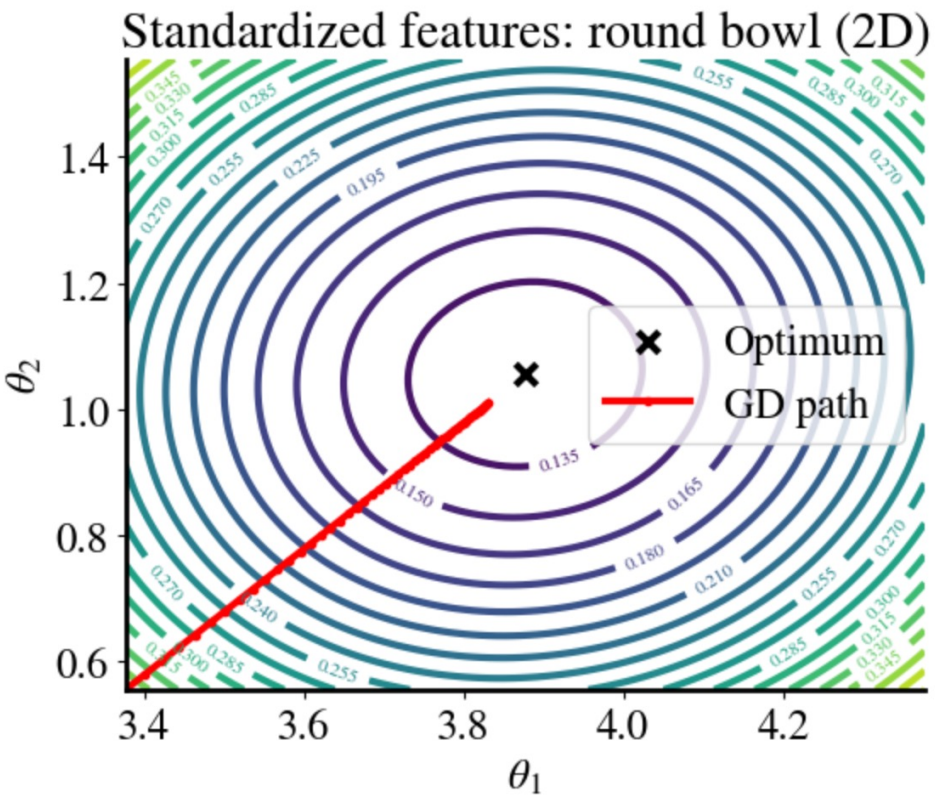


Impact on Gradient Descent

$0 < x_1 < 20, \quad 0 < x_2 < 1$

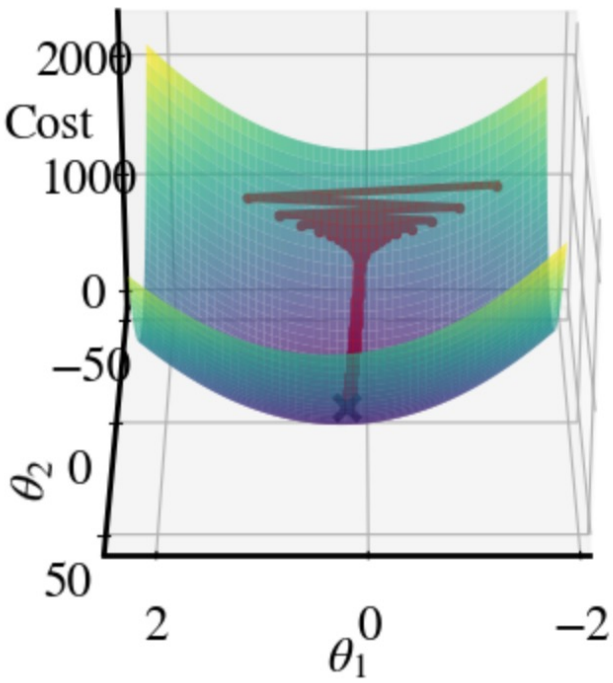


$0 < x_1 < 1, \quad 0 < x_2 < 1$

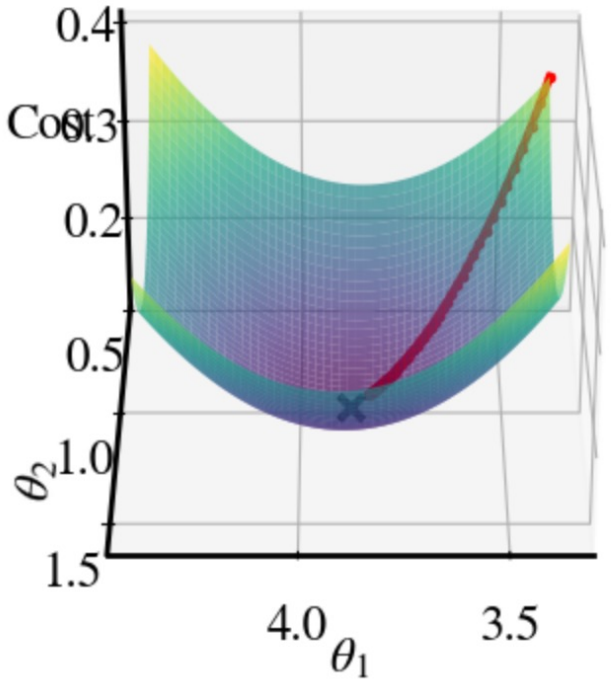


Impact on Gradient Descent

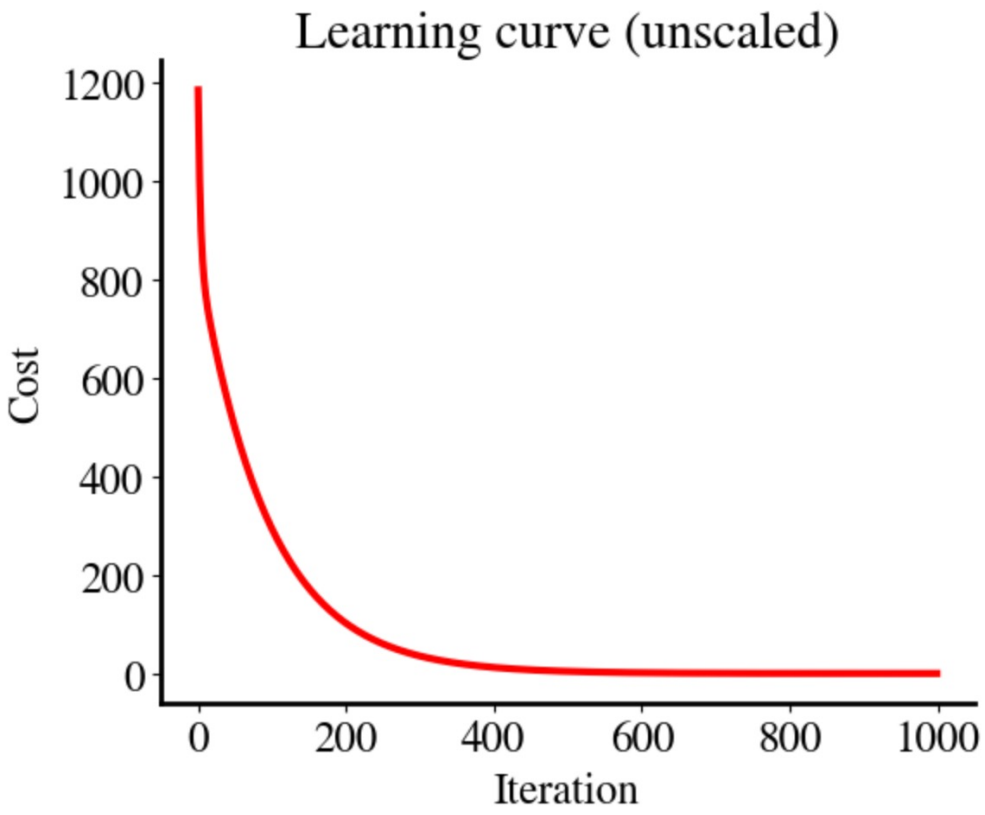
Unscaled features (3D)

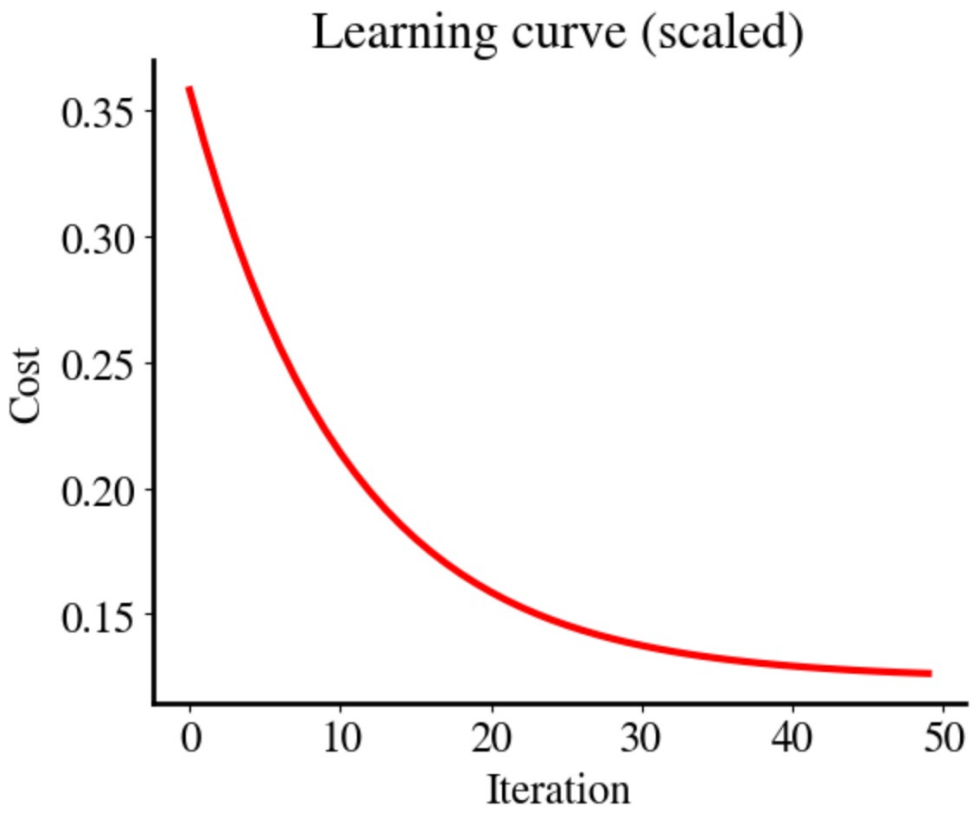


Standardized features (3D)



Learning Curves





Complementary Solution

- Different and **adaptive learning rates for different features**
- During gradient descent, keep track of gradients in each features
- Adjust learning rates accordingly
- Idea behind common optimization schemes like ‘Adam’

Summary: Pipelines

- For ‘Big Data’, data preprocessing generally done in (on-the-fly) **pipelines**
  - **Feature engineering**: what features to (not) include?
  - Data acquisition, e.g. from web, HPC storage, data center
  - Fuse multimodal data into suitable python structure (e.g., pandas)
  - Fix NaN, **impute** missing values, **one-hot-encode** categorical values; output pure numerical numpy array
  - Or xarray for geospatial data where pixels have coordinates and metadata
  - Remove outliers or reduce noise
  - **Feature scaling**; for satellite imagery perhaps histogram matching
  - Crop/clip data into smaller chunks if necessary for RAM memory
  - **Image/data augmentation**: generate additional synthetic training data
  - Split into training and test data; consider K-fold CV training-validation splits
  - Select hyperparameters and optimize during CV (next week), etc.