1. **Data Collection**
2. **Data Exploration**
   - Correlation
   - Distribution
   - Feature engineering
3. **Data Processing**
   - Numeric value (missing value)
     - Interpolation
     - Scaling problem
   - Non-Numeric value (Categorical variable)
     - Drop
     - Label Encoding
     - One-Hot Encoding
4. **Train and Evaluating**
   - Supervised Learning
   - Unsupervised Learning
   - Semi-supervised Learning

5. **Fine-Tune**
   - Optimal hyperparameter

Skill:
- Sklearn, numpy, pandas
- python

ML:
- Algorithm
- Cost function (utility function)
- Evaluation

Linear Regression – linear datasets

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

$y = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$

Essentiality : weight * features+ bias

Performance measure:

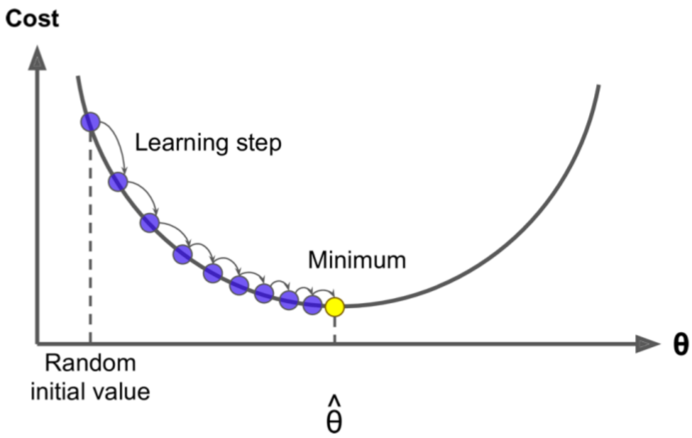
- MSE

$$MSE(X, h_\theta) = \frac{1}{m_i}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\right)^2$$

- RMSE(Root Mean Square Error)
- R2

Two way to train:

- a direct "closed-form" equation
    - the model parameters that minimize **the cost function** over the training set
    - Normal Equation
        - $\hat{\theta} = (X^T X^{-1}) X^T y$
        - SVD (Singular Value Decomposition)
- an iterative optimization approach (Gradient Descent)
    - tweak parameters iteratively in order to minimize a cost function
        - measures the local gradient of the error function with regards to the parameter vector **θ**,
        - goes in the direction of descending gradient
        - the gradient is zero = reach a minimum （global or local）

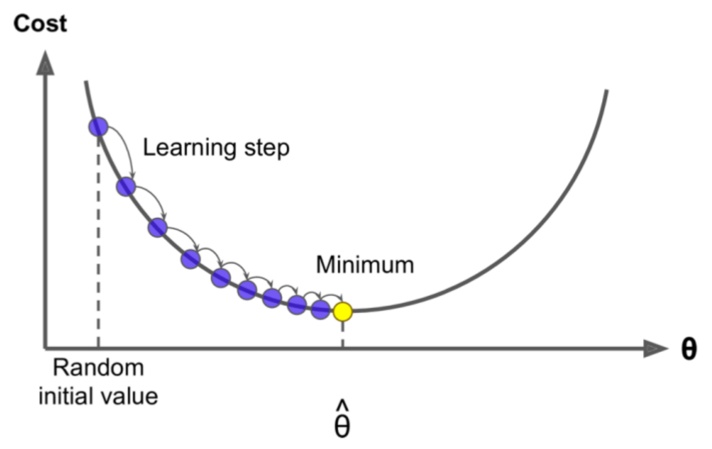


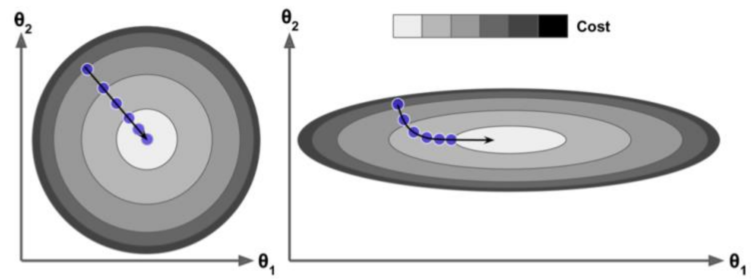
$$\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0}\mathrm{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1}\mathrm{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n}\mathrm{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta})$$

    - **notice**
        - learning rate – how far a step will take
            - gradient – the direction at which the variable varies fastest
            - use grid search to find a good learning rate
                - tolerance – almost converge (minimum)
        - features scale
            - normalization

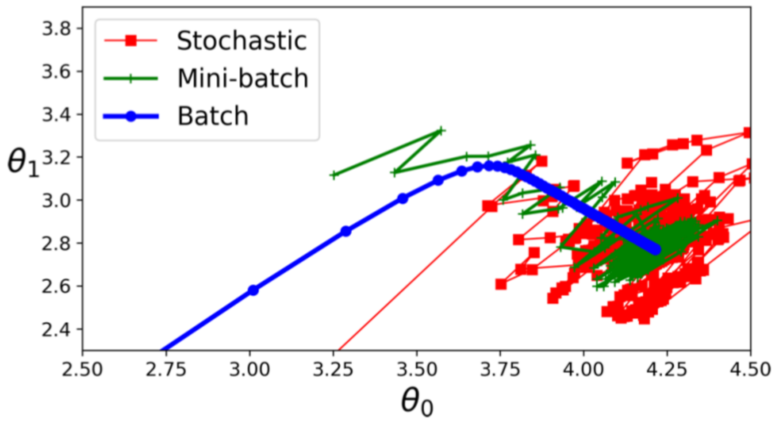
- standardization
- zero-centered



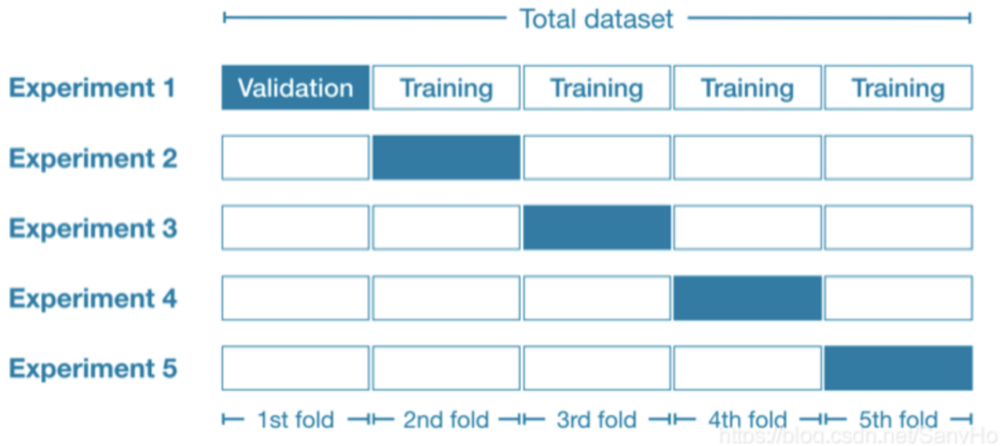


- Batch GD, Mini-batch GD, Stochastic GD
  - Batch GD -- the whole training set to compute the gradients at every step
  - Stochastic GD -- a random instance in the training set at every step
  - Mini-batch GD -- small random sets of instances

Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	n/a
SVD	Fast	No	Slow	0	No	`LinearRegression`
Batch GD	Slow	No	Fast	2	Yes	`SGDRegressor`
Stochastic GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`



Cross-validation

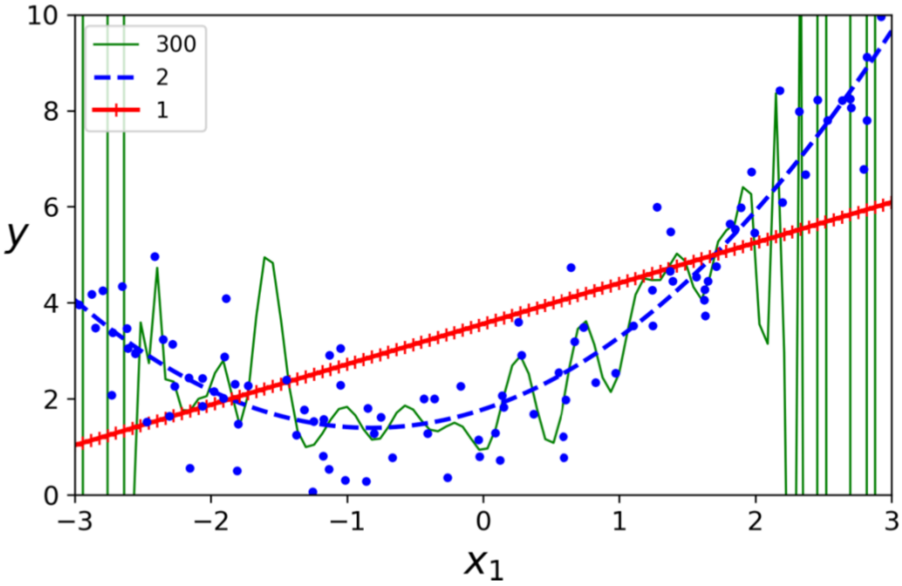


Polynomial Regression – non-linear datasets

Essence: add power of each feature as new features, then train a linear model on the augmented data set.

Drawback:

- Prone to be overfitting the training data
    - Learning curves
    - Regularization techniques



Tradeoff : bias, variance and irreducible error(noisiness)

Fine-Tune : search the optimal hyperparameter of the models

- **Grid Search:** evaluate all the listed combinations of hyperparameter values, using cross-validation
- **Randomized Search:** evaluate a given number of random combinations by selecting a random value for each hyperparameter at every iteration

Learning Curves: evaluate the model's generalization ability

- Underfitting : both curves reach a plateau, close and fairly high
- Overfitting : gap between the curves, error lower

Regularized Linear Models:

- the fewer degrees of freedom it has, the harder it will be for it to overfit the data
- Ridge Regression, Lasso Regression and Elastic Net
  - Ridge Regression
    - $\boldsymbol{cost\ function}:\ J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$
    - $\boldsymbol{regularized\ term}(L_2): \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2 = \frac{1}{2}\left(||w||_2\right)^2$
    - bias, variance and irreducible error(noisiness)
      - α – reduced variance, increase bias
    - Sensitive to the scale of the input features (StandardScaler)
  - Lasso Regression
    - $\boldsymbol{cost\ function}:\ J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha\sum_{i=1}^{n}|\theta_i|$
    - regularized term (L_1) : $\alpha\sum_{i=1}^{n}|\theta_i|$

        - tend to completely eliminate the weights of the least important features
        - performs feature selection and outputs a *sparse model*
    - Elastic Net
        - $\boldsymbol{cost\ function}: \boldsymbol{J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^{n} \theta_i^2}$
        - A middle ground between Ridge Regression and Lasso
    - Plain Linear Regression < Ridge(regularized) < Lasso(feature selection, but erratically when number of features greater or some features strongly correlated) < Elastic Net
    -

Early Stopping : prevent the model from overfitting, get the best model