# Practical Methodology

ESM5205 Learning from Big Data | Oct 2, 2019

Seokho Kang

성균관대학교
SUNG KYUN KWAN UNIVERSITY(SKKU)

# In Practice…

- **Practitioners need to know…**
  - how to choose an algorithm for a particular application?
  - how to monitor and respond to feedback obtained from experiments in order to improve a machine learning system?
  - whether to gather more data, increase or decrease model capacity, add or remove regularizing features, improve the optimization of a model, improve approximate inference in a model, or debug the software implementation of the model.

- Blindly applying an algorithm to a dataset **without understanding the assumptions the model makes and the meanings of the hyperparameter settings** will **rarely lead to an accurate model.**
- One can usually do much better with **a correct application of a commonplace algorithm** than by **sloppily applying an obscure algorithm**.

# Practical Process

1. **Determine your goals**
2. **Build an end-to-end system**
3. **Refine the system**

# 1. Determine your goals

- **Task (to be addressed)**
- **Data**
  - **The amount of available data**
  - **Input and Output variables**
  - **Characteristics** (class imbalance, etc.)
  - **Preprocessing *****
- **Requirements**
  - **Computational Resources:** CPU/GPU, Memory, Storage, etc.
  - **Practical Constraints:** Time, Speed, etc.
- **Performance Metrics**
  - **Which metric to use?** (depending on the target task)
  - **What level of performance do we desire?** (practically acceptable level for the task)

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for Classification Tasks**

  **Given a test set $\boldsymbol{D'} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$**

  - **Accuracy**: the fraction of correctly classified data points

$$\text{Accuracy} = \frac{1}{n}\sum_{i} \mathrm{I}(y_i = \hat{y}_i) \times 100\%$$

  - **Error Rate**: the fraction of mis-classified data points

$$\text{Error Rate} = 1 - \text{Accuracy}$$

*accuracy and error rate under class imbalance scenarios?*

  - **Confusion Matrix**

    *Positive Class (Our Main Interest), Negative Class*

    *FP: Type 1 Error, FN: Type 2 Error*

| | | Predicted class: $P$ | Predicted class: $N$ |
|---|---|---|---|
| Actual Class | $P$ | True Positives (TP) | False Negatives (FN) |
| | $N$ | False Positives (FP) | True Negatives (TN) |

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for Classification Tasks: Class Imbalance**

  ***Example***: *Two different models for the binary classification of 3000 data points, which model is better?*

**Model 1**

| | Predicted Positive | Predicted Negative |
|---|---|---|
| **Actual Positive** | 26 | 260 |
| **Actual Negative** | 214 | 2500 |

**Accuracy** = (26+2500)/3000 = 84.20%

**Precision** = 26/(26+214) = 10.83%

**Recall** = 26/(26+260) = 9.09%

**Model 2**

| | Predicted Positive | Predicted Negative |
|---|---|---|
| **Actual Positive** | 0 | 286 |
| **Actual Negative** | 0 | 2714 |

**Accuracy** = (0+2714)/3000 = 90.47%

**Precision** = 0/(0+0) = NaN

**Recall** = 0/(0+286) = 0%

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for (Binary) Classification Tasks**

  **Given a test set $\boldsymbol{D'} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$**

  - **Precision**: the fraction of predicted positives that are actually positive

    **Recall**: the fraction of actual positives that are predicted as positive

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

  - **$F_1$ Score**: the harmonic mean of Precision and Recall

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

  - **Sensitivity(=Recall):** the fraction of actual positives that are correctly classified

    **Specificity:** the fraction of actual negatives that are correctly classified

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

| | | Predicted class: $P$ | Predicted class: $N$ |
|---|---|---|---|
| Actual Class | $P$ | True Positives (TP) | False Negatives (FN) |
| | $N$ | False Positives (FP) | True Negatives (TN) |

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for (Binary) Classification Tasks: Threshold**
  - **Most classification algorithms classify each data point $x$ by …**
    - Compute $P(y = 1|\boldsymbol{x})$, the probability of belonging to class "1"
    - Compare to the threshold, and classify it accordingly
  - **In most cases, the default threshold is 0.5.**
    - If > 0.5, classify as "1", If < 0.5, classify as "0"

      *what if the threshold is 0 or 1?*
  - **The threshold controls the trade-off between …**
    - precision and recall
    - sensitivity and specificity
  - **We can use different thresholds.**

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for (Binary) Classification Tasks: Threshold**

  ***Example**: Various thresholds for the binary classification of 20 data points*

$P(y = 1|x)$

(1:positive, 0:negative)

| Actual Class | Prob. of "1" | Actual Class | Prob. of "1" |
|---|---|---|---|
| 1 | 0.996 | 1 | 0.506 |
| 1 | 0.988 | 0 | 0.471 |
| 1 | 0.984 | 0 | 0.337 |
| 1 | 0.980 | 1 | 0.218 |
| 1 | 0.948 | 0 | 0.199 |
| 1 | 0.889 | 0 | 0.149 |
| 1 | 0.848 | 0 | 0.048 |
| 0 | 0.762 | 0 | 0.038 |
| 1 | 0.707 | 0 | 0.025 |
| 1 | 0.681 | 0 | 0.022 |
| 1 | 0.656 | 0 | 0.016 |
| 0 | 0.622 | 0 | 0.004 |

**when threshold=0**

| | Predicted Positive | Predicted Negative |
|---|---|---|
| **Actual Positive** | 12 | 0 |
| **Actual Negative** | 12 | 0 |

**when threshold=0.25**

| | Predicted Positive | Predicted Negative |
|---|---|---|
| **Actual Positive** | 11 | 1 |
| **Actual Negative** | 4 | 8 |

**when threshold=1**

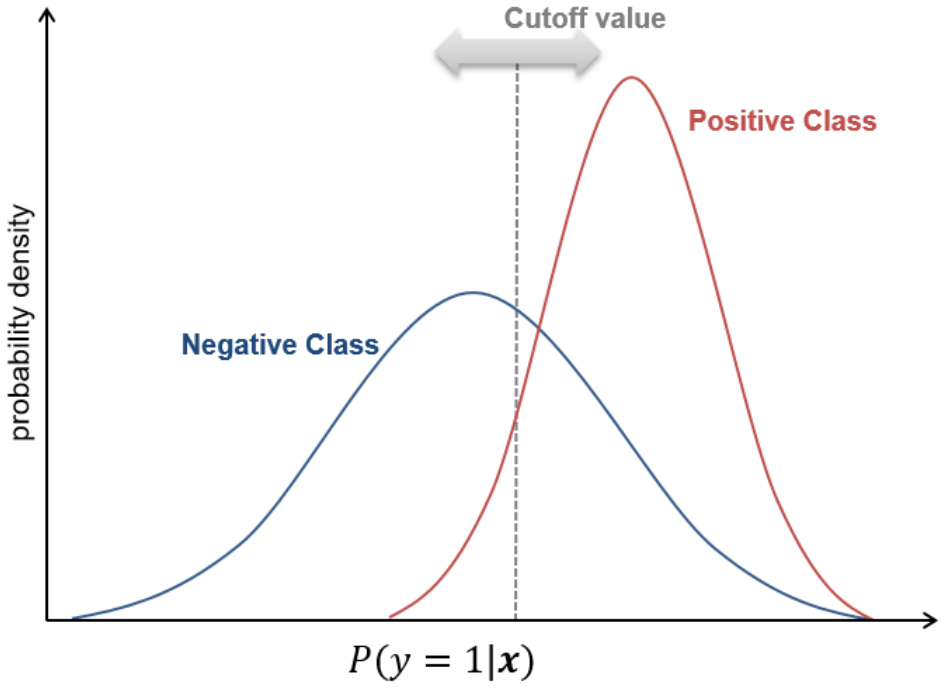
| | Predicted Positive | Predicted Negative |
|---|---|---|
| **Actual Positive** | 0 | 12 |
| **Actual Negative** | 0 | 12 |

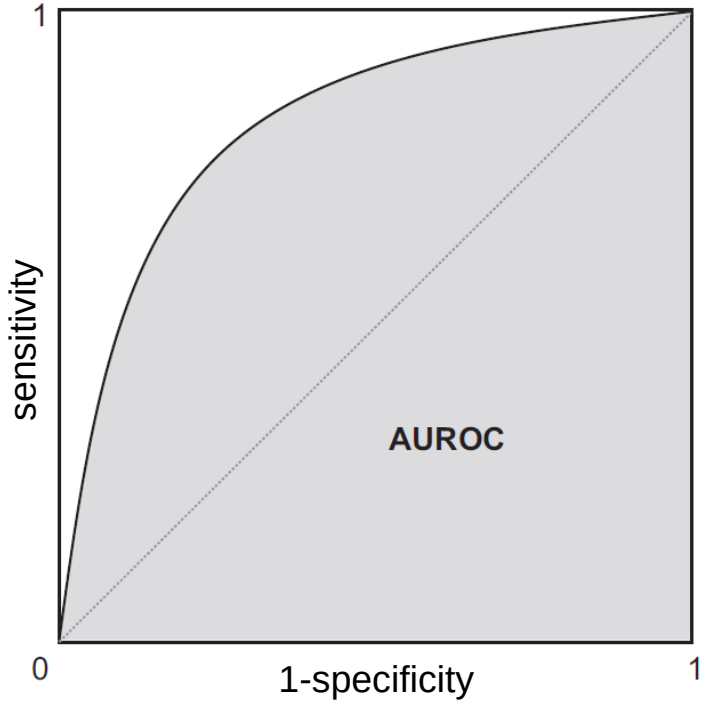
**when threshold=0.75**

| | Predicted Positive | Predicted Negative |
|---|---|---|
| **Actual Positive** | 7 | 5 |
| **Actual Negative** | 1 | 11 |

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for (Binary) Classification Tasks: Receiver Operating Characteristic**
  - **Receiver Operating Characteristic (ROC) Curve**: Performance evaluation by plotting ***sensitivity*** against ***1-specificity*** at various thresholds
  - **AUROC:** Area under the ROC curve
    - To assess how well a model performs in general (threshold independent metric)
    - Widely used for the evaluation of **imbalanced classification tasks**



# 1. Determine your goals: Performance Metrics

- **Performance Metrics for Regression Tasks**

  **Given a test set $\boldsymbol{D'} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$,**

  - **Root Mean Squared Error (RMSE)**

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2}$$

  - **Mean Absolute Error (MAE)**

$$\mathrm{MAE} = \frac{1}{n}\sum_i |y_i - \hat{y}_i|$$

  - **Mean Absolute Percentage Error (MAPE)**

$$\mathrm{MAPE} = \frac{1}{n}\sum_i \left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

*ratio scale vs interval scale?*

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for Regression Tasks: Interval Scale vs Ratio Scale**
  - **Interval Scale** – has no absolute zero, allows for the degree of difference between magnitudes, but not the ratio (*e.g.*, temperature, …)
    - **Example:** If we use MAPE to evaluate temperature predictions,

      When the true value is 1°C and its prediction is 0°C → APE=100%

      When the true value is 100°C and its prediction is 101°C → APE=1%

      When the true value is 0°C?

  - **Ratio Scale** – possess an absolute zero, allows for the degrees of both difference and ratio between magnitudes (*e.g.*, length, weight, …)
    - **Example:** If we use MAE to evaluate weight predictions,

      When the true value is 1g and its prediction is 2g → AE=1g

      When the true value is 10,000g and its prediction is 10,001g → AE=1g

# 1. Determine your goals: Performance Metrics

- **Performance Metrics for Regression Tasks: MAE vs RMSE**
  - RMSE penalizes more on large errors

**CASE 1: Evenly distributed errors**

| *ID* | Error | \|Error\| | Error^2 |
|---|---|---|---|
| *1* | 2 | 2 | 4 |
| *2* | 2 | 2 | 4 |
| *3* | 2 | 2 | 4 |
| *4* | 2 | 2 | 4 |
| *5* | 2 | 2 | 4 |
| *6* | 2 | 2 | 4 |
| *7* | 2 | 2 | 4 |
| *8* | 2 | 2 | 4 |
| *9* | 2 | 2 | 4 |
| *10* | 2 | 2 | 4 |

| MAE | RMSE |
|---|---|
| 2.000 | 2.000 |

**CASE 2: Small variance in errors**

| *ID* | Error | \|Error\| | Error^2 |
|---|---|---|---|
| *1* | 1 | 1 | 1 |
| *2* | 1 | 1 | 1 |
| *3* | 1 | 1 | 1 |
| *4* | 1 | 1 | 1 |
| *5* | 1 | 1 | 1 |
| *6* | 3 | 3 | 9 |
| *7* | 3 | 3 | 9 |
| *8* | 3 | 3 | 9 |
| *9* | 3 | 3 | 9 |
| *10* | 3 | 3 | 9 |

| MAE | RMSE |
|---|---|
| 2.000 | 2.236 |

**CASE 3: Large error outlier**

| *ID* | Error | \|Error\| | Error^2 |
|---|---|---|---|
| *1* | 0 | 0 | 0 |
| *2* | 0 | 0 | 0 |
| *3* | 0 | 0 | 0 |
| *4* | 0 | 0 | 0 |
| *5* | 0 | 0 | 0 |
| *6* | 0 | 0 | 0 |
| *7* | 0 | 0 | 0 |
| *8* | 0 | 0 | 0 |
| *9* | 0 | 0 | 0 |
| *10* | 20 | 20 | 400 |

| MAE | RMSE |
|---|---|
| 2.000 | 6.325 |

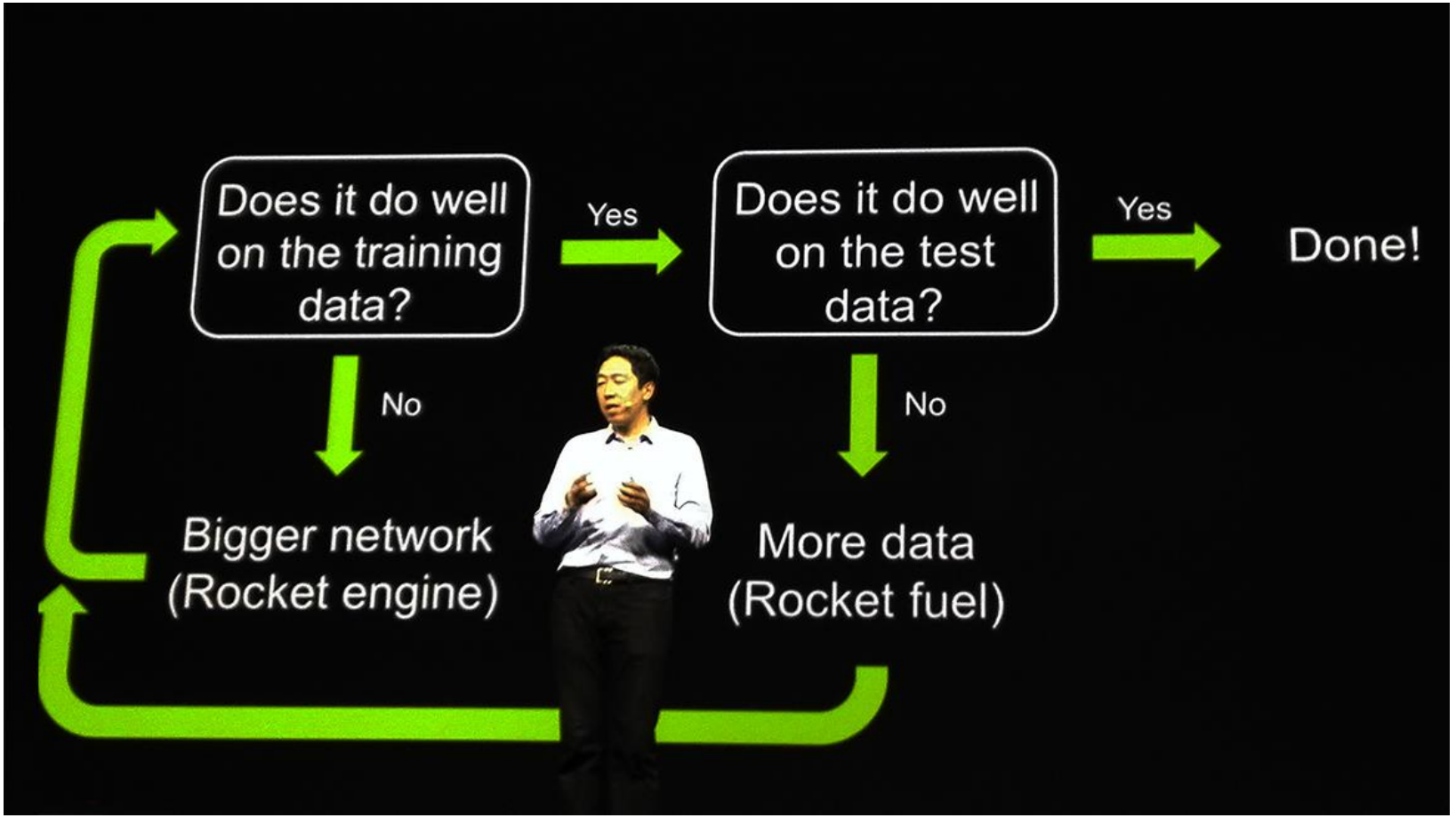
https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

# 2. Establish an end-to-end pipeline

- **Build a reasonable system ASAP as baseline**
  - Any viable system
  - or, copying the state-of-the-art system from a related publication

- **Which learning algorithm to use?**
  - Do we really need deep learning systems?
    (*e.g.*, large-scale data, complex task, previous success of deep learning)
  - Deep or not?
    - Lots of noise, little structure → shallow learning
    - Little noise, complex structure → deep learning
  - In general, it is better to start with simple and robust algorithms
    (*e.g.*, linear regression, decision tree ensemble, shallow neural network)

# 3. Refine the system




Source: GTC 2015 Keynote with Dr. Andrew Ng

# 3. Refine the system: Data

- **Determining whether to gather more data**
  - **If the performance on the training set is poor?**
    - Inspect your data and your code
    - Try increasing the representational capacity of a learning algorithm and improving optimization → hyperparameter tuning
    - **If still poor?** The problem might be quality of training data (noise, lack of variables, etc.)
  - **If the performance on the training set is acceptable, but on the test set is poor?**
    - Try improving regularization → hyperparameter tuning
    - **If still poor?** Gathering more data is one of the most effective solutions
  - **If the performance on the test set is acceptable,**
    - There is noting left to be done!

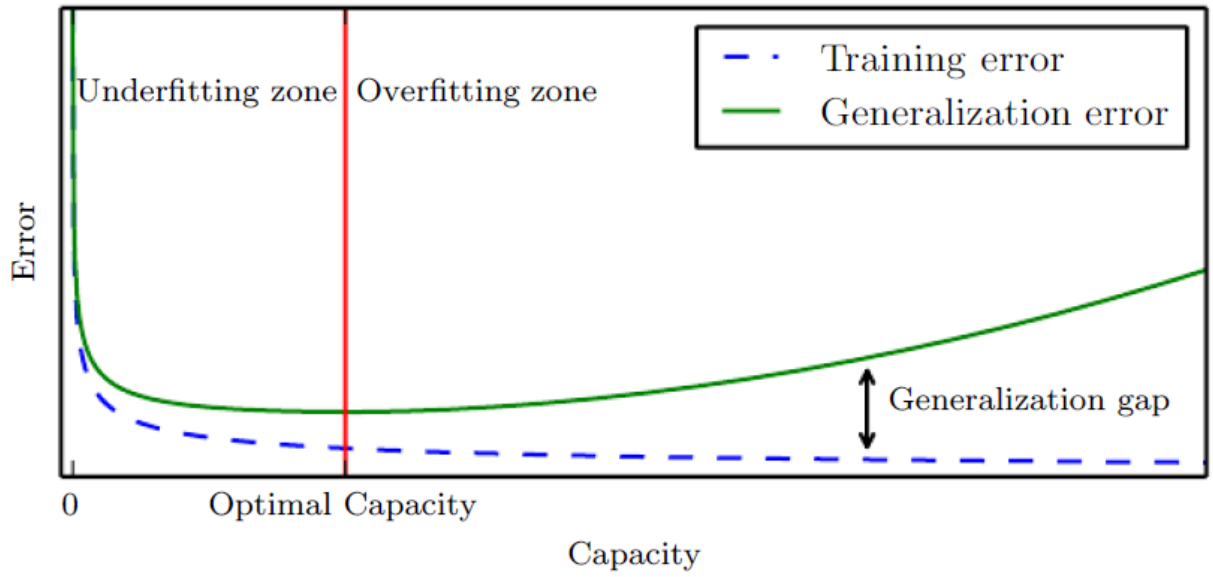
*** Choose what to do based on "data"**

# 3. Refine the system: Hyperparameters

- **The goal of hyperparameter tuning is to find the lowest generalization error subject to some runtime and memory budget.**
  - Hyperparameters control many aspects of a learning algorithm's behavior, including the representational capacity, optimization, and regularization.
  - Usually, the best generalization performance comes from a large model that is regularized well.
  - **manual hyperparameter tuning** and **automatic hyperparameter tuning**

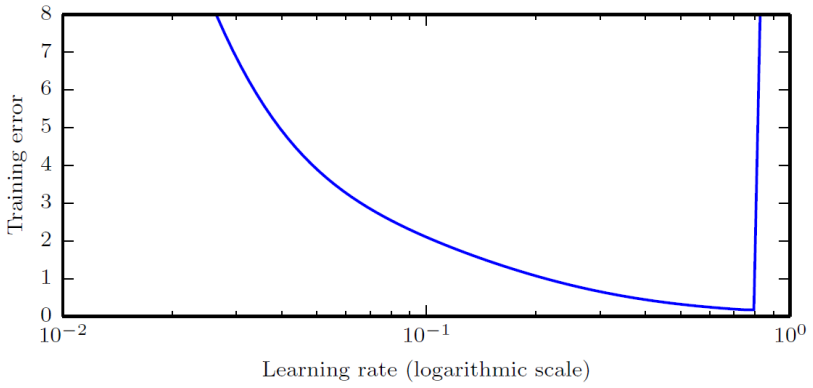
# 3. Refine the system: Hyperparameters

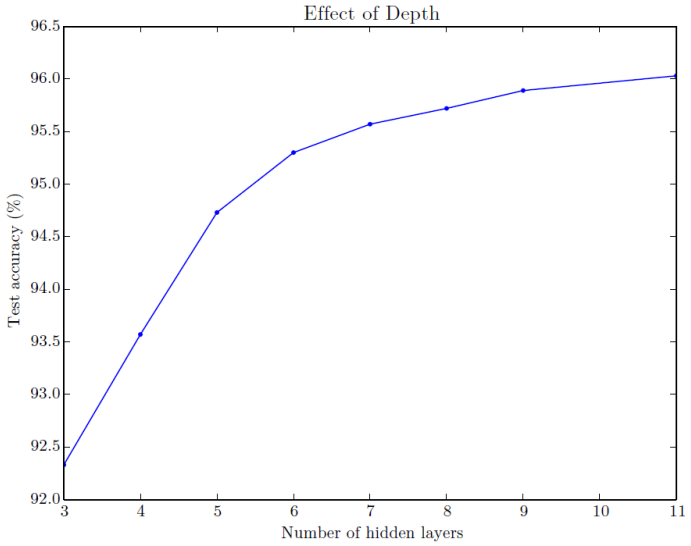
- **Manual Hyperparameter Tuning**
  - Requires a good understanding of the relationships between hyperparameters, representational capacity, optimization, and regularization
  - Can work very well when we have good starting points (experience, related publications, etc.)



# 3. Refine the system: Hyperparameters

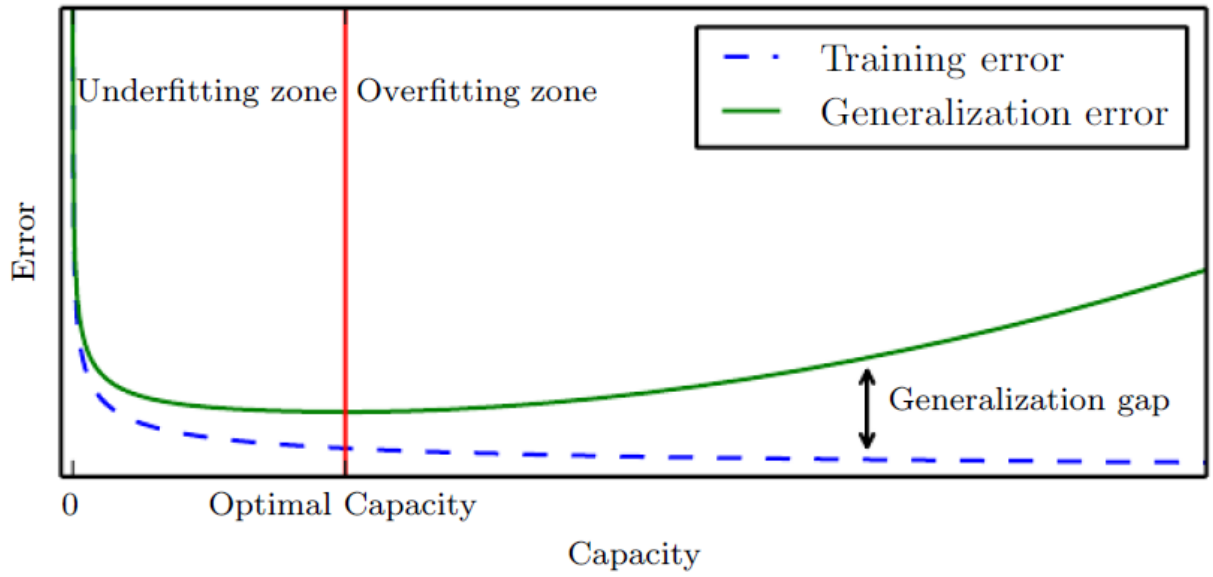
- **Manual Hyperparameter Tuning**
  - ***Examples:* deep neural network**
    - **Number of hidden layers/units:** Increasing the number of hidden layers/units increases the representational capacity
    - **Learning rate:** An improper learning rate, whether too high or too low, results in low representation capacity due to optimization failure
    - **Weight decay coefficient:** Decreasing the weight decay coefficient frees the model parameters to become larger
    - **Dropout rate:** Dropping units less often gives the units more opportunities to conspire with each other to fit the training set



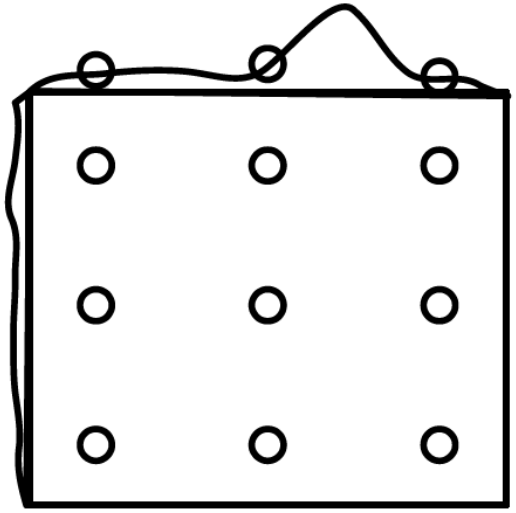
# 3. Refine the system: Hyperparameters

- **Automatic Hyperparameter Tuning:** Hyperparameter Optimization
  - Reduce the need to understand the hyperparameters, but computationally expensive
  - Can be used when good starting points are not available



# 3. Refine the system: Hyperparameters

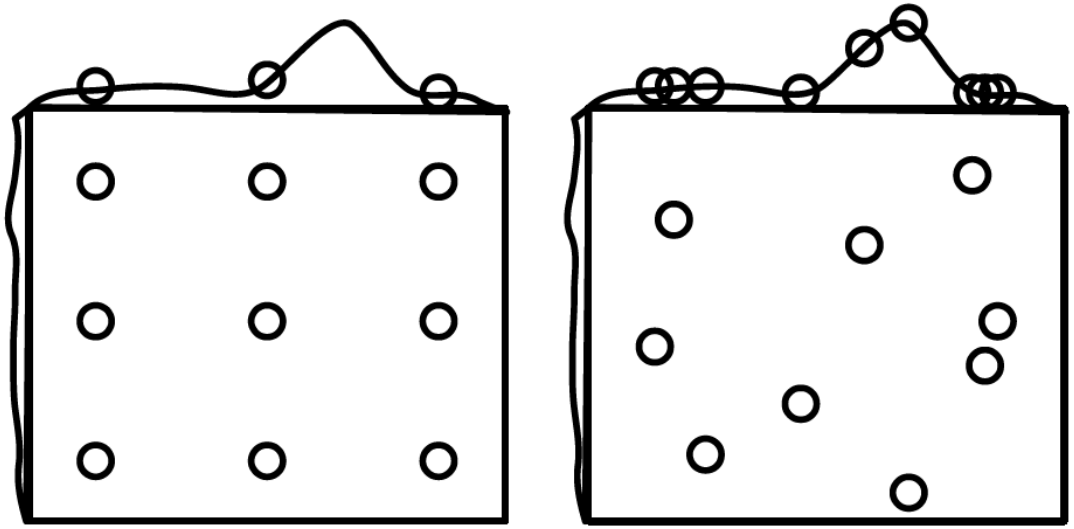
- **Automatic Hyperparameter Tuning:** Hyperparameter Optimization
  - **Grid Search**
    - Exhaustive searching through a manually specified subset of the hyperparameter space
    - The subset is the Cartesian product of the set of values for each individual hyperparameter
    - Computational cost grows exponentially with the number of hyperparameters.
      : if there are *m* hyperparameters, each taking at *n* values, then the number of trials is $n^m$.
    - All the trials are independent → Easy to parallelize



Grid

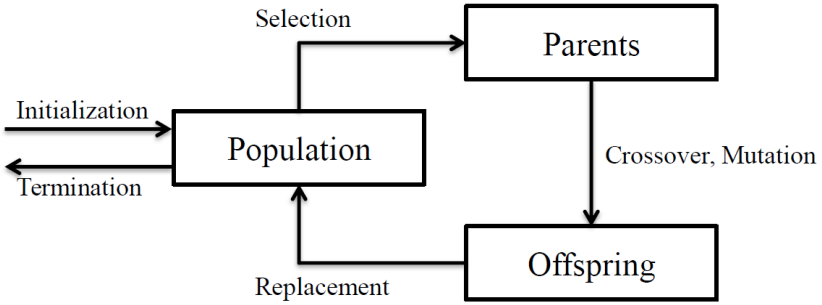
# 3. Refine the system: Hyperparameters

- **Automatic Hyperparameter Tuning:** Hyperparameter Optimization
  - **Random Search**
    - Search by random sampling from the hyperparameter space
    - Random search is often more efficient than grid search.
    - Prior knowledge can be incorporated by specifying the distribution from which to sample.
    - All the trials are independent → Easy to parallelize

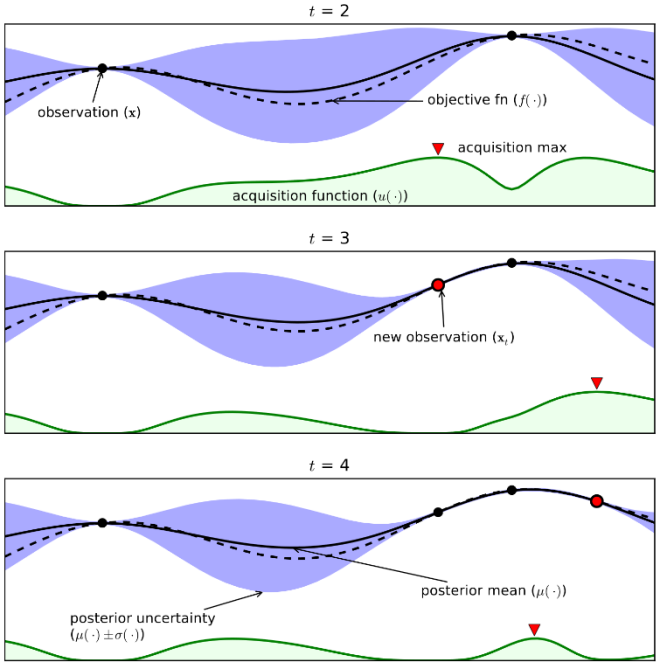


# 3. Refine the system: Hyperparameters

- **Automatic Hyperparameter Tuning:** Hyperparameter Optimization
  - **Meta-heuristics**
    : efficiently explore the hyperparameter space to find near-optimal solution (*e.g.,* Genetic algorithm)




  - **Model-based Optimization**
    **:** find the functional relationship between hyperparameters and validation performance (*e.g.,* Bayesian optimization)



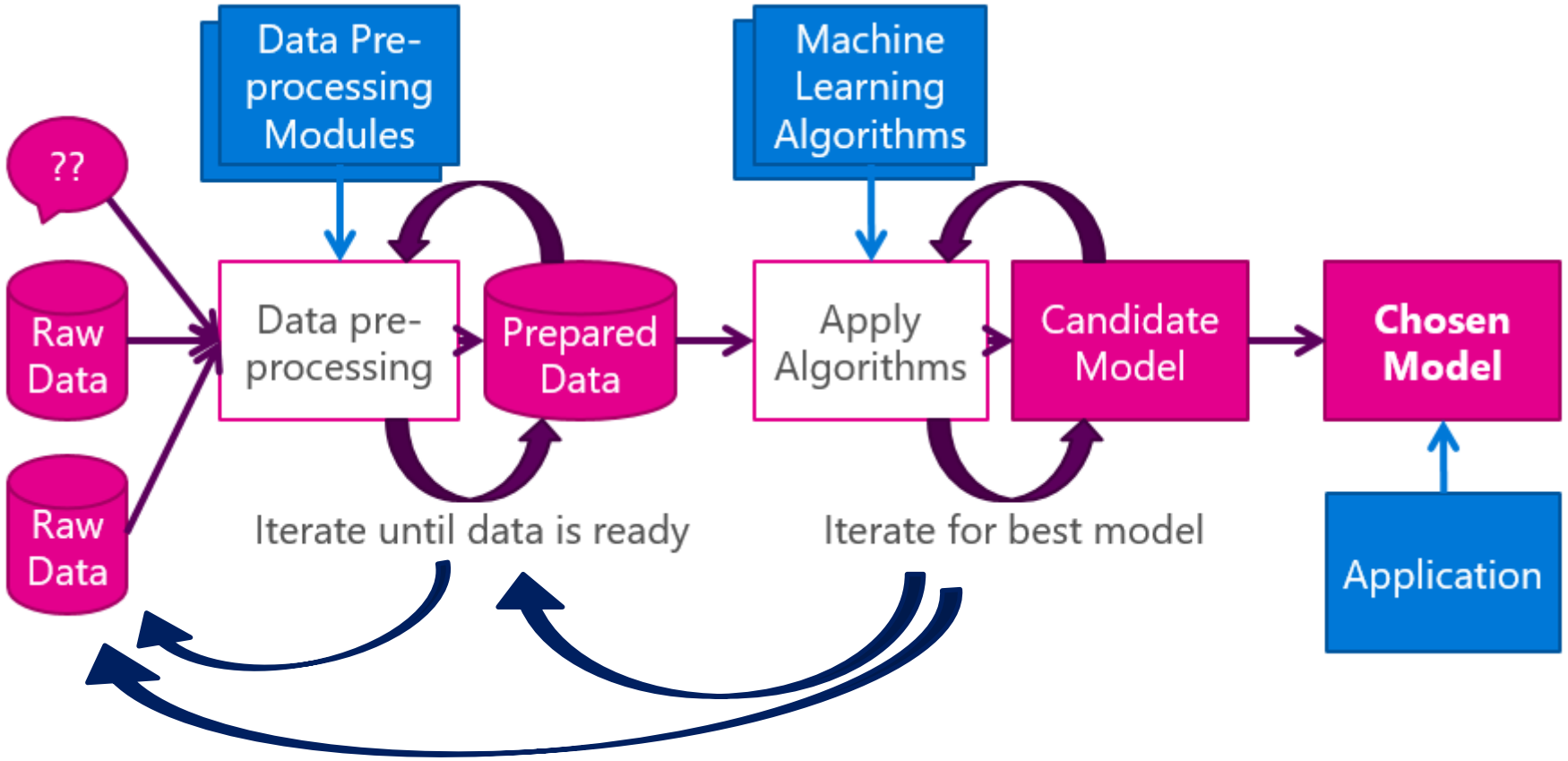
# Debugging

- **It's usually difficult to find the reason why a machine learning system doesn't work or performs poorly.**
  - In most cases, we don't know the expected behavior or suboptimal behavior of the system.
  - Multiple components in the system can adapt each other. If one component is broken, the other components can adapt.
  - Sometimes, there's a software/hardware defect.

- **If things are problematic, you should…**
  - Inspect data / software / hardware
  - Fit a small dataset
  - Visualize the model in action
  - Visualize the worst mistakes
  - …

# Machine Learning Practice

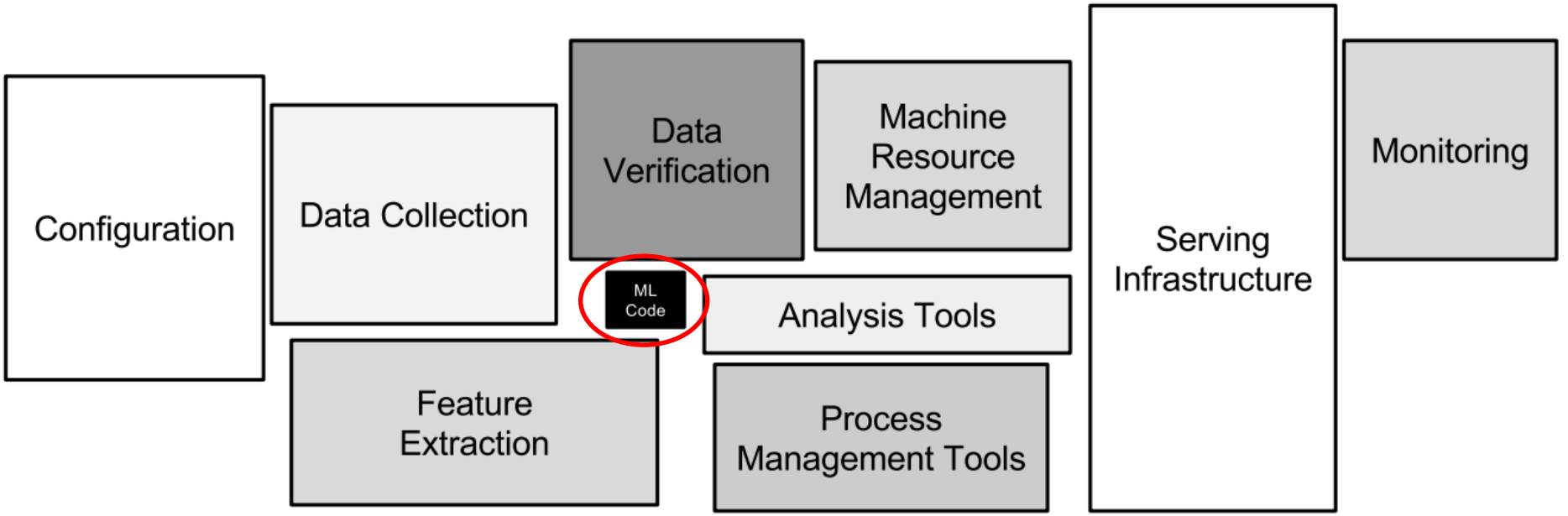
- **Typical ML-based modeling strategy**




Source from http://martink.me/articles/machine-learning-is-for-muggles-too

# Machine Learning Practice

- **Real-world ML system**




Sculley, David, et al. "Hidden technical debt in machine learning systems." *Advances in neural information processing systems*. 2015.