



Statistics
Canada

Statistique
Canada



Canada

Statistics Canada
www.statcan.gc.ca

What is *Supervised Machine Learning?*

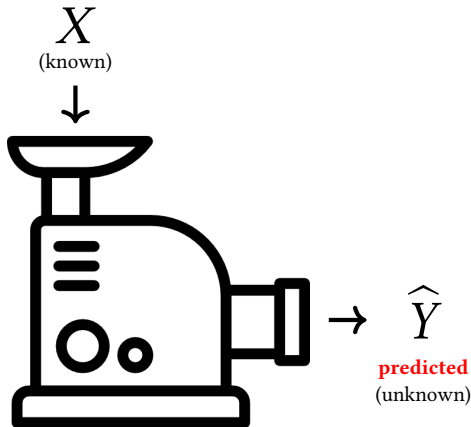
prediction, overfitting, cross validation, hyperparameter tuning

Kenneth Chu

Enquêtes spéciales, transport, technologie et assurance de la qualité, DMEE
Special Surveys, Transportation, Technology and Quality Assurance, BSMD

October 4, 2017

What is *Supervised Machine Learning*?



Icon made by Linear Household Elements from www.flaticon.com

How to build such a
prediction machine?

*“Training” it by “fitting” it
to observed (past) data :*

(x_1, y_1)

(x_2, y_2)

\vdots

(x_n, y_n)

Want :

Error(new data)
to be “small”

Generic *Supervised Machine Learning* workflow

**Goal : To produce a prediction machine
with small **Error(new data)****


1. Visualize data. Exploratory Data Analysis.
2. Randomly split available data into **training**, **validation**, and **testing** subsets.
3. Preprocess training + validation data. (Standardize, E&I, transform, etc.)
4. Choose model. Train model on training set.
 - **Hyperparameter tuning** if applicable
 - Evaluate/compare models[†] based on Error(validation set).
 - **k-fold cross validation** if deemed appropriate
 - Retrain model on training + validation data, using chosen hyperparameter values \leadsto final “trained machine”
5. Evaluate final trained machine based on **Error(testing set)**.

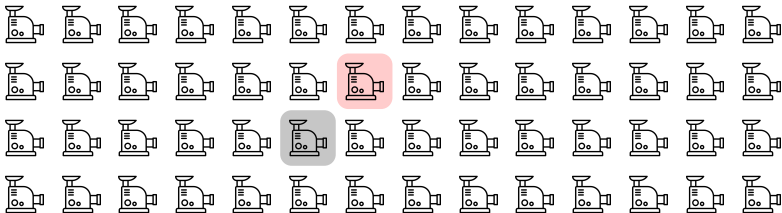
[†]. more precisely, the constituent data-generation mechanisms in the chosen model

What is *training*?


Want : a machine to make prediction \hat{Y} of unknown Y based on observed X

Training data : $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$

Assume : True underlying *random* data-generation mechanism  belongs to a chosen class (the **model**) of similar mechanisms .



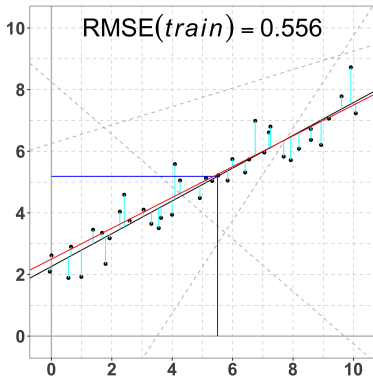
Icon made by Linear Household Elements from www.flaticon.com

Training : Find the member  in model that “best” fits training data.

Need to solve the associated **optimization** problem.

(Choose **cost function** defined on model. Minimize cost function.)

Sounds familiar ? *Recall Simple Linear Regression*



Cost function : Root Mean Squared Error

$$f(\beta_0, \beta_1) = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2}$$

Model (class of admissible underlying
data-generation mechanisms) :

$$Y = \beta_0 + \beta_1 x + \text{noise}$$

$$E[Y | x] = \beta_0 + \beta_1 x$$

where $(\beta_0, \beta_1) \in \mathbb{R}^2 =: \Theta$.

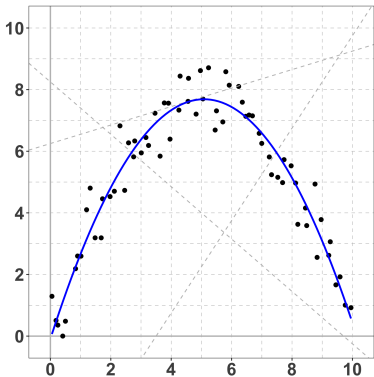
Model is parametrized by $\Theta := \mathbb{R}^2$.

Model is 2-dimensional.

$$\text{Model} \sim \left\{ \begin{array}{l} \text{all straight} \\ \text{lines in } \mathbb{R}^2 \end{array} \right\} \sim \left\{ \begin{array}{l} \text{all affine} \\ \text{functions of } x \end{array} \right\}$$

Best fit : Least Squares
(trained prediction machine)

But, not every phenomenon is linear ...



Linear (all affine functions in x) :

$$\begin{aligned} Y &= \beta_0 + \beta_1 x + \text{noise} \\ E[Y | x] &= \beta_0 + \beta_1 x \end{aligned},$$

where $(\beta_0, \beta_1) \in \mathbb{R}^2 =: \Theta$.

Quadratic (all quadratic functions in x) :

$$\begin{aligned} Y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \text{noise} \\ E[Y | x] &= \beta_0 + \beta_1 x + \beta_2 x^2 \end{aligned},$$

where $(\beta_0, \beta_1, \beta_2) \in \mathbb{R}^3 =: \Theta$.

Model needs to be sufficiently complex in order to have a chance to well approximate the true underlying data-generation mechanism.

Insufficient model complexity \leadsto **underfitting**

Common Supervised Machine Learning Techniques

- logistic regression
- nearest neighbour (an imputation technique)
- decision trees (used to form homogeneous response groups)
- random forest
- support vector machines
- artificial neural networks

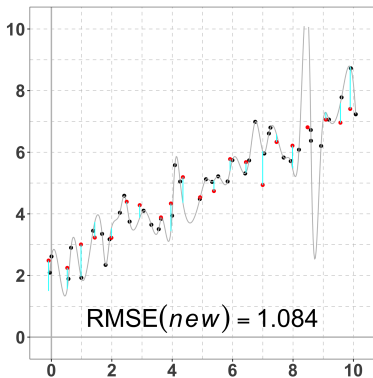
Many supervised machine learning techniques have high model complexity, or a high-dimensional parameter space.

i.e., $\#(\text{observations}) \not\gg \#(\text{model parameters})$

Warning : They may be *too* complex !

Excessive model complexity \leadsto **overfitting**

What is *overfitting*?



$$\text{Model} \sim \left\{ \begin{array}{l} \text{all } \cancel{\text{affine}} \text{ polynomial} \\ \text{functions of } x \end{array} \right\}$$

Cost function : RMSE

$$\text{RMSE} \left(\begin{array}{c} \text{new} \\ \text{data} \end{array} \right) > 0 = \text{RMSE} \left(\begin{array}{c} \text{training} \\ \text{data} \end{array} \right)$$

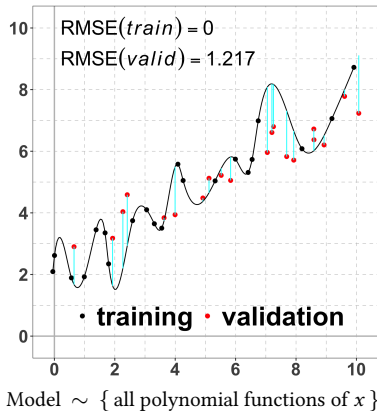
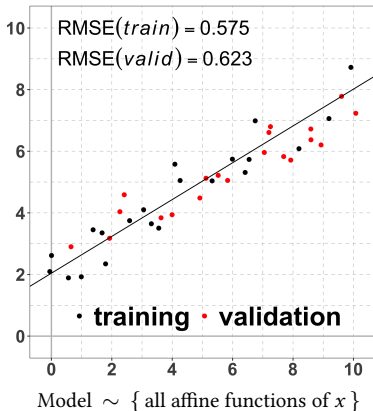
Generally, **overfitting leads to**

$$\text{RMSE} \left(\begin{array}{c} \text{new} \\ \text{data} \end{array} \right) \gg \text{RMSE} \left(\begin{array}{c} \text{training} \\ \text{data} \end{array} \right)$$

- Overfitted machines give poor predictions on new values of x .
- You won't even know it is happening (remember : Y will be unknown).

Validation : mimicking model evaluation on new data ...

- Randomly divide **non-testing** data set into training and validation subsets.
- Train on the training subset.
- Evaluate the trained machine on the validation subset.



k -fold Cross Validation

- Results of validation may be **unstable**.

Put another way, one wonders how “representative” $\text{Error}(\text{validation data})$ is based on just one way of splitting the non-testing data set into training and validation subsets.

- To partially mitigate for this, “repeat” validation k times.
- More precisely :
 - Randomly divide the non-testing data set into k “folds” (subsets) of roughly equal sizes.
 - Perform validation k times, each time using a different fold as validation subset, and the rest of the $k - 1$ folds as training subset.

$$\text{Error} \left(\begin{array}{c} k\text{-fold cross} \\ \text{validation} \end{array} \right) := \left(\begin{array}{c} \text{average of the } k \text{ resulting} \\ \text{validation errors} \end{array} \right)$$

What is *hyperparameter tuning*?

- For many models, there are no well established optimization methods that can optimize on the full set of model parameters.
- For example, the cost function of Ridge Regression can be written as :

$$C(\beta_0, \beta_1, \dots, \beta_{p-1}, \lambda) := \frac{1}{n} \cdot \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{k=1}^{p-1} x_{ik} \beta_k \right)^2 + \lambda \cdot \sum_{k=1}^{p-1} \left| \beta_k \right|^2,$$

where $(\beta_0, \beta_1, \dots, \beta_{p-1}, \lambda) \in \mathbb{R}^p \times [0, \infty) =: \Theta$.

- For each fixed $\lambda \in [0, \infty)$, convex programming (on the primal problem) yields :

$$\hat{\beta}(\lambda) = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} C(\beta, \lambda)$$

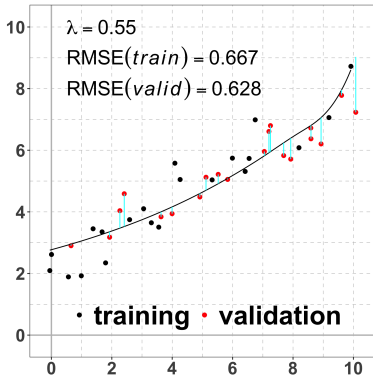
- But, we really want : $\underset{(\beta, \lambda) \in \mathbb{R}^p \times [0, \infty)}{\operatorname{argmin}} C(\beta, \lambda)$

- Here, λ is the hyperparameter.

To optimize on λ , one can perform a “grid search” (or other similar approaches).

This is **hyperparameter tuning**.

What is *hyperparameter tuning* ?



λ	RMSE training	RMSE validation
0	0.000	1.217
1e-6	0.500	0.696
1e-5	0.501	0.696
1e-4	0.501	0.694
1e-3	0.503	0.689
0.01	0.509	0.682
0.1	0.537	0.663
0.2	0.570	0.648
0.3	0.601	0.635
0.4	0.630	0.629
0.5	0.655	0.628
0.55	0.667	0.628
0.6	0.677	0.628
0.7	0.698	0.631
0.8	0.716	0.634
0.9	0.733	0.638
1.0	0.750	0.643

Generic *Supervised Machine Learning* workflow

**Goal : To produce a prediction machine
with small **Error(new data)****

1. Visualize data. Exploratory Data Analysis.
2. Randomly split available data into **training**, **validation**, and **testing** subsets.
3. Preprocess training + validation data. (Standardize, E&I, transform, etc.)
4. Choose model. Train model on training set.
 - **Hyperparameter tuning** if applicable
 - Evaluate/compare models[†] based on Error(validation set).
 - **k-fold cross validation** if deemed appropriate
 - Retrain model on training + validation data, using chosen hyperparameter values \leadsto final “trained machine”
5. Evaluate final trained machine based on **Error(testing set)**.

[†]. more precisely, the constituent data-generation mechanisms in the chosen model



Merci !!

Kenneth Chu

`kenneth.chu@canada.ca`

Enquêtes spéciales, transport, technologie et assurance de la qualité, DMEE
Special Surveys, Transportation, Technology and Quality Assurance, BSMD

Discussion

- Possible ML techniques to showcase in demo pipelines :
 - Classification : support vector machines, decision trees, random forests, nearest neighbour, neural networks
 - Regression : support vector machines, random forests, regularized regression
 - Unsupervised learning : clustering, PCA, dimension reduction, etc.
- Contents of demo pipelines :
 - Master program (philosophy : one “click” makes all)
 - README file : how to run the master program, refer user to blog post on Confluence page for associated blog.
 - Documentation in code
 - Optional : PDF version of associated blog post to explain the ML technique of the demo pipeline
- Outreach activities
 - Publish blog posts on Confluence page about demo pipelines.
 - Share demo pipelines as ZIP files through the associated blog posts.
 - Give branch-level seminars – probably a series of two back-to-back seminars
 - Optional : use **Jupyter** Notebook to give a live demo during seminars
 - Optional : share demo pipelines through StatCan’s (Team Foundation Service) Git repository on Network A.
- Proposed timeframe (subject to adjustment) :
 - Demo pipeline (choose ML technique, learn the technique, choose language, build demo pipeline) :
 - Blog posts :
 - Seminars : **Early February, 2018**
- Keywords :
 - bagging, boosting, ensemble methods, deep learning
 - sensitivity, specificity, precision, recall, confusion matrix, ROC Curve