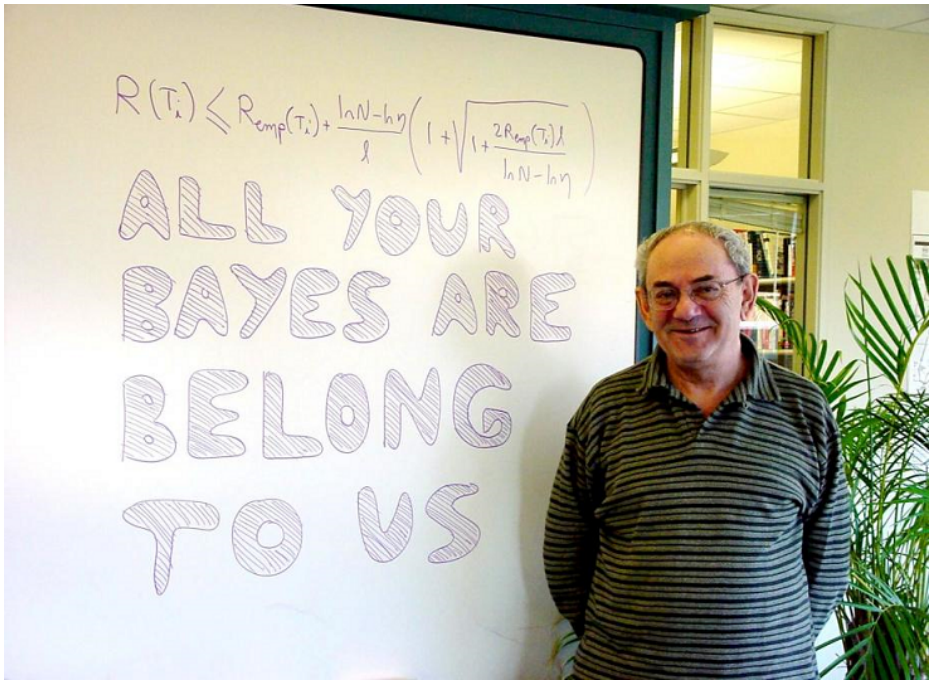
STAT3612 Lecture 9

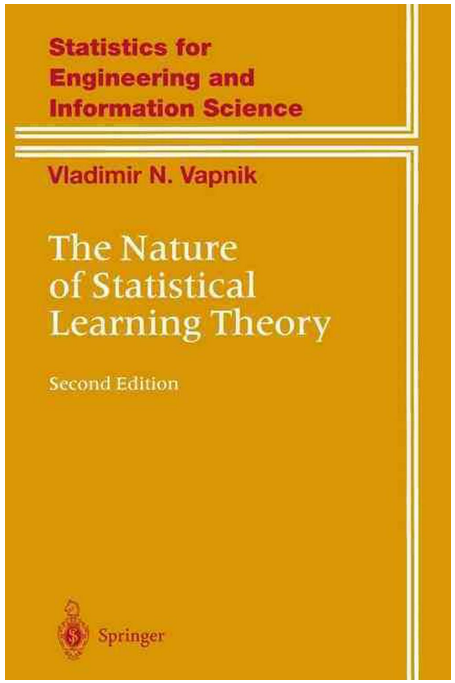
Support Vector Machines and AutoML

Dr. Aijun Zhang

3 November 2020

Department of 統計及精算學系
Statistics & Actuarial Science



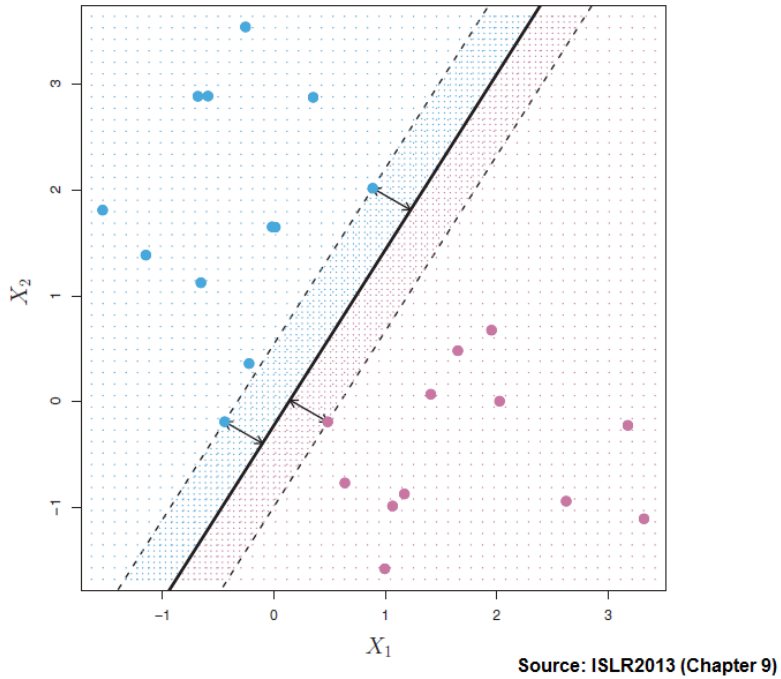


- **Vladimir Vapnik** (Wikipedia), co-developed the Vapnik–Chervonenkis theory of statistical learning, and the support vector machines.
- Authored the famous "Nature of Statistical Learning Theory (1995)"

Table of Contents

Maximal Margin Classifier

Key idea: to find the optimal separating hyperplane with maximal margin



Source: ISLR2013 (Chapter 9)

For two-class classification problem with $y \in \{-1, +1\}$:

$$\max_{\beta_0, \boldsymbol{\beta}} M$$

$$\text{s.t.} \quad y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M, \; i = 1, \ldots, n$$

Separable vs. Non-Separable Cases

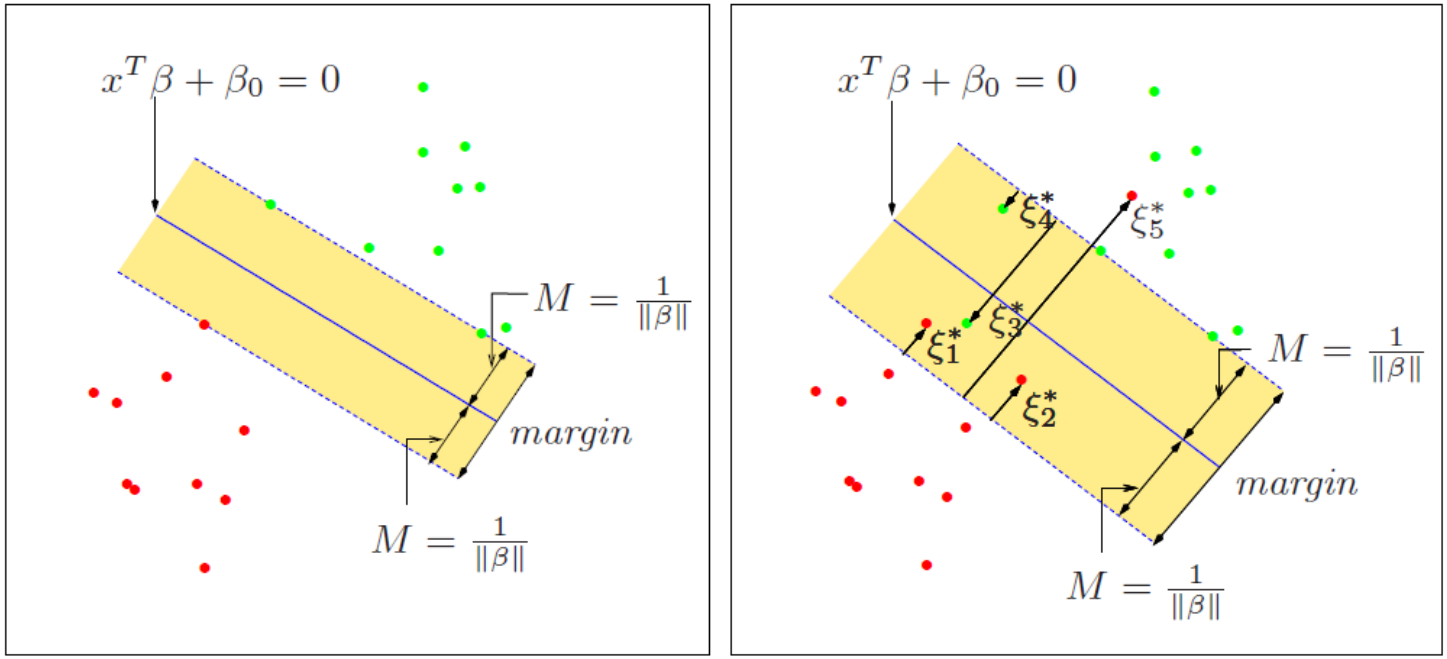


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Source: HTF2009 (Chapter 12)

Support Vector Classifier

Relaxed maximal margin problem: (Margin $M = 1/\|\boldsymbol{\beta}\|$ by geometry)

$$\begin{aligned} &\max_{\beta_0, \boldsymbol{\beta}, \xi_i} M \\ \text{s.t.} \quad &y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M(1 - \xi_i), \; i = 1, \ldots, n \\ &\xi_i \geq 0, \; \sum_{i=1}^{n} \xi_i \leq C \end{aligned}$$

- Misclassifications occur when $\xi_i > 1$;
- It can be solved by the method of Lagrange multiplier:

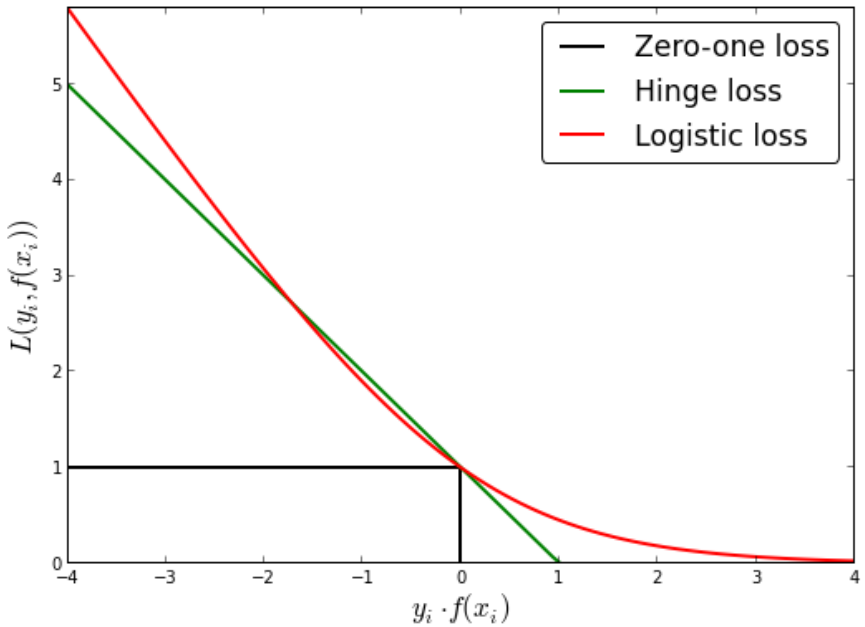
$$\begin{aligned} &\min_{\beta_0, \boldsymbol{\beta}, \xi_i} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad &y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \\ &\xi_i \geq 0, \; i = 1, \ldots, n \end{aligned}$$

Support Vector Classifier

Furthermore, it can be reformulated as minimizing ℓ_2-penalized hinge loss:

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \left[1 - y_i(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}) \right]_+ + \lambda \|\boldsymbol{\beta}\|_{\ell_2}^2$$

The hinge loss $L_H(y, f(\mathbf{x})) = [1 - y f(\mathbf{x})]_+$ is a piecewise linear function:



Support Vector Classifier

The previous quadratic problem can be solved by convex optimization. The optimal solution[1] is given by

$$\hat{\boldsymbol{\beta}} = \sum_{i \in \mathcal{S}} \hat{\alpha}_i y_i \mathbf{x}_i$$

where $\mathcal{S}$ denotes the set of support vectors. The decision function becoim

$$\hat{G}(x) = \text{sign}[\hat{f}(\mathbf{x})], \text{ where } \hat{f}(\mathbf{x}) = \sum_{i \in \mathcal{S}} \hat{\alpha}_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + \hat{\beta}_0$$

Note that the decision rule is a linear function in $\boldsymbol{x}$.

[1] You may refer to Chapter 12 of the ESL book for the derivation.

Table of Contents

Support Vector Machines

- Use of the kernel mapping in the place of pairwise inner product:

$$\begin{aligned} d\text{th-Degree polynomial: } & K(x, x') = (1 + \langle x, x' \rangle)^d, \\ \text{Radial basis: } & K(x, x') = \exp(-\gamma \|x - x'\|^2), \\ \text{Neural network: } & K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2). \end{aligned}$$

- Then the SVM decision becomes

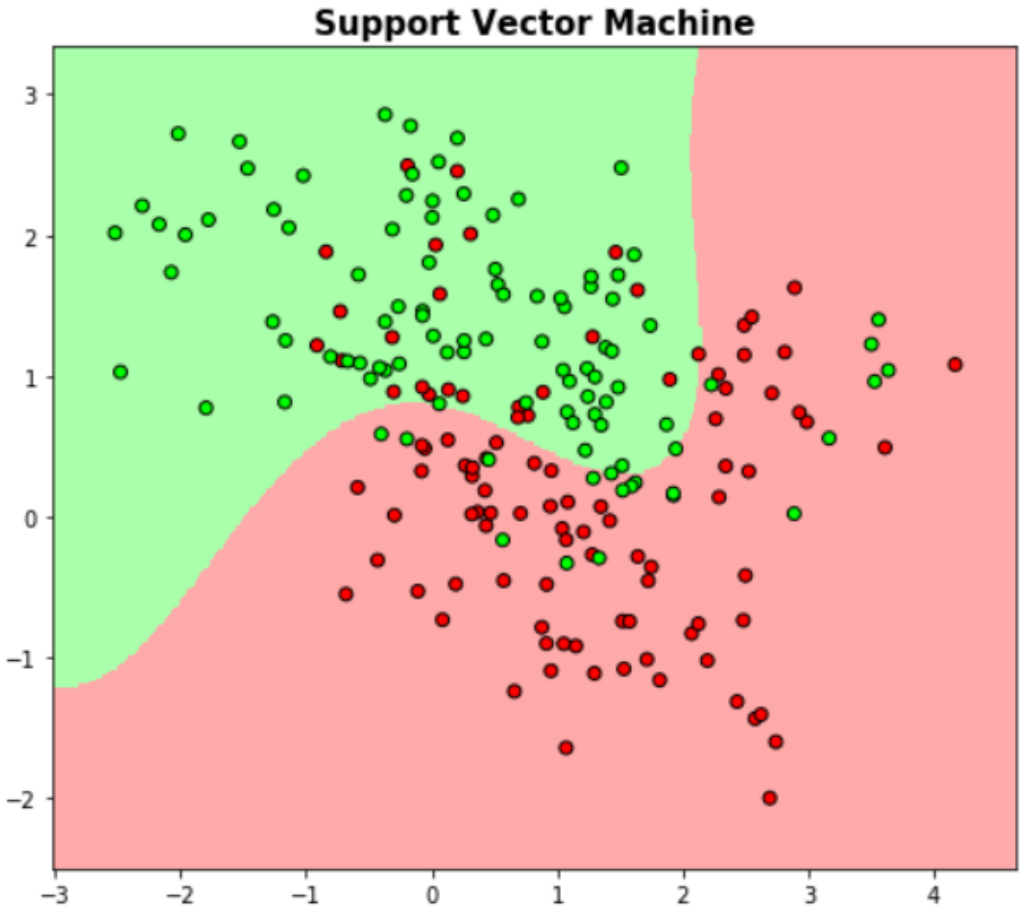
$$\hat{f}(\mathbf{x}) = \sum_{i \in \mathcal{S}} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + \hat{\beta}_0$$

- It is linear in kernel reproduced bases, but non-linear in original features.

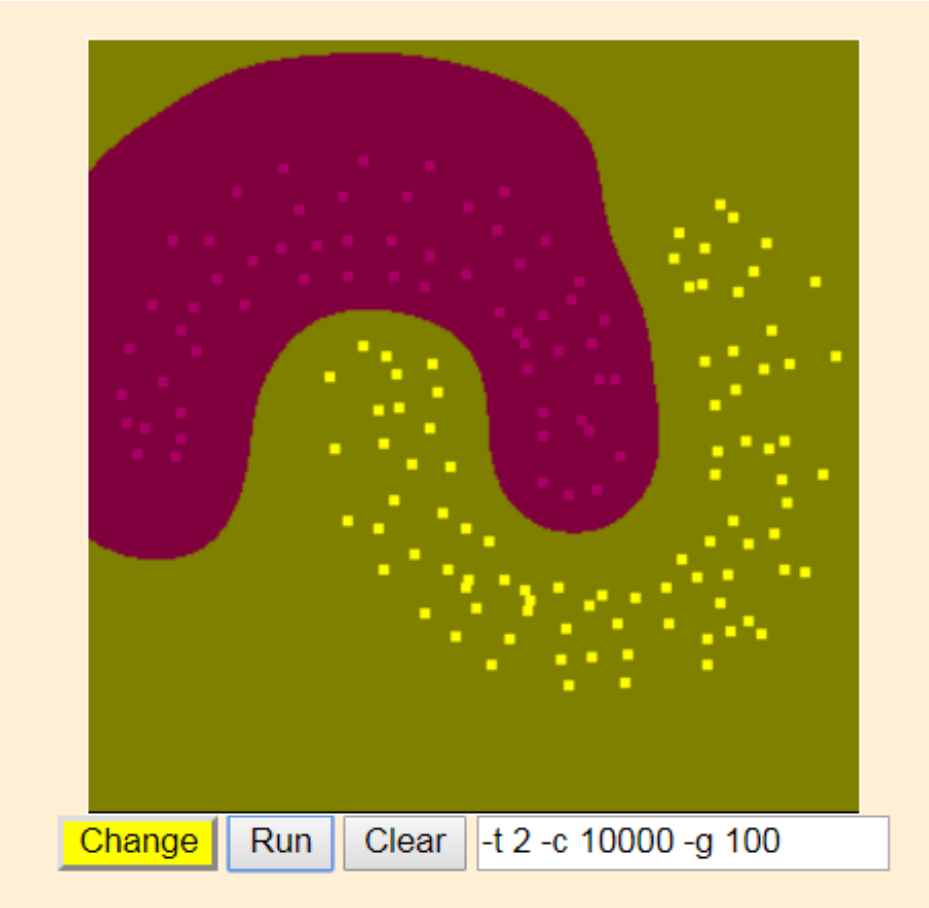
SVM by Scikit-Learn

```
from sklearn import svm

clf = svm.SVC(kernel = 'rbf', gamma=0.3)
```



Play with LibSVM Applet



URL: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

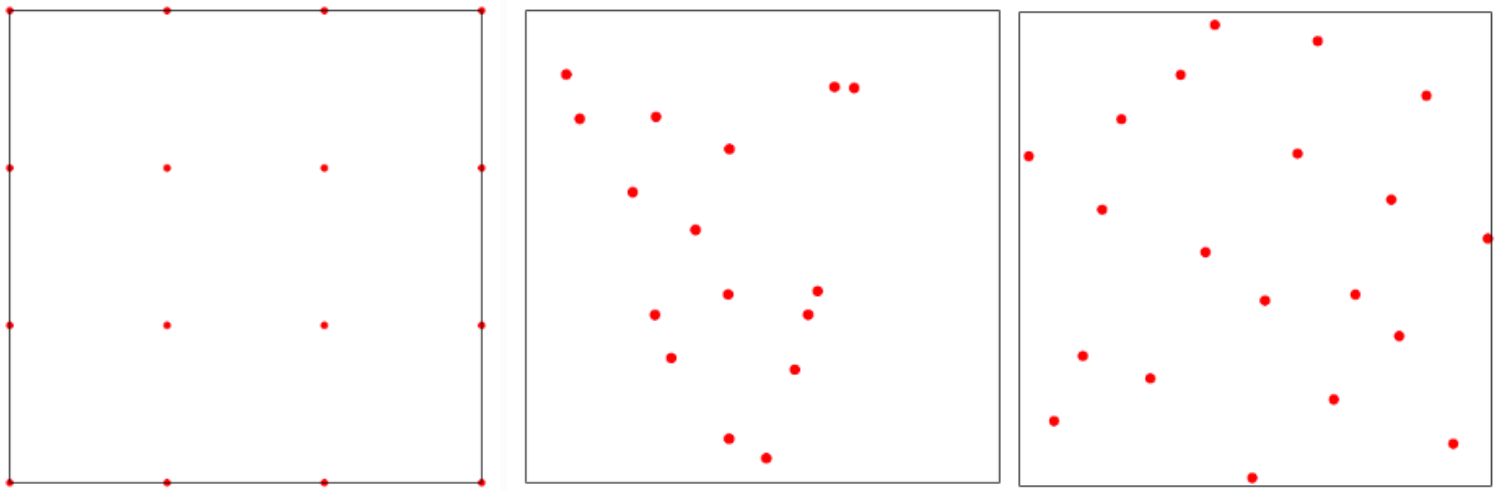
Table of Contents

Hyperparameter Optimization

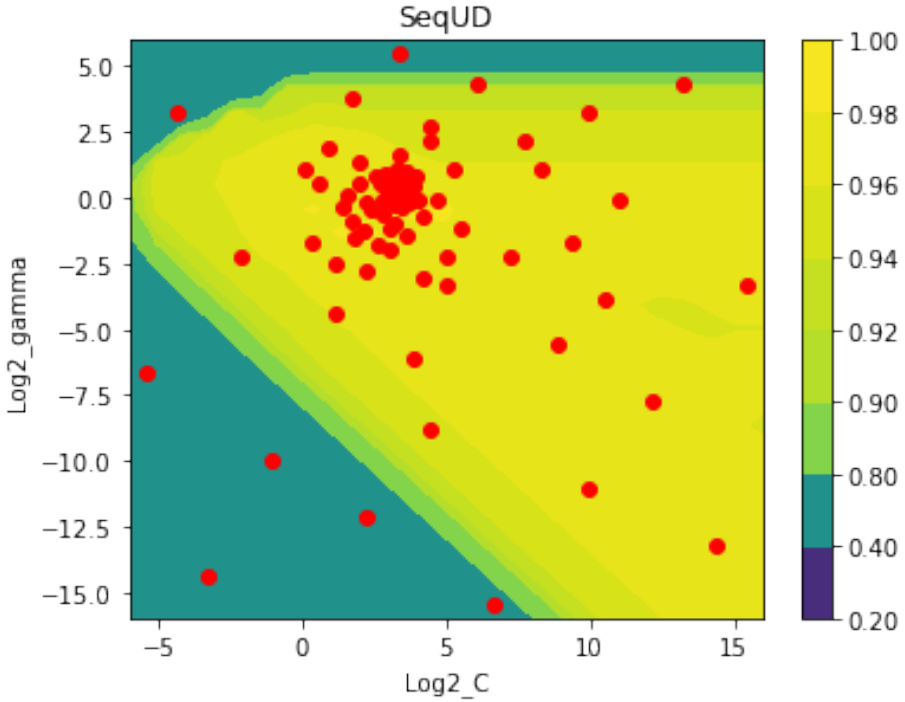
- SVMs with the RBF kernel is a popular choice for complex prediction problem. It involves two hyperparameters:

$$C \text{ (cost/regularization)} \quad \& \quad \gamma \text{ (kernel width)}$$

- **Hyperparameter optimization** or **parameter tuning** is to determine the best choices of hyperparameters.

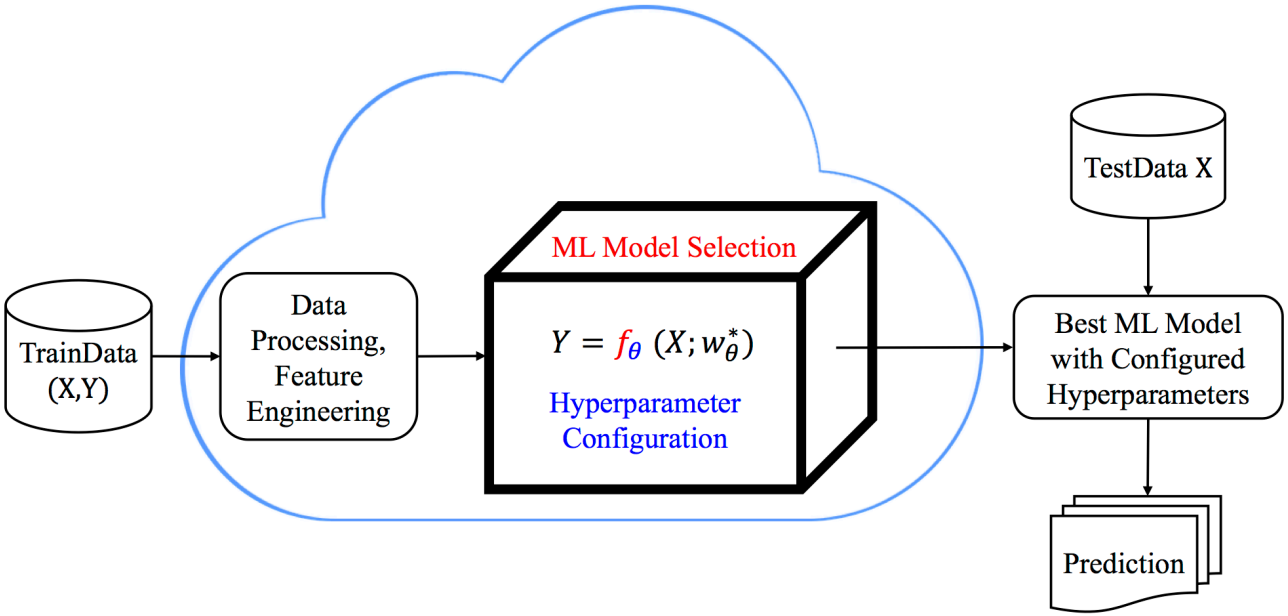


HyperOpt by SeqUD

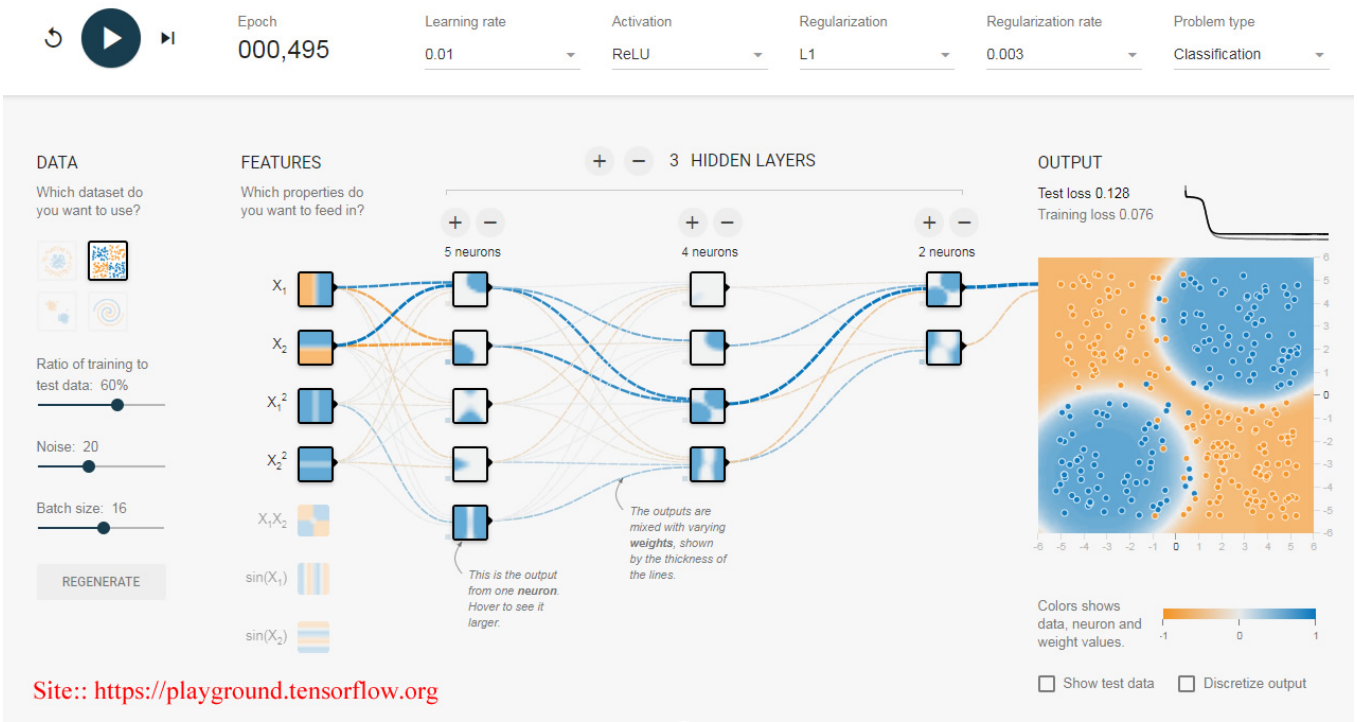


- Python **SeqUD** package: https://github.com/ZebinYang/SeqUD
- See the documentation at https://zebinyang.github.io/SeqUD/
- Reference: Yang and Zhang (2020) at https://arxiv.org/abs/2009.03586

Automated Machine Learning (AutoML)



- Automated model selection and hyperparameter optimization ...
- It also targets progressive automation of data preprocessing, feature extraction/transformation, post-processing and interpretation.
- It is to make Machine Learning easy to use by non-ML-experts.



Site:: https://playground.tensorflow.org

Hyperparameter Optimization: Existing Methods

- **Grid search:** exhaustive search over grid combinations (most popular)
- **Random search:** random sampling (Bergstra and Bengio, 2012)
- **Bayesian optimization:** sequentially sampling one-point-at-a-time through maximizing the expected improvement (Jones et al., 1998)
  - **GP-EI:** surface modeled by Gaussian process (Snoek et al., 2012)
  - **SMAC:** surface modeled by random forest (Hutter et al., 2011)
  - **TPE:** Tree-structured Parzen Estimator (Bergstra et al., 2011)
- **Genetic algorithm:** Goldberg & Holland (Machine Learning 1988)
- **Reinforcement learning:** DNN architecture search (Zoph and Le, 2016)

Grid Search vs. Random Search

BERGSTRA AND BENGIO (2012) RANDOM SEARCH FOR HYPER-PARAMETER OPTIMIZATION

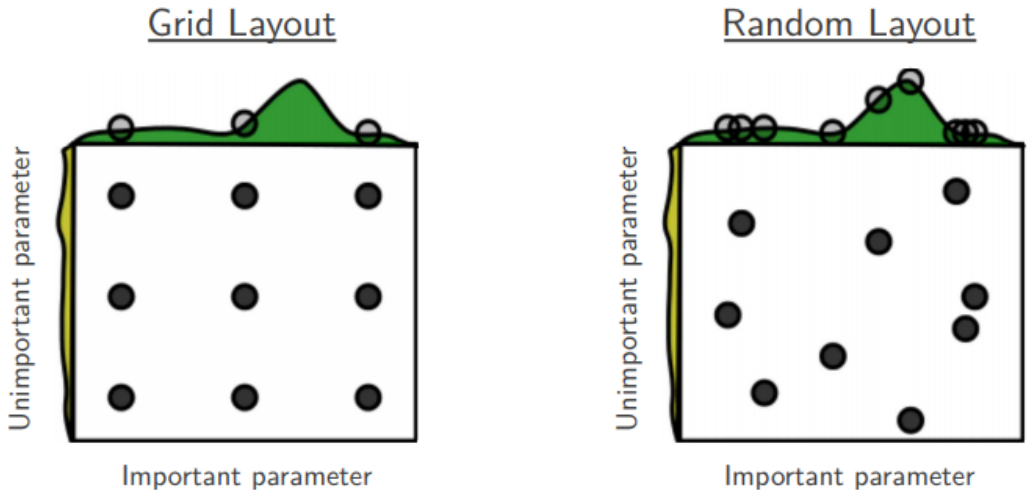
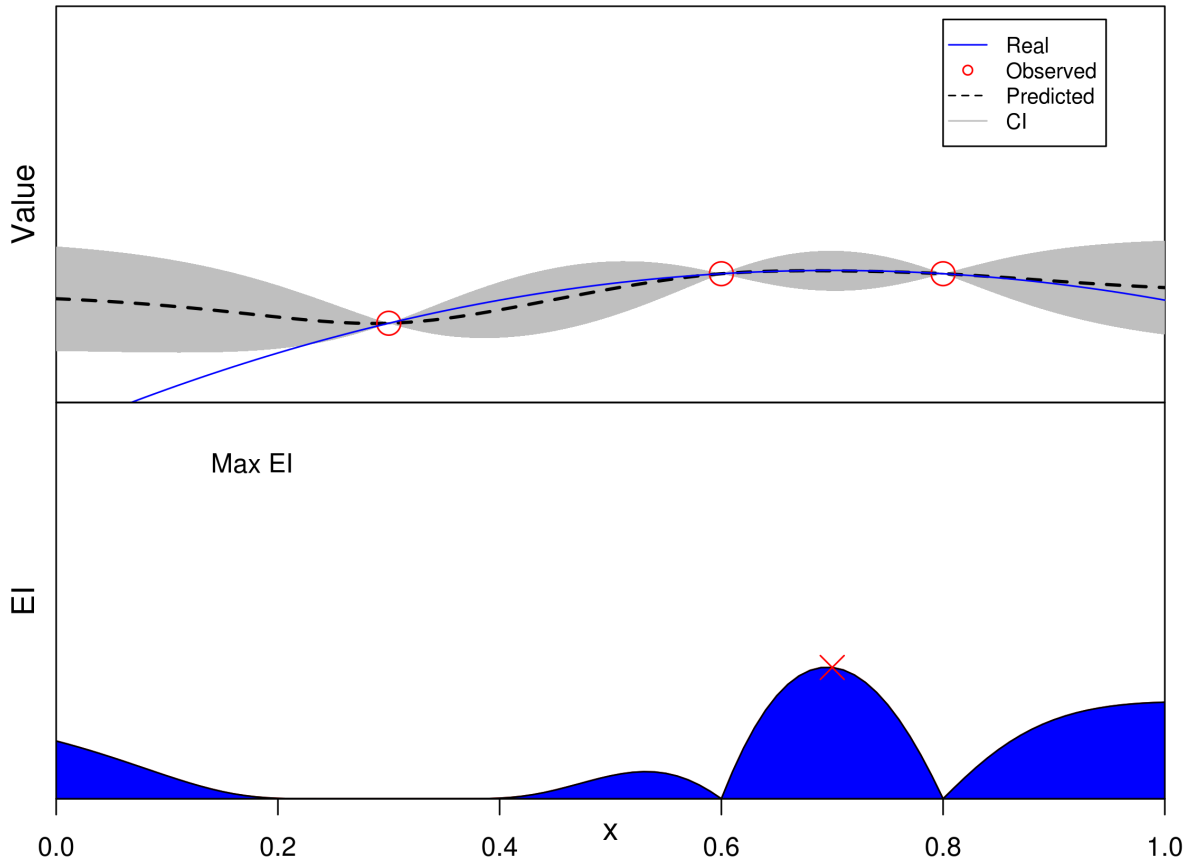


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Bayesian Optimization

Trials are sequentially sampled one-point-at-a-time through maximizing the expected improvement (EI). Here is an univariate example for illustration.



Bayesian Optimization

Algorithm 1: Bayesian optimization

1: for $n = 1, 2, \ldots,$ do
2: select new $\mathbf{x}_{n+1}$ by optimizing acquisition function α

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$$

3: query objective function to obtain y_{n+1}
4: augment data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
5: update statistical model
6: end for

E.g. the acquisition function used by **GP-EI** (Snoek, et al., 2012):

$$\alpha_{\text{EI}}(\boldsymbol{x}) = \int_{y^*}^{\infty} (y - y^*) p_{\text{GP}}(y|\boldsymbol{x}) dy = \sigma(\boldsymbol{x}) \Big[z^*(\boldsymbol{x}) \Phi(z^*(\boldsymbol{x})) + \phi(z^*(\boldsymbol{x})) \Big]$$

where y^* is the observed maximum, $(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x}))$ are the GP-predicted (posterior) mean and variance, and $z^*(\boldsymbol{x}) = (\mu(\boldsymbol{x}) - y^*)/\sigma(\boldsymbol{x})$.

Thank You!

Q&A or Email ajzhang@umich.edu