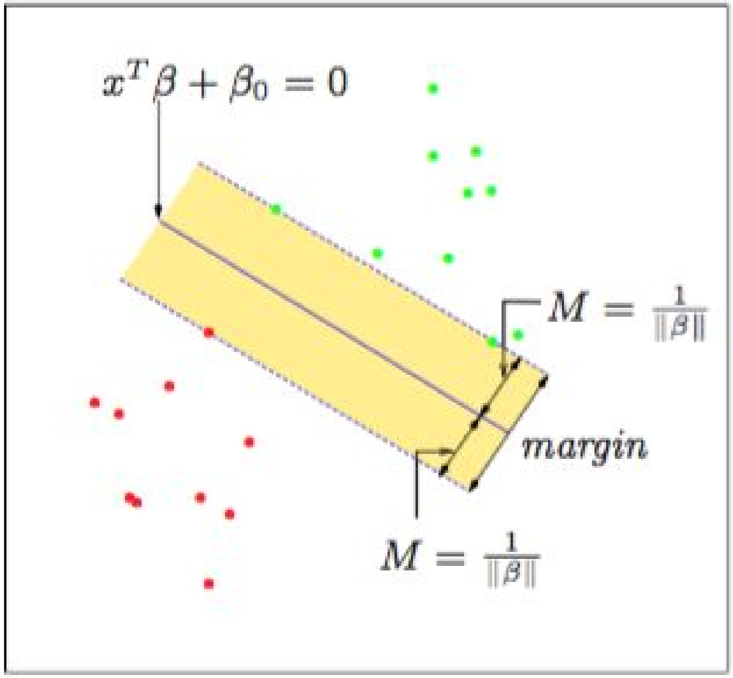
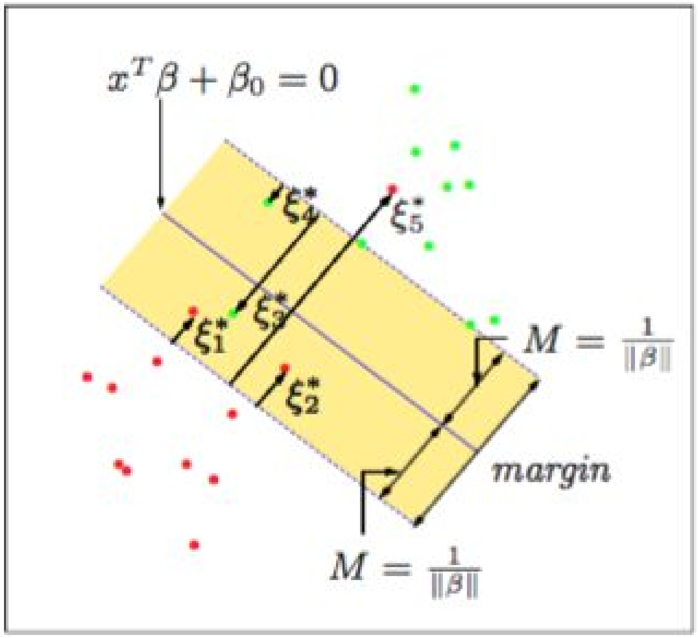
Support Vector Machines



$$\max_{\beta,\beta_0,\|\beta\|=1} M$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M,\ i = 1, \ldots, N,$$

$$\min_{\beta,\beta_0} \|\beta\|$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1,\ i = 1, \ldots, N,$$



$$y_i(x_i^T\beta + \beta_0) \quad \geq \quad M(1 - \xi_i),$$

$$\forall i,\ \xi_i \geq 0,\ \textstyle\sum_{i=1}^N \xi_i \leq \text{constant}.$$

$$\min \|\beta\| \quad \text{subject to} \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i, \\ \xi_i \geq 0,\ \sum \xi_i \leq \text{constant}. \end{cases}$$

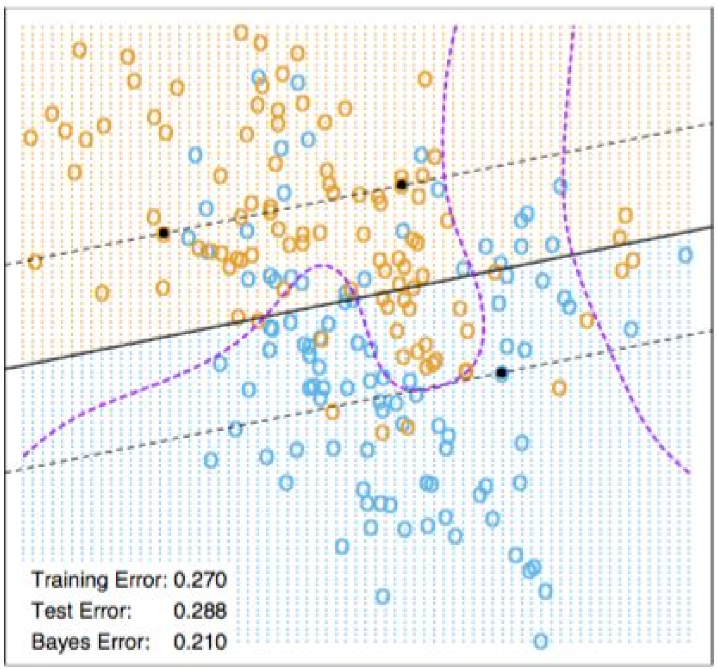
$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\ \forall i,$$

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i,$$

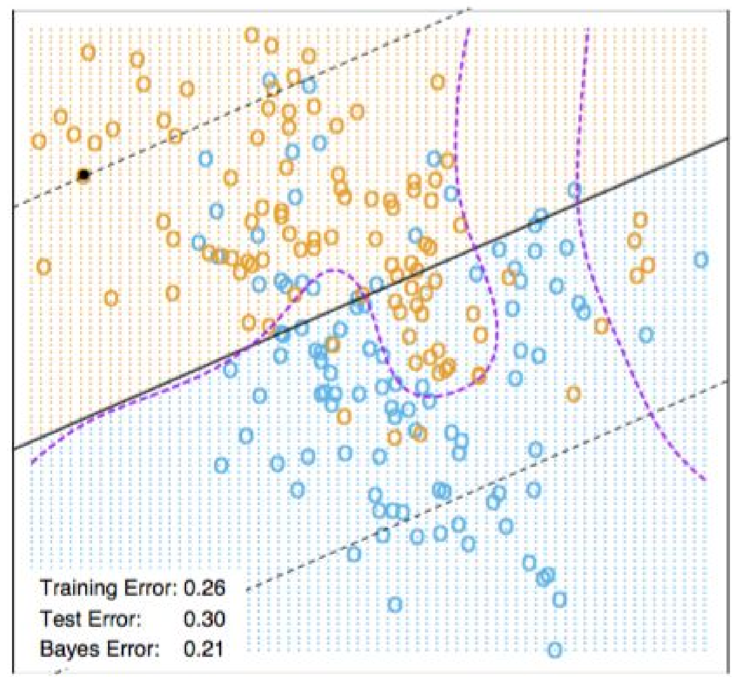
$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}y_iy_{i'}x_i^Tx_{i'},$$

Maximizing the dual (12.13) is a simpler convex quadratic programming problem than the primal (12.9), and can be solved with standard techniques

$$\hat{\beta} = \sum_{i=1}^{N}\hat{\alpha}_i y_i x_i,$$


Training Error: 0.270
Test Error: 0.288
Bayes Error: 0.210

$C = 10000$


Training Error: 0.26
Test Error: 0.30
Bayes Error: 0.21

$C = 0.01$

$$h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$$

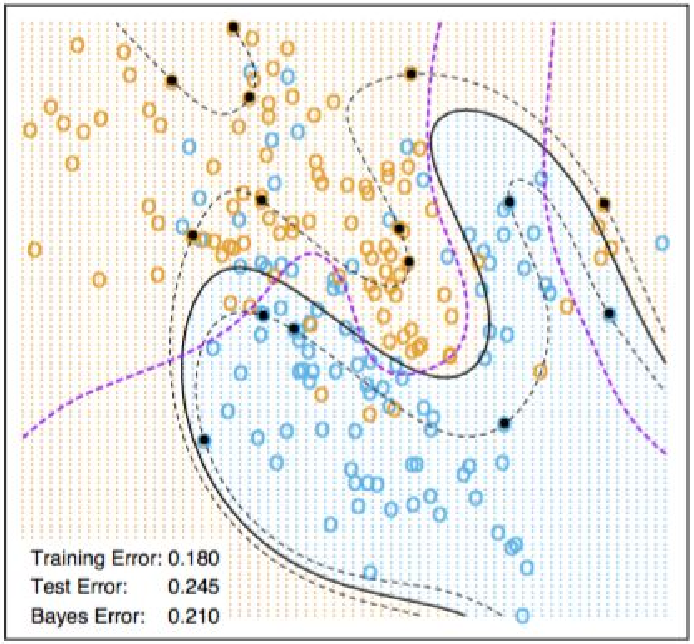
$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle.$$

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0. \end{aligned}$$

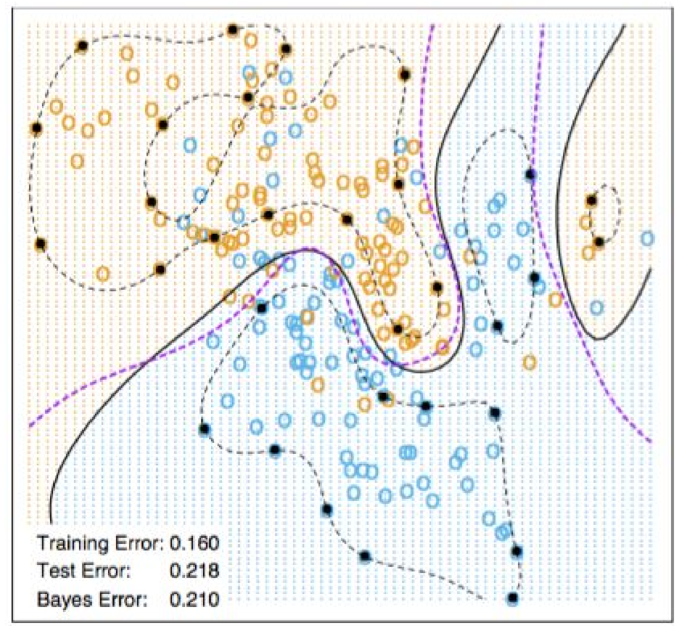
$$K(x, x') = \langle h(x), h(x') \rangle$$

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

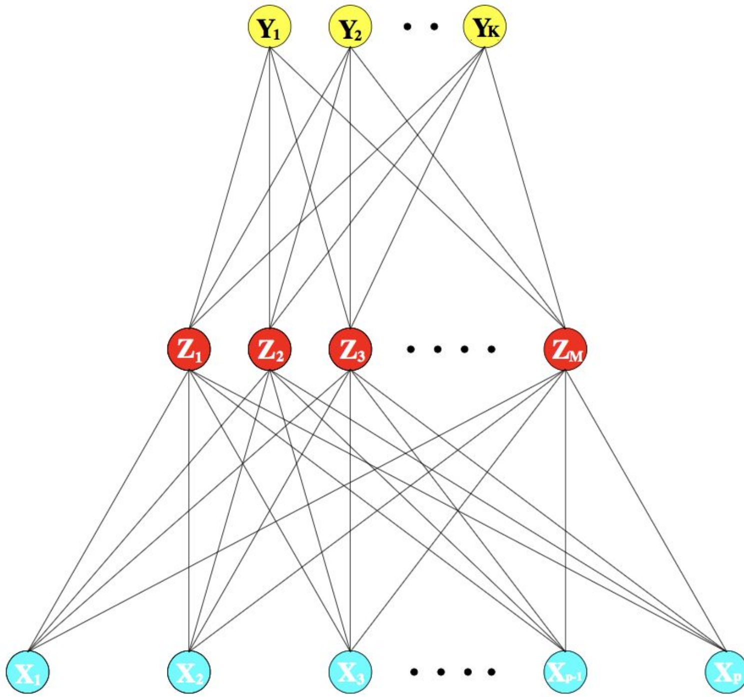
SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



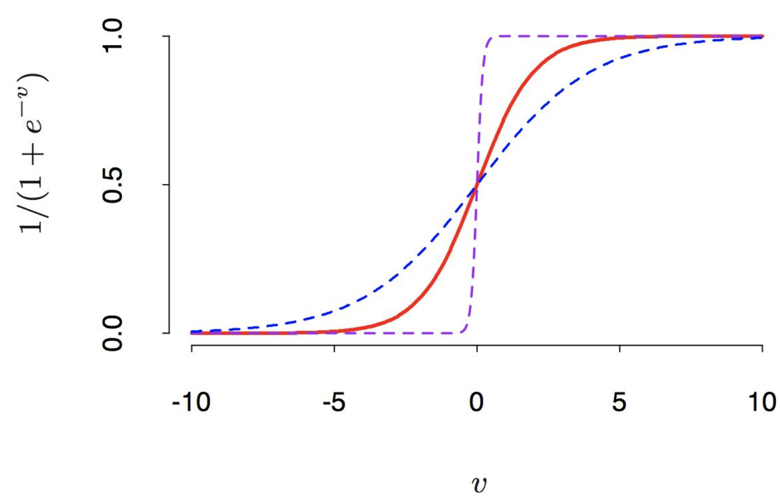
Neural Networks



$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \ldots, M,$$
$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K,$$
$$f_k(X) = g_k(T),\ k = 1, \ldots, K,$$

$$g_k(T) = \frac{e^{T_k}}{\sum_{\ell=1}^K e^{T_\ell}}.$$

FIGURE 11.2. *Schematic of a single hidden layer, feed-forward neural network.*



$$\begin{aligned} \{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\} &\quad M(p+1) \text{ weights}, \\ \{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\} &\quad K(M+1) \text{ weights}. \end{aligned}$$

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \qquad R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i),$$

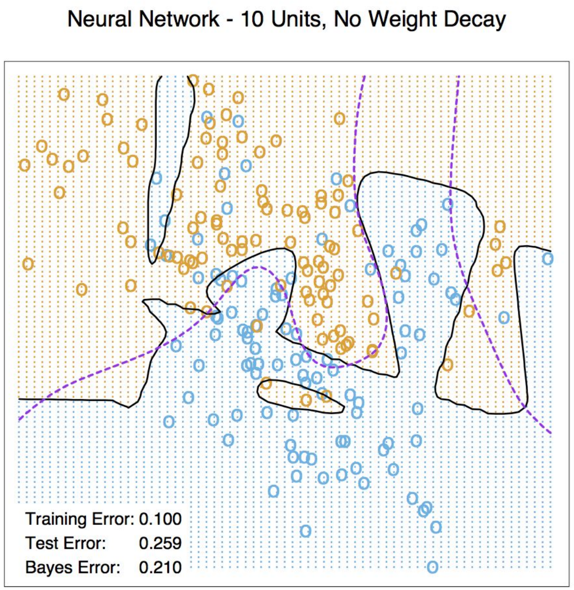
$$\begin{aligned} \frac{\partial R_i}{\partial \beta_{km}} &= -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{m\ell}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{i\ell}. \end{aligned}$$

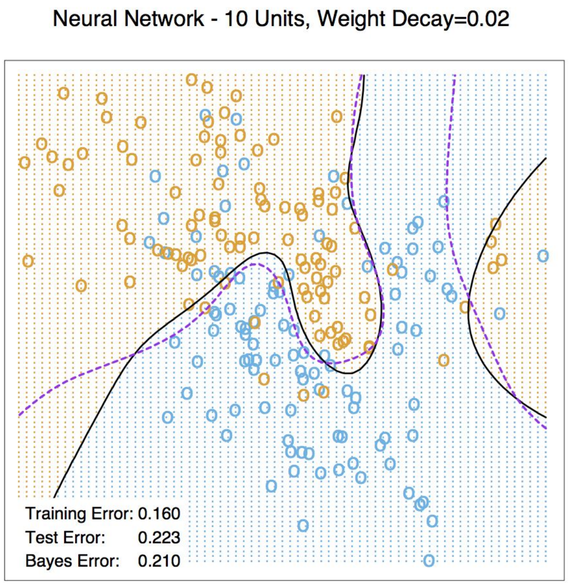
$$\begin{aligned} \beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}, \\ \alpha_{m\ell}^{(r+1)} &= \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}}, \end{aligned}$$

Neural Networks: Tricks of the Trade

initial weights

regularization (weight decay)





Neural Networks: Tricks of the Trade

scaling of inputs

online/mini-batch

Momentum

$$\Delta w(t+1) = \eta \frac{\partial E_{t+1}}{\partial w} + \mu \Delta w(t)$$

Neural Networks: Tricks of the Trade

adaptive learning rate

early stopping

dropout

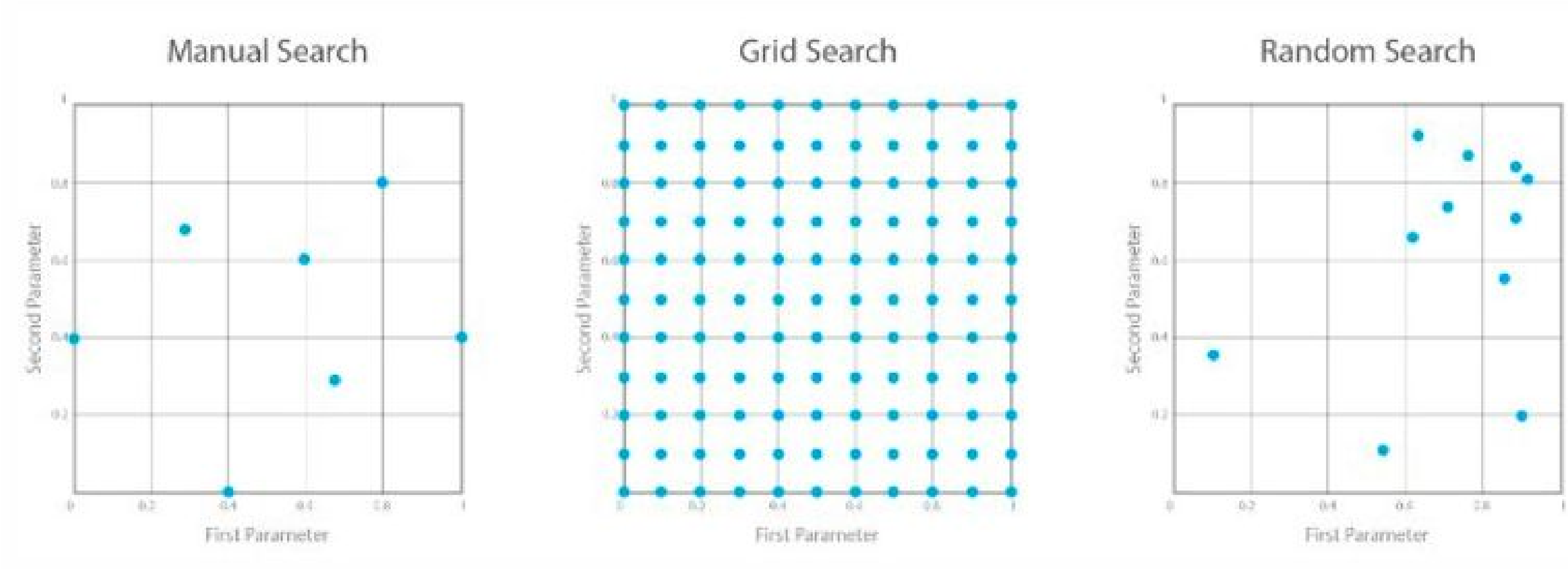
sigmoid / tanh / ReLU

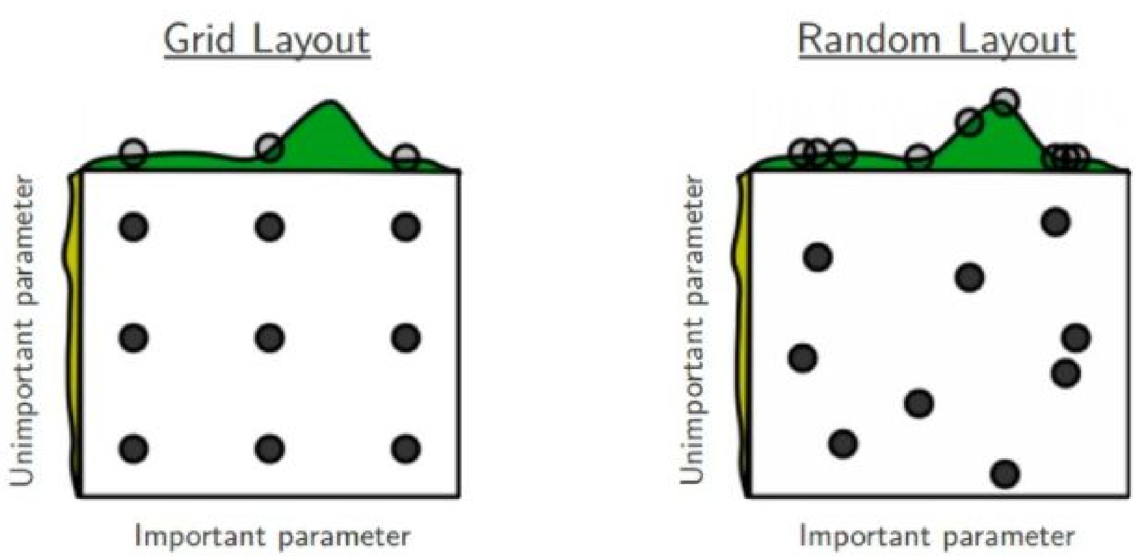
Deep Learning

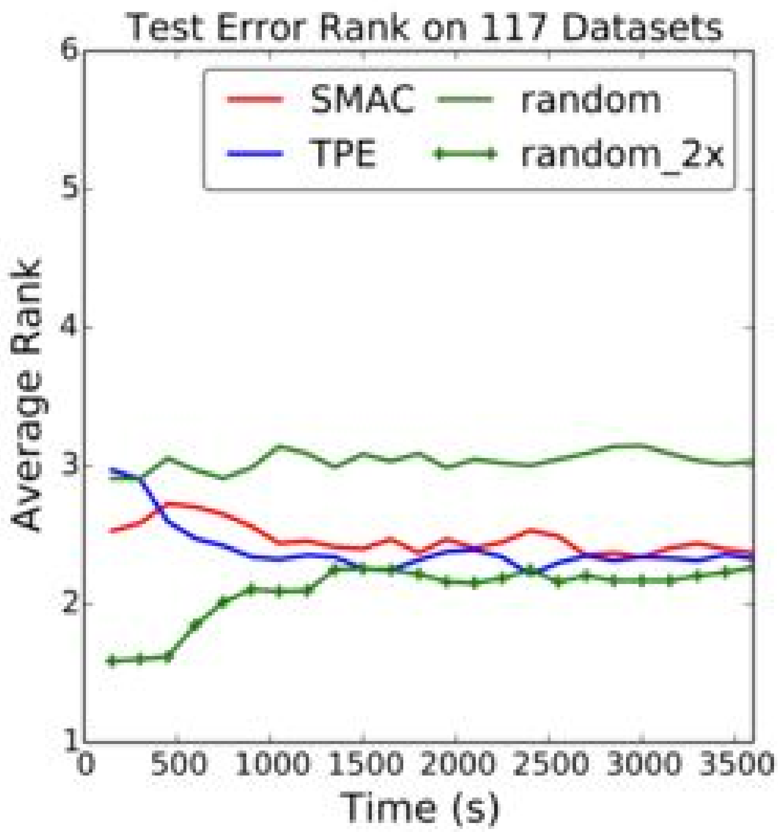
Deep Learning

CNN, RNN next week
(already done FC)

Hyperparameter Tuning


Manual Search
Grid Search
Random Search
Second Parameter
First Parameter


Grid Layout
Random Layout
Unimportant parameter
Important parameter



- Gaussian Processes (GP)
- Tree of Parzen Estimators (TPE)
- Sequential Model-based Algorithm Configuration (SMAC)

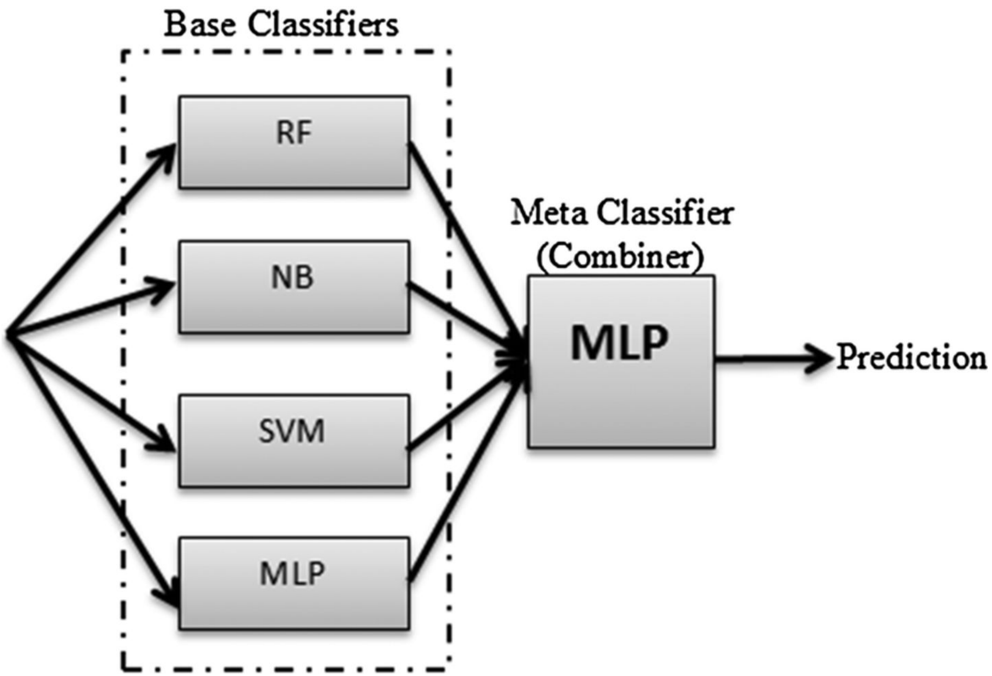
The `caret` package (short for _C_lassification _A_nd _RE_gression _T_raining) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

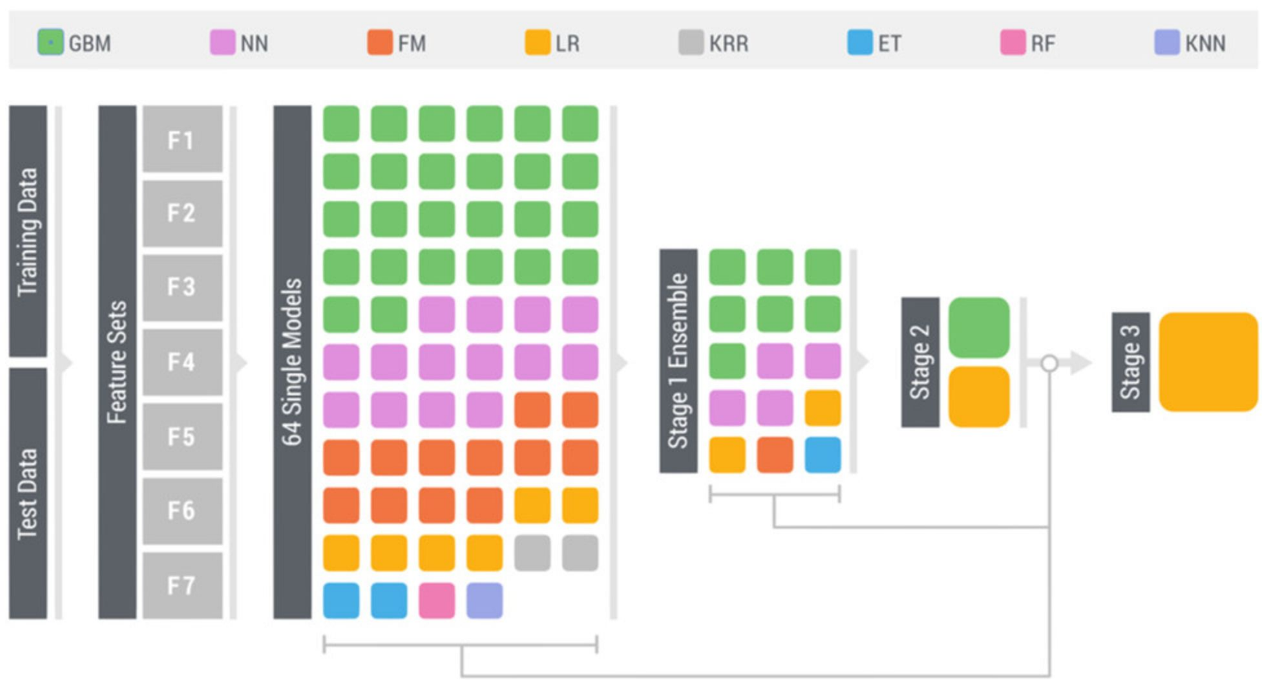
- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

also a unified API to 200+ algos (R packages):

Model	`method` Value	Type	Libraries
AdaBoost Classification Trees	adaboost	Classification	fastAdaboost
AdaBoost.M1	AdaBoost.M1	Classification	adabag, plyr
Adaptive Mixture Discriminant Analysis	amdai	Classification	adaptDA
Adaptive-Network-Based Fuzzy Inference System	ANFIS	Regression	frbs
Adjacent Categories Probability Model for Ordinal Data	vglmAdjCat	Classification	VGAM

Ensembles


Base Classifiers
RF
NB
SVM
MLP
Meta Classifier
(Combiner)
MLP
Prediction


GBM
NN
FM
LR
KRR
ET
RF
KNN
Training Data
Test Data
Feature Sets
F1
F2
F3
F4
F5
F6
F7
64 Single Models
Stage 1 Ensemble
Stage 2
Stage 3

Stacked Ensemble

$$n \left\{ \overbrace{\begin{bmatrix} \quad & X & \quad \end{bmatrix}}^{\mathrm{m}} \begin{bmatrix} y \end{bmatrix} \right.$$

"Level-zero" data

$$n \left\{ \begin{bmatrix} p_1 \end{bmatrix} \cdots \begin{bmatrix} p_L \end{bmatrix} \begin{bmatrix} y \end{bmatrix} \right. \rightarrow n \left\{ \overbrace{\begin{bmatrix} \quad & Z & \quad \end{bmatrix}}^{\mathrm{L}} \begin{bmatrix} y \end{bmatrix} \right.$$

"Level-one" data

ML in Practice

szilard / ml-prod