Erasmus
School of
Economics

Supervised Machine Learning
Week 4

Patrick J.F. Groenen

2020-2021

Erasmus University Rotterdam

Table of Contents

Table of Contents

Summary

Summary:

Week	Topics	Material
1	Introduction; Introduction to R; Linear methods for regression, model selection, and assessment	3.1, 3.2, 3.3, Xiong (2014)
2	Regularized regression and k-fold cross validation	3.4.1-3.4.3, 3.8.4, 7.10
3	Basis function expansions, kernels, bias-variance trade-off	5.1-5.2.1, 5.8, 7.3
4	Support vector machines	Groenen, Nalbantov, Bioch (2009); 12.1-12.3
5	Classification and regression trees, random forests, bootstrap	7.11, 9.2, 15
6	Boosting	10

Table of Contents

Individual Assignment

Requirements individual assignment:

- Write a report of at most 5 pages (12 pt font, single column, 1.5 line spacing)
- You answer a substantive research question using a combination of (at least) two techniques. List 1: regression, ridge regression, lasso, elastic net, binary SVM, and kernel ridge regression. List 2: regression trees, random forrests, bootstrap, permutation test, and boosting.
- You will have to add some coding similarly as has been done during the team assignments.
- Search your own data set from the book or another place. Describe the data briefly. Do not use a data set that you have used in this course before (on the same topic).
- The report should be in the form of a small article (Introduction with substantive research question, description of the data, methods, results, discussion and conclusions).

Individual Assignment

Requirements individual assignment (cont'd):

- Show that you understand how to use the techniques sensibly.
- Substantive conclusions are important.
- Tables and figures support your text and are in the main text. Appendices only for less important output.
- Do not literally copy the methods section from your group assignment (plagiarism): write it in your own words (similar set up is fine).
- Justify your conclusions by reporting appropriate results (possibly in tables or figures).
- Provide an appendix with code that replicates your analysis.
- The deadline will be two weeks after the final lecture.

Individual Assignment

Data resources:

- Use your own data.
- Use data that are applicable from a different course.
- UCI machine learning repository `http://archive.ics.uci.edu/ml/`, data sets, searchable (some relevant, others not).
- Also, use data of a known data set in a different way:
  - make useful selection of variables (appropriate for the technique);
  - sometimes transformation of one or more variables can be useful.
- Data from the datasets package in R.
- `https://www.kaggle.com/` has many data sets.

Table of Contents

The Binary SVM

Consider the binary classification problem:

- Two outcomes.
- Several predictor variables.
- Find a linear combination of predictor variables to predict outcomes (decision rule):
  - above a threshold predict one of the categories
  - below the threshold predict the other category

The Binary SVM

Examples of binary classification:

- Predict buyers from nonbuyers out of previous shopping behavior.
- Predict illness out of blood values.
- Predict unemployment of individuals out of background characteristics.
- Predict college degree out of genetic variables.

The Binary SVM

Some classification techniques:

Technique	No. Classes	Type
Fisher discriminant analysis	Multi-class	Maximum likelihood
Logistic regression	Binary	Maximum likelihood
Multinomial regression	Multi-class	Maximum likelihood
Classification trees	Multi-class	Optimization criterion
Neural networks	Multi class	Optimization criterion
Support vector machines	Binary	Optimization criterion
GenSVM	Multi-class	Optimization criterion

The Binary SVM

Support vector machine (SVM):

- The technique of support vector machines (SVM) is used to predict a binary grouping of objects by a (non)linear combination of predictor variables.

The Binary SVM

Vapnik (2000):

> *"The support vector machine implements the following idea: It maps the input vectors* $\mathbf{x}$ *into a high-dimensional feature space* $\mathbf{Z}$ *through some nonlinear mapping, chosen a priori. In this space, an optimal hyperplane is constructed."*

The Binary SVM

Four features of an SVM that make it work:

1. Optimal scaling on the response variable:
   Makes solution depend only on bad fitting observations
2. Robustness of the errors:
   Robustness against outliers
3. Regularization (shrinkage):
   Avoids overfitting
4. Allowing of nonlinearity of the predictors:
   Much more flexibility for allowing nonlinear prediction.

The Binary SVM

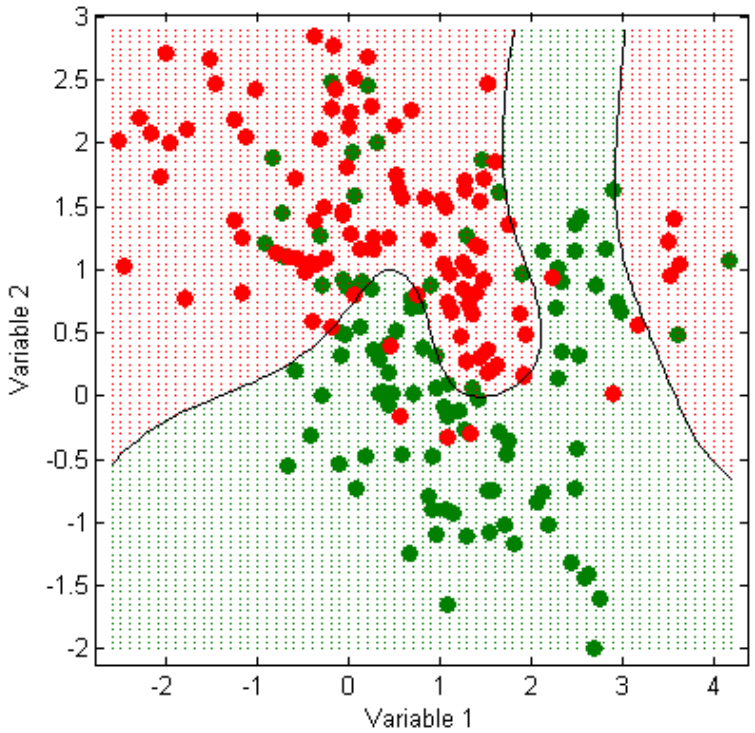
- SVMs are often explained by their solution using the dual of a quadratic program.
  - Advantage: very elegant theory and resulting in a powerful technique.
  - Disadvantage: not very insightful.
- Here, we follow an optimization approach related to multiple regression, see Groenen et al. (2008).

Groenen, P. J. F., Nalbantov, G., and Bioch, J. C. (2008). SVM-Maj: a majorization approach to linear support vector machines with different hinge errors. *Advances in Data Analysis and Classification*, *2*, 17–43.

The Binary SVM

Example: mixture data of Hastie, Tibshirani, and Friedman (2001)

- 200 observations drawn from 20 distributions from two groups.
- Border line is the true separation line according to the Bayes criterion (Bayes error rate: .21).



The Binary SVM

SVM uses the predicted value q_i as the linear combination of the predictor variables:

$$q_i = c + \mathbf{x}_i^\top \mathbf{w} = c + \sum_{j=1}^{m} x_{ij} w_j$$

with:
$\mathbf{w}$ the $m \times 1$ vector of (unknown) weights
c an (unknown) constant.

(Show intuition of an SVM.)

The Binary SVM

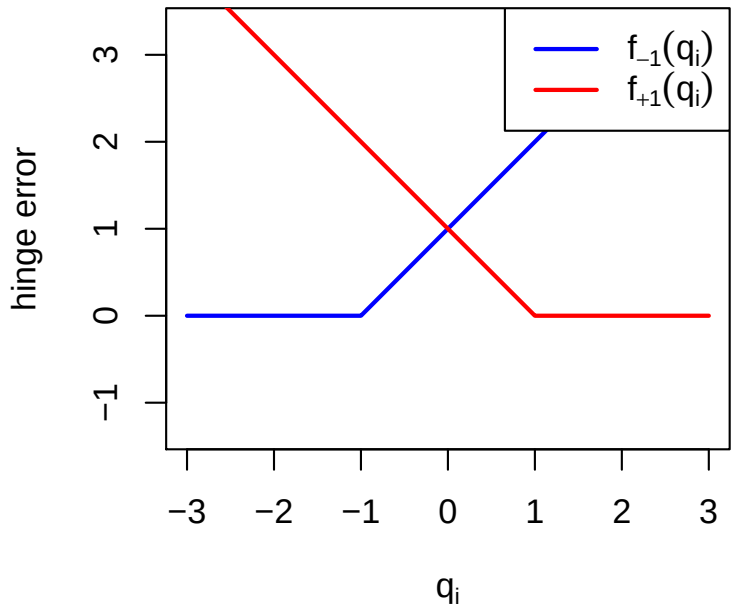
Notation:

- G_{-1}: the set of indexes i for which $y_i = -1$
- G_{+1}: the set of indexes i for which $y_i = +1$
- n: the number of observations.
- $\mathbf{X}$: the $n \times m$ matrix of predictor variables.
- $\mathbf{y}$: the $n \times 1$ vector with the groups: values are either -1 or $+1$.
- λ: a positive penalty (regularization) factor for the penalty term fixed by the user.

The Binary SVM

SVM hinge error:

$$\begin{aligned} \text{Hinge error for } i \text{ in } -1 \text{ group}: \quad f_{-1}(q_i) &= \max(0, q_i + 1) \\ \text{Hinge error for } i \text{ in } +1 \text{ group}: \quad f_{+1}(q_i) &= \max(0, 1 - q_i) \end{aligned}$$



The Binary SVM

SVM loss function:

$$L_{\text{SVM}}(c, \mathbf{w}) = \sum_{i \in G_{-1}} \max(0, q_i + 1) + \sum_{i \in G_{+1}} \max(0, 1 - q_i) + \lambda \mathbf{w}^\top \mathbf{w}$$

- $\sum_{i \in G_{-1}} \max(0, q_i + 1)$: Group -1 error
- $\sum_{i \in G_{+1}} \max(0, 1 - q_i)$: Group $+1$ error
- $\lambda \mathbf{w}^\top \mathbf{w}$: Penalty term

The Binary SVM

Support vectors are those observations $\mathbf{x}_i^\top$ that have nonzero error.

- Only the support vectors are needed to compute the solution.
- Support vectors are not known in advance, only after the solution is computed.

SVM

Standard formulation of SVM in the machine learning literature is different (at first sight):

$$L_{\text{SVMClas}}(c, \mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2\lambda} \sum_{i=1}^{n} \xi_i + \frac{1}{2} \mathbf{w}^\top \mathbf{w}$$

subject to

$$\begin{aligned} 1 - y_i q_i &\leq \xi_i \\ \xi_i &\geq 0 \\ q_i &= c + \mathbf{x}^\top \mathbf{w} \end{aligned}$$

The Binary SVM

Step 1 to get to L_{SVM}: separate groups

$$L_{\text{SVMClas}}(c, \mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2\lambda} \sum_{i \in G_{-1}} \xi_i + \frac{1}{2\lambda} \sum_{i \in G_{+1}} \xi_i + \frac{1}{2} \lambda \mathbf{w}^\top \mathbf{w}$$

subject to

$$\begin{aligned} 1 + q_i &\leq \xi_i \text{ for } i \in G_{-1} \\ 1 - q_i &\leq \xi_i \text{ for } i \in G_{+1} \\ \xi_i &\geq 0 \\ q_i &= c + \mathbf{x}^\top \mathbf{w} \end{aligned}$$

The Binary SVM

Step 2 to get to L_{SVM}: multiply by 2λ

$$
\begin{aligned}
& 2\lambda L_{\text{SVMClas}}(c, \mathbf{w}, \boldsymbol{\xi}) \\
&= \sum_{i \in G_{-1}} \xi_i + \sum_{i \in G_{+1}} \xi_i + \lambda \mathbf{w}^\top \mathbf{w} \\
&= \sum_{i \in G_{-1}} \max(0, q_i + 1) + \sum_{i \in G_{+1}} \max(0, 1 - q_i) + \lambda \mathbf{w}^\top \mathbf{w} \\
&= \boxed{\text{Group } -1 \text{ error}} \quad \boxed{\text{Group } +1 \text{ error}} \quad \boxed{\text{Penalty term}} \\
&= L_{\text{SVM}}(c, \mathbf{w})
\end{aligned}
$$

- Thus both formulations are exactly the same.

The Binary SVM: Example

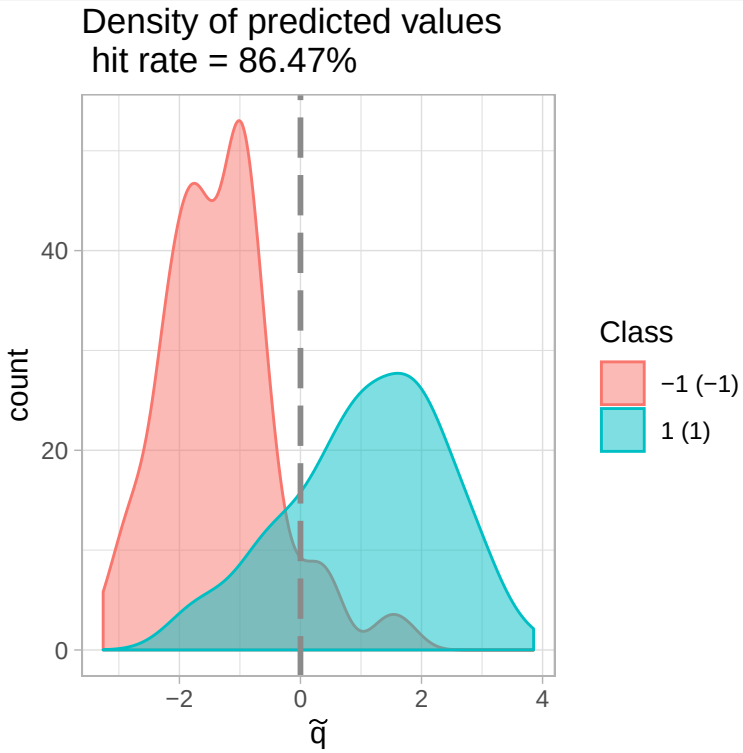
Heart data:

```
R> ## Prepare Heart data for SVM
R> data("Heart", package = "dsmle")
R> Heart <- na.omit(Heart)
R> y <- Heart$AHD
R> X <- model.matrix(AHD ~ ., data = Heart)
R> # Create a training and a test set
R> set.seed(1)                                  # Initialize random generator for repl
R> n.train <- 207                               # Use 100 training samples (too small,
R> ind <- sample(nrow(X), size = n.train)       # Generate random indexes for training
R> X.train <- X[ind,]                           # Select rows for training predictors
R> y.train <- y[ind]                            # Select training response y.train
R> X.test  <- X[-ind,]                          # Select rows for test predictors X.te
R> y.test  <- y[-ind]                           # Select test response y.test
```

The Binary SVM: Example

Heart data:

```
R> ## Do cross validation of linear SVM on heart data with SVMMaj
R> result <- svmmajcrossval(X.train, y.train, scale= "zscore",
                            return.model = TRUE)
R> plot(result$model)
```



The Binary SVM: Example

Heart data:

```
R> summary(result$model)
Call:
   svmmaj.default(X = X, y = y, weights.obs = w, scale = "zscore",
    initial.point = NULL, check.positive = FALSE, convergence = 1e-08)

Settings:
    lambda                            4.67
    hinge error                       absolute
    spline basis                      no
    type of kernel                    linear

Data:
    class labels                      -1 1
    rank of X                         17
    number of predictor variables     17
    number of objects                 207
    omitted objects                   0

Model:
    update method                     svd
    number of iterations              83
    loss value                        73.5
    number of support vectors         85

Confusion matrix:
            Predicted(yhat)
Observed (y)  -1  1 Total
       -1    105  9   114
       1      19 74    93
       Total 124 83   207

Classification Measures:

   hit rate                          0.865
   weighted hit rate                 0.865
   misclassification rate            0.135
   weighted missclassification rate  0.135

              TP        FP Precision
    -1     0.921    0.0789     0.847
    1      0.796    0.2043     0.892
```

The Binary SVM: Example

Heart data:

```
R> ## Print weights ordered by their (absolute) value
R> tt <-sort(abs(result$model$beta), decreasing = TRUE, index.return = TRUE)
R> beta <- result$model$beta[tt$ix,, drop = FALSE]
R> rownames(beta) <- colnames(result$model$Xnew)[tt$ix]
R> round(beta, digits = 3)
                         [,1]
X.Ca                    0.586
X.ChestPaintypical     -0.436
X.ChestPainnonanginal -0.374
X.Sex                   0.350
X.MaxHR                -0.268
X.ExAng                 0.262
X.Thalreversable        0.226
X.ChestPainnontypical -0.226
X.Thalnormal           -0.220
X.Oldpeak               0.218
X.RestBP                0.172
X.RestECG               0.157
X.Slope                 0.136
X.Age                  -0.066
X.Fbs                  -0.049
X.Chol                  0.027
```

The Binary SVM: Example

```
R> ## Show effect of weights per variable in plot
R> plotWeights(result$model, plotdim = c(4, 4))
```

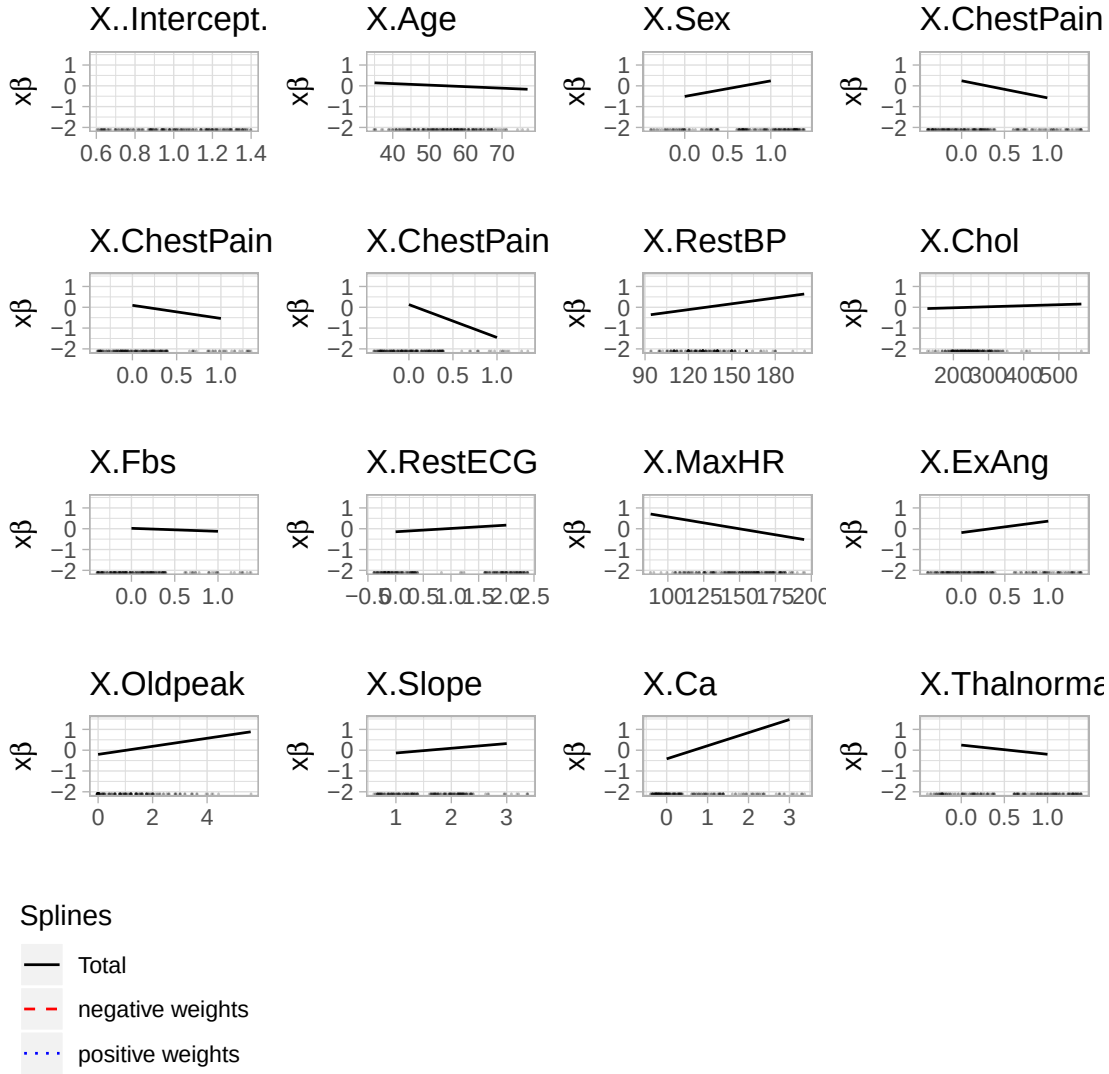


Table of Contents

SVM Diagnostics

Diagnostics in binary classification

- Emphasis on classifying observations in the right class.
- Overall goal: correct out-of-sample classification.
- Same problem as with multiple regression: danger of overfitting of training data.

SVM Diagnostics

Confusion matrix:

		Predicted class		
		-1	$+1$	Total
True class	-1	True neg. (TN)	False pos. (FP)	N
	$+1$	False neg. (FN)	True pos. (TP)	P
	Total	N*	P*	n

SVM Diagnostics

Binary classification measures (non-exhaustive):

Terms	Definition
Type I error	FP/N
Specificity	1−FP/N
Recall[1]	TP/P
Precision[2]	TP/P*
Hit rate	(TP+TN)/n
Misclassification rate	1−(TP+TN)/n
F1	$2 \cdot \text{precision} \cdot \text{recall}/(\text{precision} + \text{recall})$

[1] 1−type II error, power, sensitivity
[2] 1−false discovery proportion

SVM Diagnostics

Diagnostics in binary classification

- All measures of confusion matrix can be interesting.
- Focus can be weighted by cost weights:
  weight FN and FP according to their costs
- Example: in a diagnostic test for HIV, false negative (FN) may be weighted more than false positive (FP)

SVM Diagnostics SVM: Example

Diagnostics training sample (heart data set):

```
            Predicted(yhat)
Observed (y)  -1   1 Total
       -1    105  9   114
       1      19 74    93
       Total 124 83   207

Classification Measures:

   hit rate                           0.865
   weighted hit rate                  0.865
   misclassification rate             0.135
   weighted missclassification rate   0.135

              TP        FP Precision
    -1     0.921    0.0789     0.847
    1      0.796    0.2043     0.892
...
```

SVM Diagnostics: Example

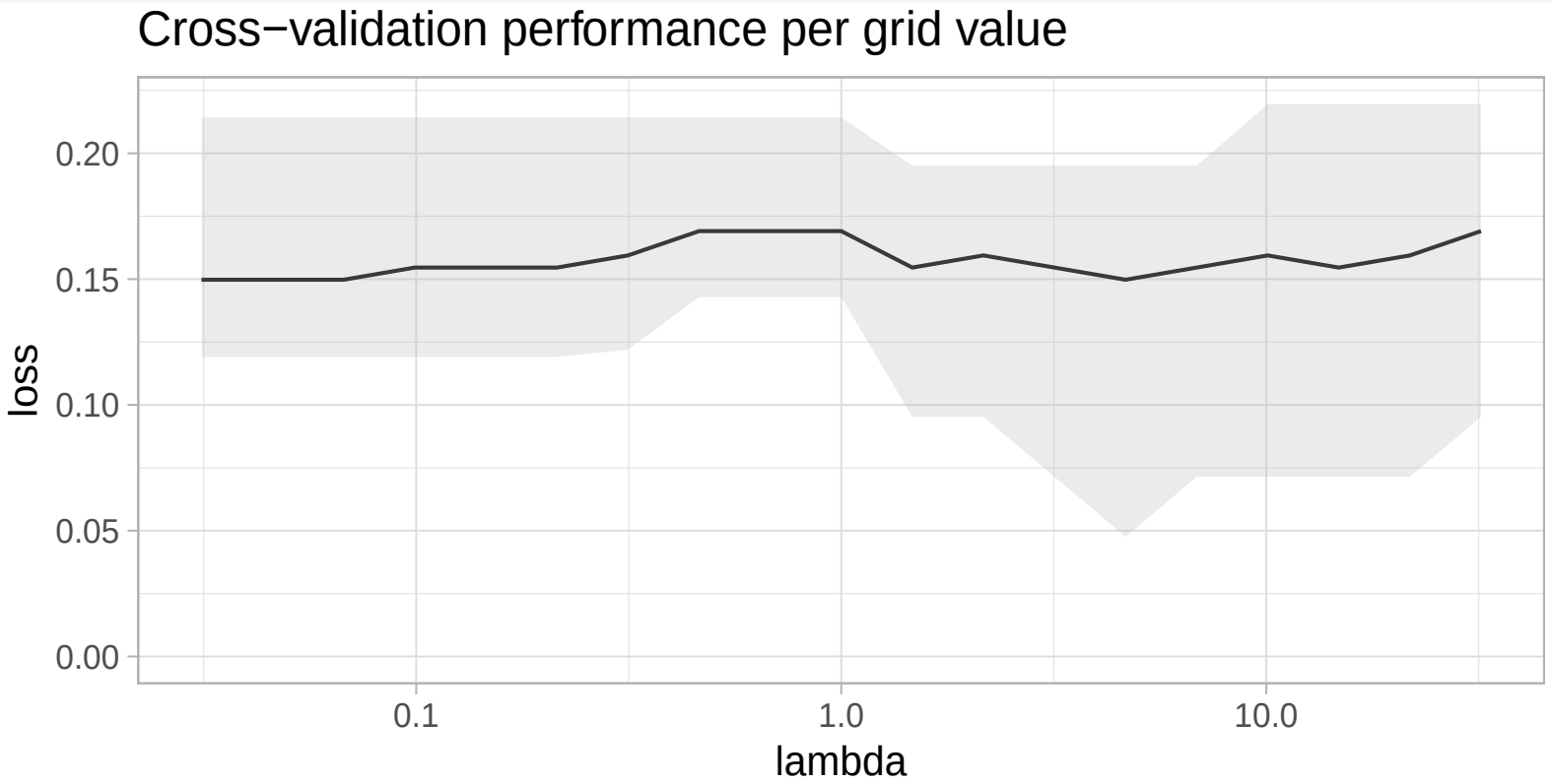
Performance of test data (heart data set):

```
R> ## Compute confusion table for test set
R> tt <- predict(result$model, X.test, y = y.test, show.plot = FALSE)
R> # Get the predicted values out of tt and make it a factor
R> y.test.hat <- factor(attr(tt, "yhat"), c(-1, 1), labels = c("No", "Yes"))
R> # Call confusionMatrix (from caret package) to get statistics
R> confusionMatrix(y.test, y.test.hat)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  44   2
       Yes 10  34

               Accuracy : 0.867
                 95% CI : (0.779, 0.929)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : 2.93e-08

                  Kappa : 0.732

 Mcnemar's Test P-Value : 0.0433

            Sensitivity : 0.815
            Specificity : 0.944
         Pos Pred Value : 0.957
         Neg Pred Value : 0.773
             Prevalence : 0.600
         Detection Rate : 0.489
   Detection Prevalence : 0.511
      Balanced Accuracy : 0.880

       'Positive' Class : No
```

SVM Diagnostics: Example

Determining λ through *K*-fold cross validation on training data:

```
R> # Make plot of K-fold cross validated missclassification error
R> plot(result)
```



SVM Diagnostics: Example

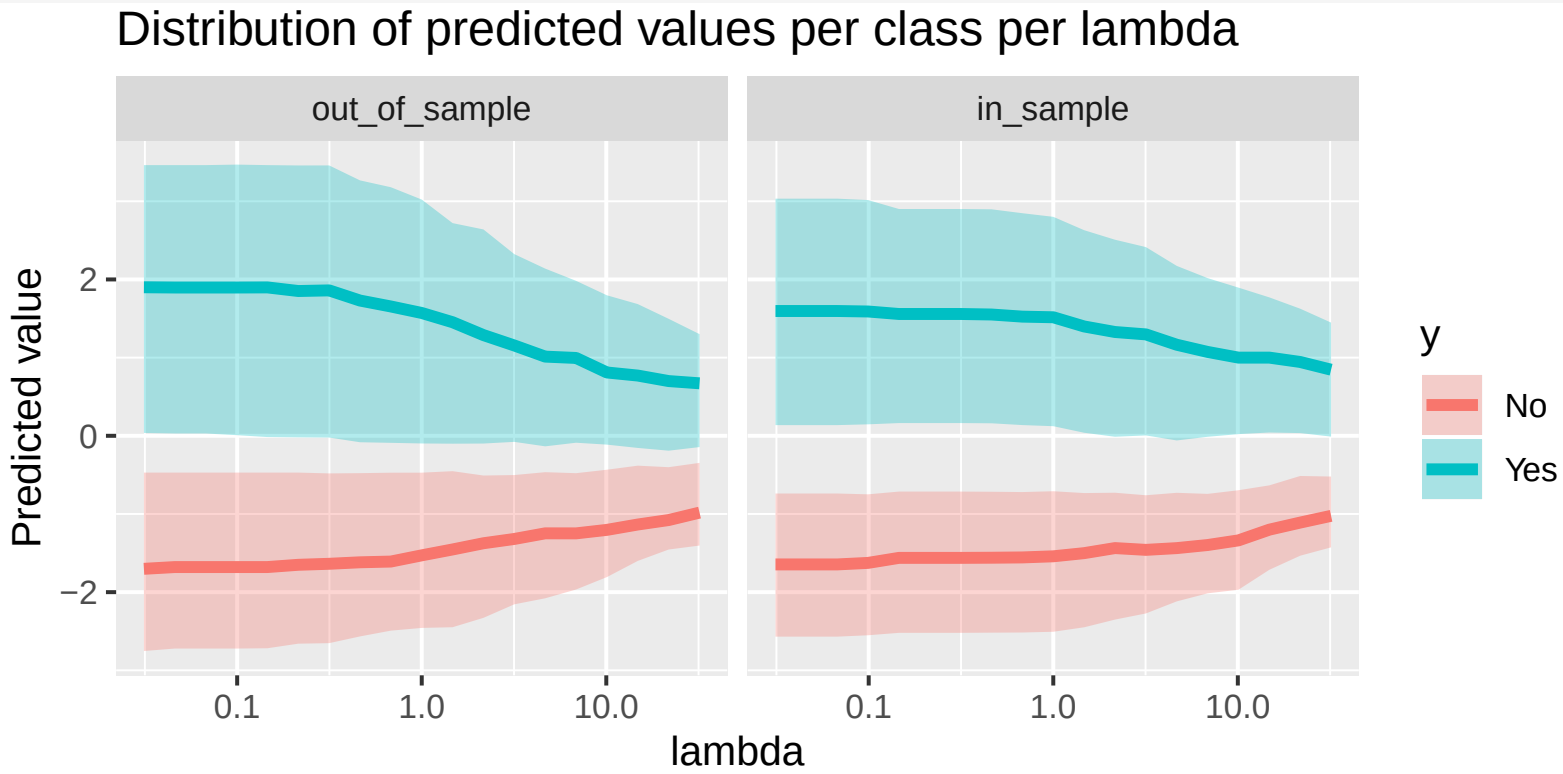
Summary of *K*-fold cross validation on training data:

```
R> # Make plot of K-fold cross validated missclassification error
R> summary(result)
Search grid:
$lambda
 [1] 32.0000 21.7726 14.8140 10.0794  6.8580  4.6661  3.1748  2.1601  1.4697
[10]  1.0000  0.6804  0.4629  0.3150  0.2143  0.1458  0.0992  0.0675  0.0459
[19]  0.0312

$loss
[1] 0.169 0.159 0.155 0.150

Optimal value ( missclassification rate =  0.15 ):
  lambda
6   4.67
Optimal model:
Model:
    update method                   svd
    attribute dimension             207 17
    degrees of freedom              17
    number of iterations            83
    loss value                      73.5
    number of support vectors       85
```

SVM Diagnostics: Example

Profile of correctly predicted objects for different λ through K-fold cross validation:

```
R> plot(result, type = "profile")
```



SVM: Diagnostics

How to do an SVM:

- Make sure the data do not have missing values.
- Standardize the predictors variables in $\mathbf{X}$ (e.g., to z-scores by using `scale = "zscore"` in `svmmaj()`).
- If possible: set a super test part of the data apart for validation.
- Apply K-fold cross validation do determine hyper parameters (such as λ) and run total model
  (`svmmajcrossval()` does both steps in one go).
- Consider confusion table of super test data as a measure for prediction quality.
- Interpret your favorite binary classification measure for performance.
- When doing a linear SVM, interpret the weights w_j.

Table of Contents

Thursday Meeting

Schedule for Thursday November 19, 2020, topic of Week 3

Team Task	Team 1	2	3	5	6
Presentation methods, results and interpretation			+		
Discussion methods,				+	
Discussion results and interpretation					+
Presentation code	+				
Discussion code		+			

- Discussions address three items:
  - what you think was good;
  - possibly address issues that were unclear to you;
  - suggestions of issues that you think could be improved.

Table of Contents

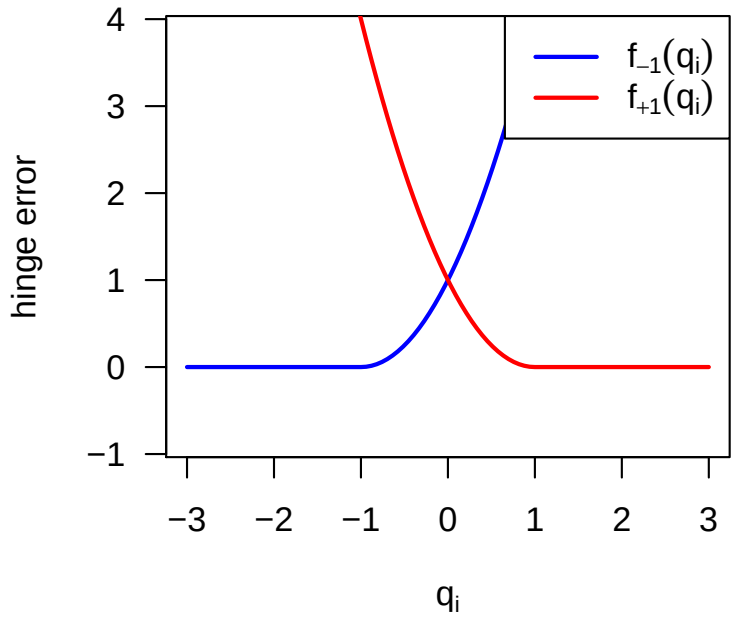
Quadratic Hinge Errors

Quadratic hinge error: $f_{-1}(q_i) = \max(0, q_i + 1)^2$

$$L_{\text{Quad-SVM}}(c, \mathbf{w})$$

$$= \sum_{i \in G_{-1}} \max(0, q_i + 1)^2 + \sum_{i \in G_{+1}} \max(0, 1 - q_i)^2 + \lambda \mathbf{w}^\top \mathbf{w}$$

$$= \boxed{\text{Group } -1 \text{ error}} \quad \boxed{\text{Group } +1 \text{ error}} \quad \boxed{\text{Penalty term}}$$



Quadratic Hinge Errors

Quadratic hinge error:

- Advantages:
  - Solution quadratic SVM can be computed by solving a quadratic program.
  - Loss function is smooth.
  - Fast algorithms.
- Disadvantages:
  - Large errors have a disproportional influence the solution.
  - Sensitive for outliers.

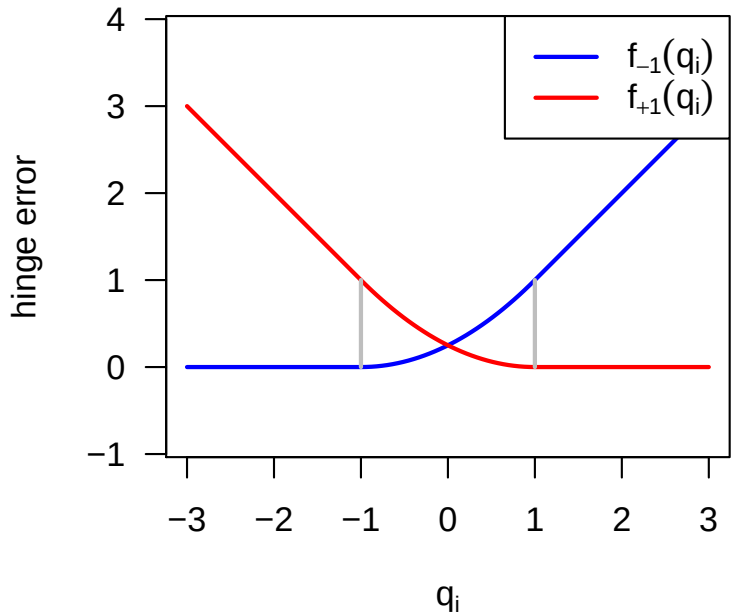
Huber Hinge Errors

Huber hinge error: $f_{-1}(q_i) = h(q_i, k)$

$$L_{\text{Quad-SVM}}(c, \mathbf{w})$$

$$= \sum_{i \in G_{-1}} f_{-1}(q_i) + \sum_{i \in G_{+1}} f_{+1}(q_i) + \lambda \mathbf{w}^\top \mathbf{w}$$

$$= \boxed{\text{Group } -1 \text{ error}} \quad \boxed{\text{Group } +1 \text{ error}} \quad \boxed{\text{Penalty term}}$$



Huber Hinge Errors

Huber hinge error:

- Advantages:
  - Robust error function (not sensitive for outliers).
  - Loss function is smooth.
  - Iterative solution is computationally fast.
- Disadvantages:
  - No analytic solution.

Different Hinge Errors

- With different choices of error functions, several known and unknown methods can be specified:
  - Absolute hinge: Classic SVM
  - Quadratic hinge: Quadratic SVM
  - Huber hinge: Smooth robust error
  - Quadratic around -1 and $+1$ without penalty: Linear discriminant analysis
  - Quadratic around -1 and $+1$ with penalty: Ridge regression
- If the error functions are convex, then the SVM loss function is also convex and only a global minimum exists.

Table of Contents

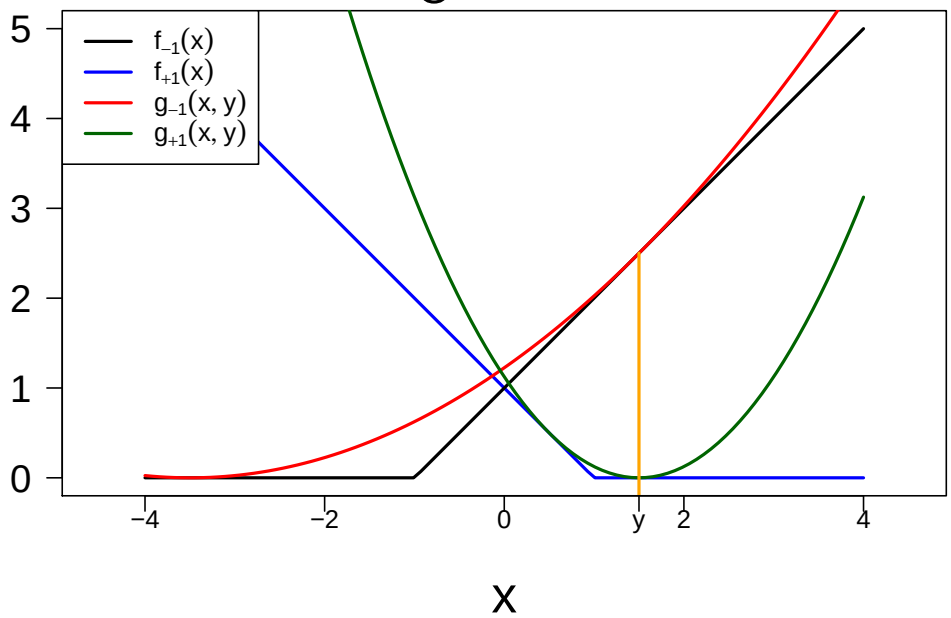
SVMMaj

Steps for deriving the majorization update:

1. For each i, find a quadratic majorizing function $g(q_i, \tilde{q}_i)$ of the hinge error $f_{-1}(q_i)$ or $f_{+1}(q_i)$.
2. Sum over all hinges and add penalty term to get the a majorizing function for $L_{\text{SVM}}(c, \mathbf{w})$.
3. Find minimum of the quadratic majorizing function

SVMMaj: Majorization of Absolute Hinge

Step 1a: Majorization of absolute hinge:

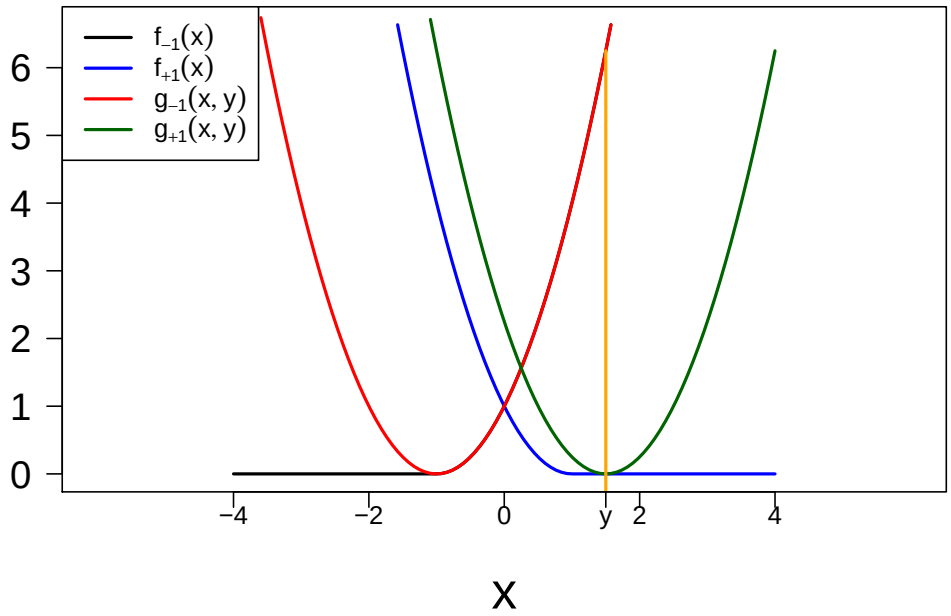


Majorization of absolute hinge is of the form: $g(x, y) = ax^2 - 2bx + c$,

Original function	a	b	c
$f_{-1}(x) = \max(0, x + 1)$	$(4\|y + 1\|)^{-1}$	$-(a + 1/4)$	$a + 1/2 + \|y + 1\|/4$
$f_{+1}(x) = \max(0, 1 - x)$	$(4\|1 - y\|)^{-1}$	$(a + 1/4)$	$a + 1/2 + \|1 - y\|/4$

SVMMaj: Majorization of Quadratic Hinge

Step 1b: Majorization of quadratic hinge :

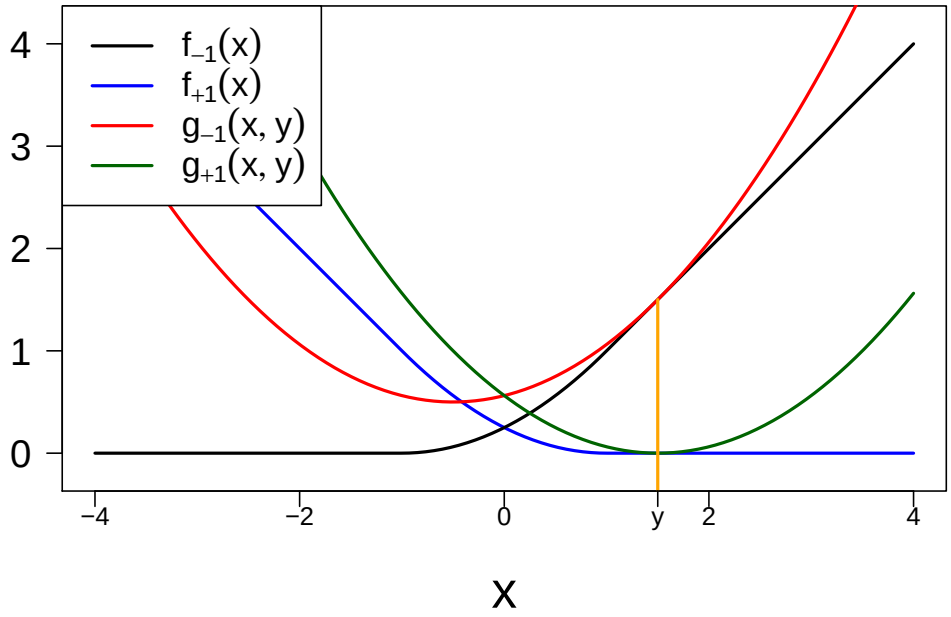


Majorization of quadratic hinge of the form: $g(x, y) = ax^2 - 2bx + c$

Original function	a	b	c
$f_{-1}(x) = \max^2(0, x + 1)$	1	$\begin{cases} y & \text{if } y \leq -1 \\ -1 & \text{if } y > -1 \end{cases}$	$\begin{cases} 1 - 2(y+1) + (y+1)^2 & \text{if } y \leq -1 \\ 1 & \text{if } y > -1 \end{cases}$
$f_{+1}(x) = \max^2(0, 1 - x)$	1	$\begin{cases} 1 & \text{if } y \leq 1 \\ y & \text{if } y > 1 \end{cases}$	$\begin{cases} 1 & \text{if } y \leq 1 \\ 1 - 2(1-y) + (y-1)^2 & \text{if } y > 1 \end{cases}$

SVMMaj: Majorization of Huber Hinge

Step 1c: Majorization of Huber hinge:



- Majorization of Huber hinge is of the form: $g(x, y) = ax^2 - 2bx + c$.
- For definition a, b, and c for Huber hinge, see Groenen et al. (2008).

SVMMaj

Step 2: Sum over all hinges and add penalty term

$$L_{\text{SVM}}(c, \mathbf{w}) \quad \leq \quad \sum_{i=1}^{n} a_i q_i^2 - 2\sum_{i=1}^{n} b_i q_i + \sum_{i=1}^{n} c_i + \lambda \mathbf{w}^\top \mathbf{w}$$

- Define $n \times (p+1)$ matrix $\mathbf{X}$ to have $\mathbf{1}$ as first column.
- $\mathbf{v}^\top = [c \ \mathbf{w}^\top]$ so that $q_i = c + \mathbf{x}_i^\top \mathbf{w}_i$ can be expressed as $\mathbf{q} = \mathbf{X}\mathbf{v}$.

$$\begin{aligned} L_{\text{SVM}}(\mathbf{v}) \quad &\leq \quad \sum_{i=1}^{n} a_i (\mathbf{x}_i^\top \mathbf{v})^2 - 2\sum_{i=1}^{n} b_i \mathbf{x}_i^\top \mathbf{v} + \sum_{i=1}^{n} c_i + \lambda \sum_{j=2}^{p+1} v_j^2 \\ &= \quad \mathbf{v}^\top \mathbf{X}^\top \mathbf{A}\mathbf{X}\mathbf{v} - 2\mathbf{v}^\top \mathbf{X}^\top \mathbf{b} + c_m + \lambda \mathbf{v}^\top \mathbf{P}\mathbf{v} \\ &= \quad \mathbf{v}^\top (\mathbf{X}^\top \mathbf{A}\mathbf{X} + \lambda \mathbf{P})\mathbf{v} - 2\mathbf{v}^\top \mathbf{X}^\top \mathbf{b} + c_m = g_{\text{SVM}}(\mathbf{v}, \tilde{\mathbf{v}}) \end{aligned}$$

$\mathbf{A}$ $n \times n$ diagonal matrix with elements $\{a_i\}$
$\mathbf{b}$ is an n-vector with elements b_i, $c_m = \sum_{i=1}^{n} c_i$, and
$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$

SVMMaj

Step 3: Minimize majorizing function

$$g_{\text{SVM}}(\mathbf{v}, \tilde{\mathbf{v}}) = \mathbf{v}^\top(\mathbf{X}^\top\mathbf{A}\mathbf{X} + \lambda\mathbf{P})\mathbf{v} - 2\mathbf{v}^\top\mathbf{X}^\top\mathbf{b} + c_m$$

- Setting gradient $\nabla g_{\text{SVM}}(\mathbf{v}, \tilde{\mathbf{v}}) = \mathbf{0}$ leads to the update $\mathbf{v}^+$ solving the linear system

$$(\mathbf{X}^\top\mathbf{A}\mathbf{X} + \lambda\mathbf{P})\mathbf{v} = \mathbf{X}^\top\mathbf{b}.$$

- Step-doubling $\mathbf{v}_{\text{upd}} = \mathbf{v}_{\text{old}} - 2(\mathbf{v}^+ - \mathbf{v}_{\text{old}})$ tends to halve the number of iterations because of quadratic majorization.
- Acceleration for quadratic hinge:
  - $\mathbf{A} = \mathbf{I}$,
  - Compute once $\mathbf{Z} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{P})^{-1}\mathbf{X}^\top$ and update iteratively by $\mathbf{v}^+ = \mathbf{Z}\mathbf{b}$

SVMMaj

- The SVMMaj algorithm for the binary SVM:

Choose with some initial $c^{(0)} \in \mathbb{R}^1$ and $\mathbf{w}^{(0)} \in \mathbb{R}^p$
Matrix $\mathbf{X}$ is $n \times (p+1)$ and has $\mathbf{1}$ as first column
Compute $L_{\text{SVM}}^{(0)} = L_{\text{SVM}}(c^{(0)}, \mathbf{w}^{(0)})$
Set $k \leftarrow 1$
while $k = 1$ or $\left(L_{\text{SVM}}^{(k-1)} - L_{\text{SVM}}^{(k)}\right) / L_{\text{SVM}}^{(k-1)} > \epsilon$ **do**
  $k \leftarrow k + 1$
  Compute $\mathbf{A}$ with elements a_{ii} depending on the chosen hinge
  Compute $\mathbf{b}$ with elements b_i depending on the chosen hinge
  Update $\mathbf{v}^+$ solves the linear system $(\mathbf{X}^\top \mathbf{A} \mathbf{X} + \lambda \mathbf{P})\mathbf{v} = \mathbf{X}^\top \mathbf{b}$
  Set $[c^{(k)}, \mathbf{w}^{(k)^\top}] = (\mathbf{v}^+)^\top$
  As a check, print k, $L_{\text{SVM}}^{(k)}$, and $L_{\text{SVM}}^{(k-1)} - L_{\text{SVM}}^{(k)}$
end

Table of Contents

Summary and Assignment

- GenSVM is a coherent generalization of binary SVM to multiple classes.
  - The SVM loss function is convex function so global minimum.
  - Interpretation in terms of a linear combination.
  - Nonlinear basis expansion including kernels can be used.
  - MM-algorithm for SVM (SVMMaj) is available.
  - Guaranteed descent until (close to) global minimum.
  - Classic hinge is robust but could be slow.
  - Huber hinge guarantees robustness and allows for smoothness.
  - Quadratic hinge is smooth and fast but could suffer from outiers.
- SVM extension to more than two classes exist: GenSVM and others.
- GenSVM algorithm scales to larger problems (up to $n = 500,000$).
- Nonlinear prediction through kernels is possible.
- For binary SVM, the SVMMaj package in R is on CRAN.
- GenSVM (using C-code) is available as an R-package and for python.

Summary and Assignment

Summary:

Week	Topics	Material
1	Introduction; Introduction to R; Linear methods for regression, model selection, and assessment	3.1, 3.2, 3.3, Xiong (2014)
2	Regularized regression and k-fold cross validation	3.4.1-3.4.3, 3.8.4, 7.10
3	Basis function expansions, kernels, bias-variance trade-off	5.1-5.2.1, 5.8, 7.3
4	Support vector machines	Groenen, Nalbantov, Bioch (2009); 12.1-12.3
5	Classification and regression trees, random forests, bootstrap	7.11, 9.2, 15
6	Boosting	10

To Do for Next Time

To Do for Next Time:

- Try to predict `y` (Did the client subscribed a term deposit? Answers: yes, no) in `bank.RData` through a support vector machine using the other relevant variables as predictors. More details of the orginal data are given in the file `bank-additional-names.txt`.
- Write your own R-function for a linear SVM using the SVMMaj MM algorithm provided in the slides.
- You can compare your results with the `svmmaj()` function of the `SVMMaj`-package and explain briefly whether or not they are the same and why this is so.
- These data of 4119 clients are a sample of 10% of the original sample. To accelerate your computations, you may take another random sample of 1000 clients.
- Try out some of the basis expansions discussed in Week 3 to fit a nonlinear SVM. You can use the SVMMaj package to do so
- Write a small 4-page report about the case according to the template.

References I

Patrick J. F. Groenen, Georgi Nalbantov, and Jan C. Bioch. SVM-Maj: a majorization approach to linear support vector machines with different hinge errors. Advances in Data Analysis and Classification, 2(1):17–43, 2008. URL http://link.springer.com/article/10.1007/s11634-008-0020-9.

Acknowledgement

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani