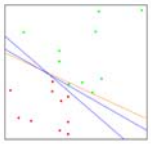


Using Support Vector Machines

Machine Learning Tutorial for Supervised Classification

Dr. – Ing. Morris Riedel
Adjunct Associated Professor
School of Engineering and Natural Sciences, University of Iceland
Research Group Leader, Juelich Supercomputing Centre, Germany

LECTURE 3

Support Vector Machines

July 7th, 2016
Barcelona

UNIVERSITY OF ICELAND SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH FORSCHUNGSZENTRUM

Review of Lecture 2

- Supervised Classification
  - Finding a target distribution for realistic learning situations
  - Assume unknown probability distribution over the input space
  - Hypothesis search with M models and we pick one
- Statistical Learning Theory

Unknown Target Distribution $P(y|\mathbf{x})$
target function $f: X \rightarrow Y$ plus noise
(ideal function)

$\mathbf{x} = (x_1, ..., x_d)$ ← Probability Distribution P on X

Hypothesis Set $\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(set of candidate formulas)

Final Hypothesis $g \approx f$

'Approximately' 'Probably' 'Number of samples N important for learning'

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$

Learning Algorithm ('train a system') $\mathcal{A}$
(set of known algorithms)

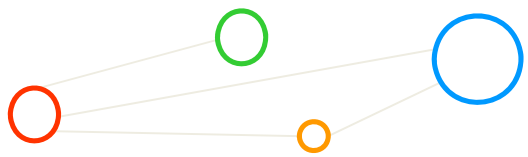
'learn: get error smaller' $E_{in}(g) \approx 0$ 'generalize well for unseen data' $E_{out}(g) \approx E_{in}(g)$

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ > \epsilon\ ]\ <=\ 4m_{\mathcal{H}}(2N)e^{-1/8\epsilon^2 N}$$

Shift the view to the data and we exchange M with growth function that is indeed depending on N

Lecture 3 – Support Vector Machines 2 / 58

Outline



Lecture 3 – Support Vector Machines 3 / 58

Outline of the Course

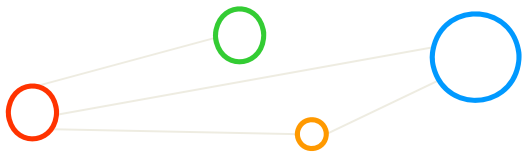
1. Machine Learning Fundamentals
2. Supervised Classification
3. Support Vector Machines
4. Applications and Serial Computing Limits
5. Kernel Methods
6. Applications and Parallel Computing Benefits

Outline

- Maximal Margin Classifier
  - Term Support Vector Machines Refined
  - Margin as Geometric Interpretation
  - Optimization Problem & Implementation
  - Solving and Limitations of Classifier
  - Apply Classifier to Flower Problem
- Support Vector Classifier
  - From Hard-margin to Soft-margin
  - Understand the slack variables
  - Role of Regularization Parameter C
  - Solving and Limitations of Classifier
  - Apply Classifier to Flower Problem

- **Promises from previous lectures reviewed…**
- ***Lecture 2:* Lecture 3 provides details on how support vector machines benefit from regularization methods**

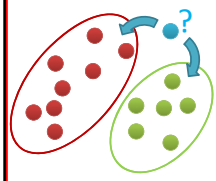
Maximum Margin Classifier



Methods Overview – Focus in this Lecture

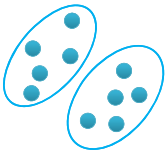
- **Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction**

Classification



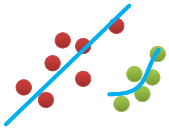
- Groups of data exist
- New data classified to existing groups

Clustering



- No groups of data exist
- Create groups from data close to each other

Regression



- Identify a line with a certain slope describing the data

Term Support Vector Machines Refined

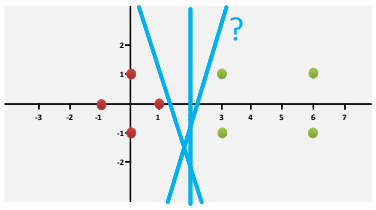
- **Support Vector Machines (SVMs) are a classification technique developed ~1990**
- **SVMs perform well in many settings & are considered as one of the best 'out of the box classifiers'**

[1] An Introduction to Statistical Learning

- Term detailed refinement into 'three separate techniques'
  - Practice: applications mostly use the SVMs with kernel methods
- 'Maximal margin classifier'
  - A simple and intuitive classifier with a 'best' linear class boundary
  - Requires that data is 'linearly separable'
- 'Support Vector Classifier'
  - Extension to the maximal margin classifier for non-linearly seperable data
  - Applied to a broader range of cases, idea of 'allowing some error'
- 'Support Vector Machines' → Using Non-Linear Kernel Methods
  - Extension of the support vector classifier
  - Enables non-linear class boundaries & via kernels;

Expected Out-of-Sample Performance for 'Any Line'

- We believe there is a (linear) pattern to be detected
  - Assumption: linearly seperable data (later non-seperable cases)
  - Performance question: What is the optimal line (decision boundary)?
  - E.g. green data: $\left\{\begin{pmatrix}3\\1\end{pmatrix},\begin{pmatrix}3\\-1\end{pmatrix},\begin{pmatrix}6\\1\end{pmatrix},\begin{pmatrix}6\\-1\end{pmatrix}\right\}$ red data: $\left\{\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}0\\-1\end{pmatrix},\begin{pmatrix}-1\\0\end{pmatrix}\right\}$



(PLA gives us just any line as soon as all samples are correctly classified)

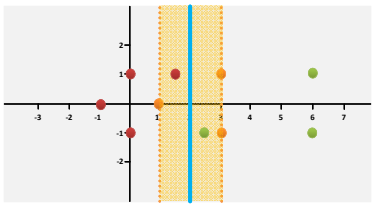
(How can we craft a margin expressing 'furthest away')

- **Intuition tells us just 'furthest away' from the closest points is a good position for the line – why?**

Exercises

Expected Out-of-Sample Performance for 'Best Line'

- The line with a 'bigger margin' seems to be better – but why?
  - Intuition: chance is higher that a new point will still be correctly classified
  - Fewer hypothesis possible: constrained by sized margin (cf. Lecture 2)
  - Idea: achieving good 'out-of-sample' performance is goal (cf. Lecture 2)



(e.g. better performance compared to PLA technique)

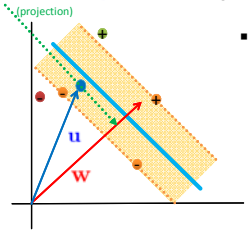
(simple line in a linear setup as intuitive decision boundary)

(Question remains: how we can achieve a bigger margin)

- **Support Vector Machines (SVMs) use maximum margins that will be mathematically established**

Geometric SVM Interpretation and Setup (1)

- Think 'simplified coordinate system' and use 'Linear Algebra'
  - Many other samples are removed (red and green not SVs) ⊖ ⊕
  - Vector $\mathbf{w}$ of 'any length' perpendicular to the decision boundary
  - Vector $\mathbf{u}$ points to an unknown quantity (e.g. new sample to classify)
  - Is $\mathbf{u}$ on the left or right side of the decision boundary?

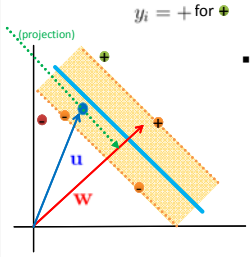


- Dot product $\mathbf{w} \cdot \mathbf{u} \geq C; C = -b$
  - With $\mathbf{u}$ takes the projection on the $\mathbf{w}$
  - Depending on where projection is it is left or right from the decision boundary
  - Simple transformation brings decison rule:
  - (1) $\mathbf{w} \cdot \mathbf{u} + b \geq 0$ → means ⊕
  - (given that b and $\mathbf{w}$ are unknown to us)

(constraints are not enough to fix particular b or w, need more constraints to calculate b or w)

Geometric SVM Interpretation and Setup (2)

- Creating our constraints to get b or $\mathbf{w}$ computed
  - First constraint set for positive samples ⊕ $\mathbf{w} \cdot \mathbf{x}_+ + b \geq 1$
  - Second constraint set for negative samples ⊖ $\mathbf{w} \cdot \mathbf{x}_- + b \leq 1$
  - For mathematical convenience introduce variables (i.e. labelled samples)
    $y_i = +$ for ⊕ and $y_i = -$ for ⊖



- Multiply equations by y_i
  - Positive samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$
  - Negative samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$
  - Both same due to $y_i = +$ and $y_i = -$
    (brings us mathematical convenience often quoted)

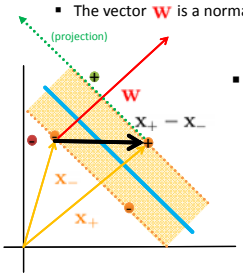
$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0$$

(additional constraints just for support vectors itself helps)

2 $$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$$

Geometric SVM Interpretation and Setup (3)

- Determine the 'width of the margin'
  - Difference between positive and negative SVs: $\mathbf{x}_+ - \mathbf{x}_-$
  - Projection of $\mathbf{x}_+ - \mathbf{x}_-$ onto the vector $\mathbf{w}$
  - The vector $\mathbf{w}$ is a normal vector, magnitude is $\|\mathbf{w}\|$



(Dot product of two vectors is a scalar, here the width of the margin)

- Unit vector is helpful for 'margin width'
  - Projection (dot product) for margin width:

$$\mathbf{x}_+ - \mathbf{x}_- \cdot \boxed{\frac{\mathbf{w}}{\|\mathbf{w}\|}} \quad \text{(unit vector)}$$

$1 - b \quad 1 + b$

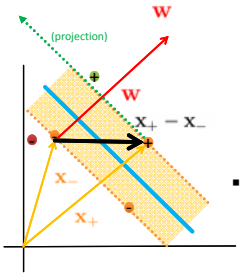
$$\frac{2}{\|\mathbf{w}\|} \quad 3$$

- When enforce constraint: $y_i = +$ ⊕, $y_i = -$ ⊖

2 $$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$$

Constrained Optimization Steps SVM (1)

- Use 'constraint optimization' of mathematical toolkit
  - Idea is to 'maximize the width' of the margin: $max \frac{2}{\|\mathbf{w}\|}$ (drop the constant 2 is possible here)



$$\Rightarrow max \frac{1}{\|\mathbf{w}\|} \quad \text{(equivalent)}$$

$$\Rightarrow min \|\mathbf{w}\| \quad \text{(equivalent for max)}$$

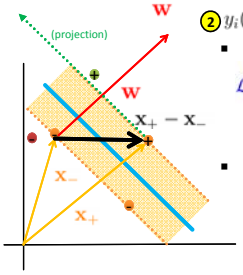
$$\Rightarrow min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{(mathematical convenience)} \quad 3$$

- Next: Find the extreme values
  - Subject to constraints

2 $$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$$

Constrained Optimization Steps SVM (2)

- Use 'Lagrange Multipliers' of mathematical toolkit
  - Established tool in 'constrained optimization' to find function extremum
  - 'Get rid' of constraints by using Lagrange Multipliers (4)



(2) $y_i(\mathbf{x}_i \cdot \mathbf{w} + b - 1) = 0$

- Introduce a multiplier for each constraint

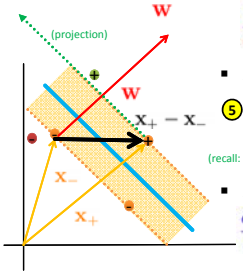
$$\mathcal{L}(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i [y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$$

(interesting: non zero for support vectors, rest zero)

- Find derivatives for extremum & set 0
  - But two unknowns that might vary
  - First differentiate w.r.t. $\mathbf{w}$
  - Second differentiate w.r.t. b

Constrained Optimization Steps SVM (3)

- Lagrange gives: $\mathcal{L}(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i [y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$



- First differentiate w.r.t $\mathbf{w}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum \alpha_i y_i \mathbf{x}_i = 0$$

- Simple transformation brings:

(5) $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$ (i.e. vector is linear sum of samples)

(recall: non zero for support vectors, rest zero → even less samples)

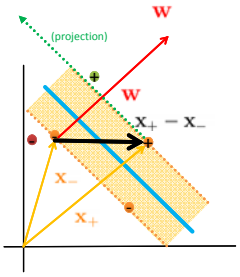
- Second differentiate w.r.t. b

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum \alpha_i y_i = 0 \Rightarrow \sum \alpha_i y_i = 0 \quad (5)$$

Constrained Optimization Steps SVM (4)

- Lagrange gives: $\mathcal{L}(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i [y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$

(plug into)



- Find minimum
  - Quadratic optimization problem
  - Take advantage of (5) $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

$$\mathcal{L} = \frac{1}{2}\left(\sum \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum \alpha_j y_j \mathbf{x}_j\right)$$

$$- \sum \alpha_i y_i \mathbf{x}_i \cdot \left(\sum \alpha_j y_j \mathbf{x}_j\right)$$

$$- \sum \alpha_i y_i b + \sum \alpha_i$$

(b constant in front sum)

(5) $\sum \alpha_i y_i = 0$

Constrained Optimization Steps SVM (5)

- Rewrite formula: $\mathcal{L} = \frac{1}{2}\left(\sum \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum \alpha_j y_j \mathbf{x}_j\right) - \sum \alpha_i y_i \mathbf{x}_i \cdot \left(\sum \alpha_j y_j \mathbf{x}_j\right)$ (the same)

$$\cancel{-\sum \alpha_i y_i b} + \sum \alpha_i$$

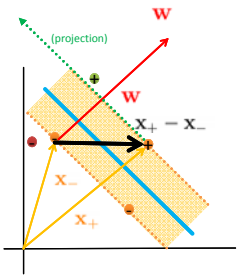
(was 0)

(results in)

(optimization depends only on dot product of samples)

$$\mathcal{L} = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (6)$$

- Equation to be solved by some quadratic programming package



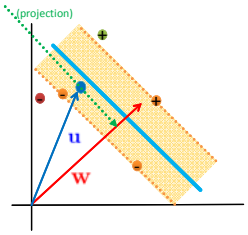
Use of SVM Classifier to Perform Classification

- Use findings for decision rule

(5) $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

(1) $\mathbf{w} \cdot \mathbf{u} + b \geq 0$ ⊕ ➡ $\sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u}_i + b \geq 0$ ⊕

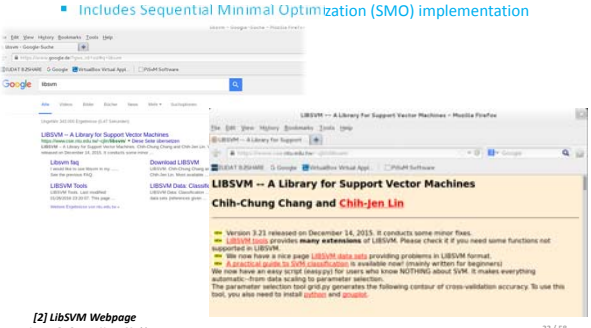
(decision rule also depends on dotproduct)



Exercises

LibSVM – Defacto Standard SVM Implementation

- Free available tool
  - Includes Sequential Minimal Optimization (SMO) implementation



[2] LibSVM Webpage

LibSVM – Download

- Download tar.gz (or in Windows zip bundle)

[2] LibSVM Webpage

- Put package in a folder of your choice
  - Alternatively copy file to your usual working environment

```
adminuser@linux-8djg:~/tools> scp libsvm-3.21.tar.gz mriedel@jureca.fz-juelich.de:/homeb/zam/mriedel
libsvm-3.21.tar.gz                                    100%  827KB 827.4KB/s   00:00

-bash-4.2$ ls -al
total 64
drwxr-xr-x  2 mriedel zam    512 Jul  6 20:00 .
drwxr-xr-x 29 mriedel zam  32768 Jul  6 19:58 ..
-rw-r--r--  1 mriedel zam 847291 Jul  6 20:00 libsvm-3.21.tar.gz
-bash-4.2$ pwd
/homeb/zam/mriedel/serialtools
```


LibSVM – Unpack the Bundle

- Untar (or Unzip in Windows)

```
/homeb/zam/mriedel/serialtools
-bash-4.2$ tar -zxvf libsvm-3.21.tar.gz
libsvm-3.21/
libsvm-3.21/COPYRIGHT
libsvm-3.21/svm-predict.c
libsvm-3.21/svm.cpp
libsvm-3.21/README
libsvm-3.21/Makefile.win
libsvm-3.21/svm.h
libsvm-3.21/heart_scale
libsvm-3.21/java/
libsvm-3.21/java/svm_toy.java
libsvm-3.21/java/svm_scale.java
libsvm-3.21/java/libsvm/
libsvm-3.21/java/libsvm/svm_model.java
libsvm-3.21/java/libsvm/svm.m4
libsvm-3.21/java/libsvm/svm_problem.java
libsvm-3.21/java/libsvm/svm.java
libsvm-3.21/java/libsvm/svm_node.java
libsvm-3.21/java/libsvm/svm_parameter.java
libsvm-3.21/java/libsvm/svm_print_interface.java
libsvm-3.21/java/svm_train.java
libsvm-3.21/java/Makefile
libsvm-3.21/java/test_applet.html
libsvm-3.21/java/libsvm.jar
libsvm-3.21/java/svm_predict.java
libsvm-3.21/Makefile
libsvm-3.21/windows/
libsvm-3.21/windows/svm-toy.exe
libsvm-3.21/windows/svm-scale.exe
libsvm-3.21/windows/svmtrain.mexw64
libsvm-3.21/windows/libsvmwrite.mexw64
libsvm-3.21/windows/libsvm.dll
```

```
/homeb/zam/mriedel/serialtools/libsvm-3.21
-bash-4.2$ ls -al
total 544
drwxr-xr-x 8 mriedel zam 32768 Dec 14  2015 .
drwxr-xr-x 3 mriedel zam   512 Jul  6 20:03 ..
-rw-r--r-- 1 mriedel zam  1497 Dec 14  2015 COPYRIGHT
-rw-r--r-- 1 mriedel zam 83989 Dec 14  2015 FAQ.html
-rw-r--r-- 1 mriedel zam 27679 Dec 14  2015 heart_scale
drwxr-xr-x 3 mriedel zam   512 Dec 14  2015 java
-rw-r--r-- 1 mriedel zam   732 Dec 14  2015 Makefile
-rw-r--r-- 1 mriedel zam  1136 Dec 14  2015 Makefile.win
drwxr-xr-x 2 mriedel zam   512 Dec 14  2015 matlab
drwxr-xr-x 2 mriedel zam   512 Dec 14  2015 python
-rw-r--r-- 1 mriedel zam 20679 Dec 14  2015 README
-rw-r--r-- 1 mriedel zam 64936 Dec 14  2015 svm.cpp
-rw-r--r-- 1 mriedel zam   477 Dec 14  2015 svm.def
-rw-r--r-- 1 mriedel zam  3382 Dec 14  2015 svm.h
-rw-r--r-- 1 mriedel zam  5536 Dec 14  2015 svm-predict.c
-rw-r--r-- 1 mriedel zam  8539 Dec 14  2015 svm-scale.c
drwxr-xr-x 5 mriedel zam   512 Dec 14  2015 svm-toy
-rw-r--r-- 1 mriedel zam  8986 Dec 14  2015 svm-train.c
drwxr-xr-x 2 mriedel zam   512 Dec 14  2015 tools
drwxr-xr-x 2 mriedel zam   512 Dec 14  2015 windows
```

LibSVM – Make (only in UNIX)

- Use make to generate executables (needs g++ compiler)

```
-bash-4.2$ pwd
/homeb/zam/mriedel/serialtools/libsvm-3.21
-bash-4.2$ make
g++ -Wall -Wconversion -O3 -fPIC -c svm.cpp
g++ -Wall -Wconversion -O3 -fPIC svm-train.c svm.o -o svm-train -lm
g++ -Wall -Wconversion -O3 -fPIC svm-predict.c svm.o -o svm-predict -lm
g++ -Wall -Wconversion -O3 -fPIC svm-scale.c -o svm-scale
```

- Check executables important for us

```
-bash-4.2$ pwd
/homeb/zam/mriedel/serialtools/libsvm-3.21
-bash-4.2$ ls -al
total 896
drwxr-xr-x 8 mriedel zam  32768 Jul  6 20:05 .
drwxr-xr-x 3 mriedel zam    512 Jul  6 20:03 ..
-rw-r--r-- 1 mriedel zam   1457 Dec 14  2015 COPYRIGHT
-rw-r--r-- 1 mriedel zam  83989 Dec 14  2015 FAQ.html
-rw-r--r-- 1 mriedel zam  27670 Dec 14  2015 heart_scale
drwxr-xr-x 3 mriedel zam    512 Dec 14  2015 java
-rw-r--r-- 1 mriedel zam    732 Dec 14  2015 Makefile
-rw-r--r-- 1 mriedel zam   1136 Dec 14  2015 Makefile.win
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 matlab
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 python
-rw-r--r-- 1 mriedel zam  28679 Dec 14  2015 README
-rw-r--r-- 1 mriedel zam  64836 Dec 14  2015 svm.cpp
-rw-r--r-- 1 mriedel zam    477 Dec 14  2015 svm.def
-rw-r--r-- 1 mriedel zam   3382 Dec 14  2015 svm.h
-rw-r--r-- 1 mriedel zam  79278 Jul  6 20:05 svm-predict
-rw-r--r-- 1 mriedel zam  18587 Jul  6 20:05 svm-scale
-rw-r--r-- 1 mriedel zam   8539 Dec 14  2015 svm-scale.c
-rwxr-xr-x 1 mriedel zam  78599 Jul  6 20:05 svm-train
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 tools
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 windows
```

(use in testing phase)

(use in training phase)

[2] LibSVM Webpage

LibSVM – svm-train Parameters

- Important parameters (training phase)

```
-bash-4.2$ ./svm-train
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC              (multi-class classification)
        1 -- nu-SVC             (multi-class classification)
        2 -- one-class SVM
        3 -- epsilon-SVR        (regression)
        4 -- nu-SVR             (regression)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
        4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates : whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-q : quiet mode (no outputs)
```

(we need a training set file)

(take default here = C-SVC)

(in this lecture we have just 'linear kernels')

(Regularization Parameter)

Training Examples

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

[2] LibSVM Webpage

LibSVM – svm-predict Parameters

- Important parameters (testing phase)

```
-bash-4.2$ ./svm-predict
Usage: svm-predict [options] test_file model_file output_file
options:
-b probability_estimates: whether to predict probability estimates, 0 or 1 (default 0); for one-class SVM only 0 is supported
-q : quiet mode (no outputs)
```

(the model file is generated in the training phase → the support vectors found in optimization)

(test file is a testing dataset set aside to be used once training is finished)

(output file gives us indications how each sample was classified)

Testing Examples

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Maximal Margin Classifier – Training Set and Test Set

- Classification technique
  - Given 'labelled dataset'
  - Data matrix X (n x p)
  - Training set:
    n training samples
  - p-dimensional space
  - Linearly seperable data
  - Binary classification problem
    (two class classification)
  - Test set:
    a vector x* with test observations

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

(n x p-dimensional vectors)

$$y_1, \ldots, y_n \in \{-1, 1\}$$

(class labels) (two classes)

$$x^* = \begin{pmatrix} x_1^* & \ldots & x_p^* \end{pmatrix}^T$$

- **Maximal Margin Classifiers create a seperating hyperplane that seperates the training set samples perfectly according to their class labels following a reasonable way of which hyperplane to use**

[1] An Introduction to Statistical Learning

Exercises

Download IRIS Dataset from LibSVM Datasets

- Scaled version of our data (cf. Lecture 1 & 2): iris.scale
  - Scaling is used in order that the optimization does not have to work with large numbers – so one can scale, but it is not a requirement
  - Sometimes the performance improved with scaling

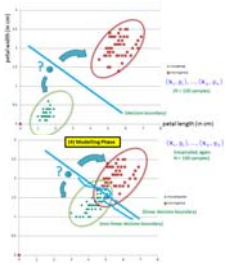
[2] LibSVM Webpage

Data Preparation Phase

- Copy IRIS Dataset in your working environment

```
adminuser@linux-8djg:~/data> scp iris.scale mriedel@jureca.fz-juelich.de:/homeb/zam/mriedel
iris.scale                                                    100% 6954     6.8KB/s   00:00
/homeb/zam/mriedel/datasets
-bash-4.2$ ls -al
total 64
drwxr-xr-x  2 mriedel zam   512 Jul  6 21:53 .
drwxr-xr-x 30 mriedel zam 32768 Jul  6 21:53 ..
-rw-r--r--  1 mriedel zam  6954 Jul  6 21:53 iris.scale
```

- Dataset Two-class problem, linearly seperable
  - Dataset Iris Setosa (class 1) and Iris Virginica (class 3)
  - Iris-class1and3-training(20)/testing(30)
- Dataset Two-class problem, not linearly seperable
  - Dataset Iris Veriscolor (class 2) and Iris Virginica (class3)
  - iris-class2and3-training(20)/testing(30)



Data Download

- Download the following files from B2Share

B2SHARE *[3] Iris Dataset LibSVM Format Preprocessing*

Training Phase: linearly seperable case (iris-class1and3)

- Use svm-train (c<=0 not allowed)

```
-bash-4.2$ more svm-train1-3.sh
./svm-train -t 0 -c 1 /homeb/zam/mriedel/datasets/iris-classland3-training

-bash-4.2$ ./svm-train1-3.sh
*
optimization finished, #iter = 11
nu = 0.035498
obj = -0.709742, rho = 0.447384
nSV = 4, nBSV = 0
Total nSV = 4
```

```
-bash-4.2$ ls -al
total 896
drwxr-xr-x 8 mriedel zam  32768 Jul  6 22:25 .
drwxr-xr-x 3 mriedel zam    512 Jul  6 20:03 ..
-rw-r--r-- 1 mriedel zam   1497 Dec 14  2015 COPYRIGHT
-rw-r--r-- 1 mriedel zam  83089 Dec 14  2015 FAQ.html
-rw-r--r-- 1 mriedel zam  27639 Dec 14  2015 heart_scale
-rw-r--r-- 1 mriedel zam    354 Jul  6 22:25 iris-classland3-training.model
-rw-r--r-- 1 mriedel zam    732 Dec 14  2015 Makefile
-rw-r--r-- 1 mriedel zam   1136 Dec 14  2015 Makefile.win
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 matlab
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 python
-rw-r--r-- 1 mriedel zam  28679 Dec 14  2015 README
-rw-r--r-- 1 mriedel zam  64836 Dec 14  2015 svm.zip
-rw-r--r-- 1 mriedel zam    477 Dec 14  2015 svm.def
-rw-r--r-- 1 mriedel zam   3382 Dec 14  2015 svm.h
-rw-r--r-- 1 mriedel zam 100224 Jul  6 20:05 svm.o
-rwxr-xr-x 1 mriedel zam  78279 Jul  6 20:05 svm-predict
-rw-r--r-- 1 mriedel zam   5536 Dec 14  2015 svm-predict.c
-rwxr-xr-x 1 mriedel zam  18587 Jul  6 20:05 svm-scale
-rw-r--r-- 1 mriedel zam   8539 Dec 14  2015 svm-scale.c
drwxr-xr-x 5 mriedel zam    512 Dec 14  2015 svm-toy
-rwxr-xr-x 1 mriedel zam  78589 Jul  6 20:05 svm-train
-rwxr-xr-x 1 mriedel zam     76 Jul  6 22:24 svm-train1-3.sh
-rw-r--r-- 1 mriedel zam   8986 Dec 14  2015 svm-train.c
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 tools
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 windows
```

- Check model file

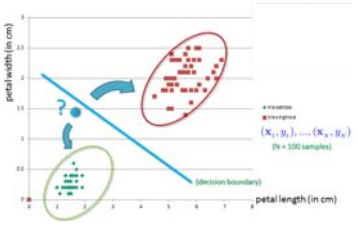
```
-bash-4.2$ more iris-classland3-training.model
svm_type c_svc
kernel_type linear
nr_class 2
total_sv 4
rho 0.447384
label 1 3
nr_sv 3 1
SV
0.137408671956105 1:-0.666667 2:-0.166667 3:-0.864407 4:-0.916667
0.191198032934598 1:-0.500889 2:0.583333 3:-0.762712 4:-0.75
0.471154854114213 1:-0.944444 2:-0.25 3:-0.864407 4:-0.916667
-0.709742798840956 1:-0.666667 2:-0.583333 3:0.186441 4:0.333333
```

Testing Phase: linearly seperable case (iris-class1and3)

- Use svm-predict (using newly created model file & testing data)

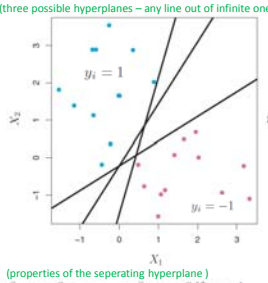
```
-bash-4.2$ more svm-predict1-3.sh
./svm-predict /homeb/zam/mriedel/datasets/iris-classland3-testing ./iris-classland3-training.model ./results.txt

-bash-4.2$ ./svm-predict1-3.sh
Accuracy = 100% (60/60) (classification)

-bash-4.2$ head results.txt
1
1
1
1
1
1
1
1
1
1
```

(consistent with our graph: 100% here possible since very easy problem, in practice rarely)

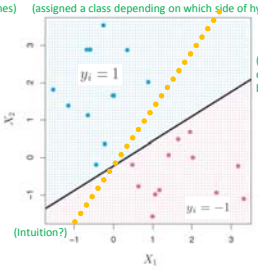


Maximal Margin Classifier – Use of Seperating Hyperplanes

(three possible hyperplanes – any line out of infinite ones)



(assigned a class depending on which side of hyperplane)



(linear decision boundary)

(but is this the best line…?)

(Intuition?)

(properties of the seperating hyperplane)

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} > 0 \text{ if } y_i = 1$$
$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} < 0 \text{ if } y_i = -1$$
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0 \text{for all } i = 1, \ldots, n$$

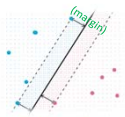
(using testset to predict and assign labels via sign)

$$\text{sign of } f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \ldots + \beta_p x_p^*$$
$$f(x^*) \text{ is negative assign it to class } -1$$
$$f(x^*) \text{ is positive assign it to class } 1$$

modified from [1] An Introduction to Statistical Learning

Maximal Margin Classifier – Reasoning and Margin Term

- Reasoning to pick the 'best line'
  - There exists a 'maximal margin hyperplane' (optimal seperating hyperplane)
  - Hyperplane that is 'farthest away' from the training set samples
- Identify the 'margin' itself
  - Compute the 'perpendicular distance' (point 'right angle 90 degrees' distance to the plane)
  - From each training sample to a given separating hyperplane
  - The smallest such distance is the 'minimal distance' from the observations to the hyperplane – the margin
- Identify 'maximal margin'
  - Identify the hyperplane that has the 'farthest minimum distance' to the training observations
  - Also named the 'optimal seperating hyperplane'



- **The maximal margin hyperplane is the seperating hyperplane for which the margin is largest**

[1] An Introduction to Statistical Learning

Maximal Margin Classifier – Margin Performance

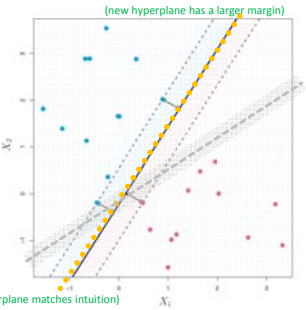
- Classification technique
  - Classify testset samples based on which side of the maximal margin hyperplane they are lying

$$\text{sign of } f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \ldots + \beta_p x_p^*$$

(ßs are the coefficients of the maximal margin hyperplane)

  - Assuming that a classifier that has a large margin on the training data will also have a large margin on the test data (cf. also 'the intuitive notion')
  - Testset samples will be thus correctly classified

(new hyperplane has a larger margin)



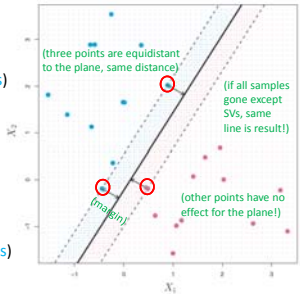
(hyperplane matches intuition)

(Compared to grey hyperplane: a 'greater minimal distance' between the data points and the seperating hyperplane)

modified from [1] An Introduction to Statistical Learning

Maximal Margin Classifier – Support Vector Term

- Observation
  - Three data points lie on the edge of margin (somewhat special data points)
  - Dashed lines indicating the width of the margin (very interesting to know)
  - Margin width is the distance from the special data points to the hyperplane (hyperplane depends directly on small data subset: SV points)



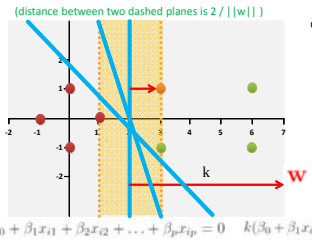
modified from [1] An Introduction to Statistical Learning

- **Points that lie on the edge of the margin are named support vectors (SVs) in p-dimensional space**
- **SVs 'support' the maximal margin hyperplane: if SVs are moved → the hyperplane moves as well**

Maximal Margin Classifier – Optimization and W Vector

- Which weight $\mathbf{w}$ maximizes the margin?
  - Margin is just a distance from 'a line to a point', goal is to minimize $\mathbf{w}$
  - Pick x_n as the nearest data point to the line (or hyper-plane)…

(distance between two dashed planes is 2 / ||w||)



- Reduce the problem to a 'constraint optimization problem' (vector $\mathbf{w}$ are the ß coefficients)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\ M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

(for points on plane w must be 0, interpret k as length of w)

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} = 0 \qquad k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) = 0 \text{ for any } k \neq 0$$

- **Support vectors achieve the margin and are positioned exactly on the boundary of the margin**

Maximal Margin Classifier – Optimization and N Samples

- Approach: Maximizing the margin
  - Equivalent to minimize objective function
    (original and modified have same w and b)

$$\min_{w,\ b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\}$$

(substituted for plain || w || for mathematical convenience)

subject to y_i ($\mathbf{w} \cdot x_i - b$) >= 1

- 'Lagrangian Dual problem' (chain of math turns optimization problem into solving this)
  - Use of already established Lagrangian method :

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m$$

(big data impact, important dot product)

- Interesting properties
  - Simple function: Quadratic in alpha
  - Simple constraints in this optimization problem (not covered here)
  - Established tools exist: Quadratic Programming (qp) (rule of thumb)

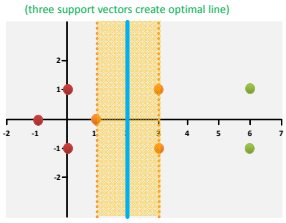
- **Practice shows that #N moderate is ok, but large #N ('big data') are problematic for computing**
- **Quadratic programming and computing the solving depends on number of samples N in the dataset**

Lecture 3 – Support Vector Machines 40 / 58

Maximal Margin Classifier – Optimization and # SV Impacts

- Interpretation of QP results (vector of alpha is returned)
  - The obtained values of alpha (lagrange multipliers) are mostly 0
  - Only a couple of alphas are > 0 and special: the support vectors (SVs)

(three support vectors create optimal line)



- N x N, usually not sparse (big data challenge)
- Computational complexity relies in the following: (e.g. all datasets vs. sampling)

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m$$

$$\begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \dots y_1 y_N x_1^T x_N \\ \dots & \dots & \ddots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \dots y_N y_N x_N^T x_N \end{bmatrix}$$

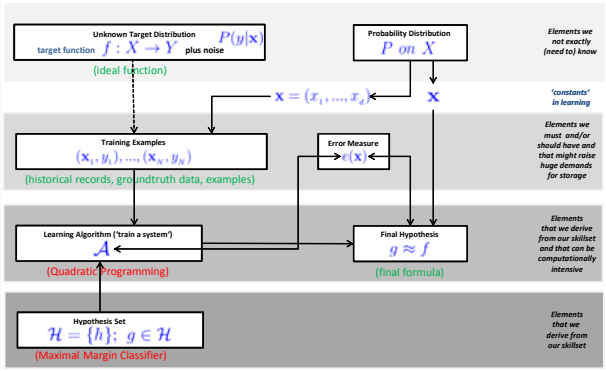
(quadratic coefficients, alphas are result from QP)

(rule of thumb)

- **Generalization measure: #SVs as 'in-sample quantity' → 10SVs/1000 samples ok, 500SVs/1000 bad**
- **Reasonsing towards overfitting due to a large number of SVs (fit many, small margin, gives bad E_{out})**

Lecture 3 – Support Vector Machines 41 / 58

Solution Tools: Maximal Margin Classifier & QP Algorithm



Lecture 3 – Support Vector Machines 42 / 58

Maximal Margin Classifier – Solving and Limitations

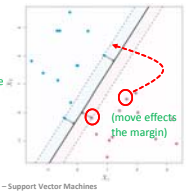
- **Solving constraint optimization problem chooses coefficients that maximize M & gives hyperplane**
- **Solving this problem efficiently is possible techniques like sequential minimal optimization (SMO)**
- **Maximal margin classifiers use a hard-margin & thus only work with exact linearly seperable data**

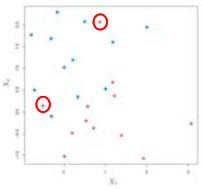
modified from [1] An Introduction to Statistical Learning

- Limitation
  - Non linearly separable data (given mostly in practice)
  - Optimization problem has no solution M > 0 (think point moves over plane)
  - No separating hyperplane can be created (classifier can not be used)

(no error allowed, a 'hard margin')

(allow some error the margin will be bigger, … maybe better E_{out})

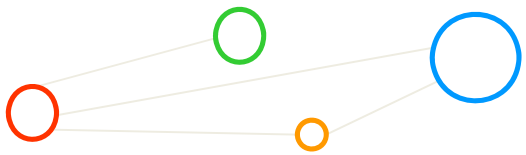




(no exact seperation possible)

(… but with allowing some error maybe, a 'soft margin'…)

Support Vector Classifier

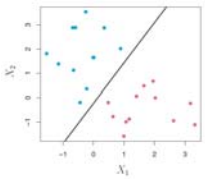


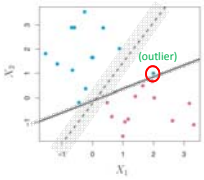
Support Vector Classifiers – Motivation

- **Support Vector Classifiers develop hyperplanes with soft-margins to achieve better performance**
- **Support Vector Classifiers aim to avoid being sensitive to individual observations (e.g. outliers)**

[1] An Introduction to Statistical Learning

- Approach
  - Generalization of the 'maximal margin classifier' (get most & but not all training data correcly classified)
  - Include non-seperable cases with a soft-margin (almost instead of exact)
  - Being more robust w.r.t. extreme values (e.g. outliers) allowing small errors





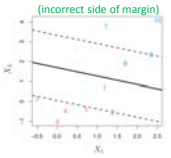
(significant reduction of the maximal margin)

(overfitting, cf Lecture 10: maximal margin classifier & hyperplane is very sensitive to a change of a single data point)

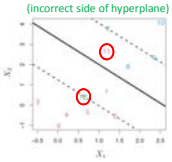
Support Vector Classifiers – Modified Technique Required

- Previous classification technique reviewed
  - Seeking the largest possible margin, so that every data point is…
  - … on the correct side of the hyperplane
  - … on the correct side of the margin
- Modified classification technique
  - Enable 'violations of the hard-margin' to achieve 'soft-margin classifier'
  - Allow violations means: allow some data points to be…
  - … on the incorrect side of the margin
  - … even on the incorrect side of the hyperplane (which leads to a misclassification of that point)

(incorrect side of margin)



(incorrect side of hyperplane)



[1] An Introduction to Statistical Learning

(SVs refined: data points that lie directly on the margin or on the wrong side of the margin for their class are the SVs → affect support vector classifer)

Support Vector Classifier – Optimization Problem Refined

- Optimization Problem
  - Still maximize margin M
  - Refining constraints to include violoation of margins
  - Adding slack variables $\epsilon_1, \ldots, \epsilon_n$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

(allow datapoints to be on the wrong side of the margin or hyperplane)

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$ (slightly violate the Margin)

$$\boxed{\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C}$$

(C is used here to bound the error)

(C is a nonnegative tuning parameter useful for regularization, cf. Lecture 10, picked by cross-validation method)

- C Parameter & slacks
  - C bounds the sum of the slack variables $\epsilon_1, \ldots, \epsilon_n$

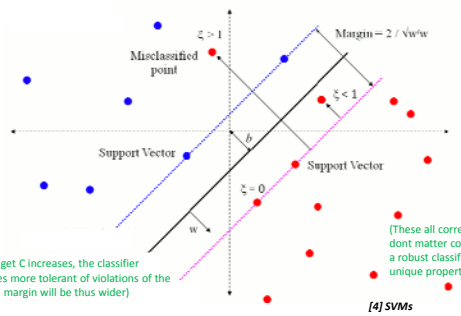
[1] An Introduction to Statistical Learning

- **C determines the number & severity of violations that will be tolerated to margin and hyperplane**
- **Interpret C as budget (or costs) for the amount that the margin can be violated by n data points**

➢ **Lecture 4 provides details on validation methods that provides a value for C in a principled way**

Support Vector Classifier – Understanding the Slack

- Allowing some errors or violations of the margin



(As budget C increases, the classifier becomes more tolerant of violations of the margin, margin will be thus wider)

(These all correct samples dont matter contributing to a robust classifier and unique property of SVCs)

[4] SVMs

Support Vector Classifier – Impact of Regularization with C

- Approach: Maximizing the margin
  - Equivalent to minimize objective function

$$\min_{w,} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 \right\}$$

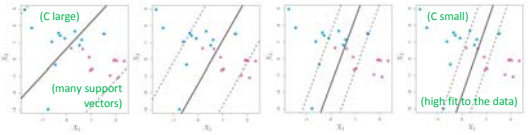
(maximal margin classifier)

- Support Vector Classifier
  - Same approach for solving
  - Adding slacks variables
  - C parameter that enables regularization (i.e. size of margin)

$$\min_{w,\xi_i} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \right\}$$

(C is used here to multiply the sum of errors for increased severity, because # errors should be kept small, not a budget here)

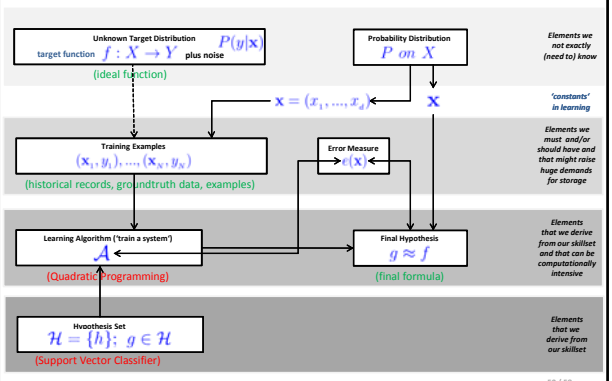
[1] An Introduction to Statistical Learning



(fewer support Vectors: just #8)

➢ **Lecture 4 provides details on validation methods that provides a value for C in a principled way**

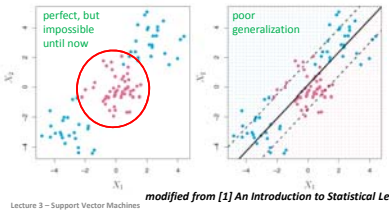
Solution Tools: Support Vector Classifier & QP Algorithm



Support Vector Classifier – Solving & Limitations

- **Solving constraint optimization problem chooses coefficients that maximize M & gives hyperplane**
- **Solving this problem efficiently is possible due to sequential minimal optimization (SMO)**
- **Support vector classifiers use a soft-margin & thus work with slightly(!) non-linearly seperable data**

- Limitation: Still linear decision boundary…
  - Non linearly separable where soft-margins are no solution
  - Support Vector Classifier can not establish a non-linear boundary



(a linear decision boundary by support vector classifier is not an option)

(… but maybe there are more advanced techniques that help…)

modified from [1] An Introduction to Statistical Learning

Exercises

Training Phase: non-linearly seperable case (iris-class2and3)

- Use svm-train (c<=0 not allowed, change value, what happens?)

```
-bash-4.2$ more svm-train2-3.sh
./svm-train -t 0 -c 1 /homeb/zam/mriedel/datasets/iris-class2and3-training
```

```
-bash-4.2$ ./svm-train2-3.sh
*
optimization finished, #iter = 22
nu = 0.360979
obj = -17.632638, rho = -1.602841
nSV = 27, nBSV = 24
Total nSV = 27
```

```
-bash-4.2$ ls -al
total 896
drwxr-xr-x 8 mriedel zam  32768 Jul  6 22:36 .
drwxr-xr-x 3 mriedel zam    512 Jul  6 20:03 ..
-rw-r--r-- 1 mriedel zam   1497 Dec 14  2015 COPYRIGHT
-rw-r--r-- 1 mriedel zam  83989 Dec 14  2015 FAQ.html
-rw-r--r-- 1 mriedel zam  27670 Dec 14  2015 heart_scale
-rw-r--r-- 1 mriedel zam   1436 Jul  6 22:36 iris-class2and3-training.model
-rw-r--r-- 1 mriedel zam    732 Dec 14  2015 Makefile
-rw-r--r-- 1 mriedel zam   1136 Dec 14  2015 Makefile.win
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 matlab
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 python
-rw-r--r-- 1 mriedel zam  28679 Dec 14  2015 README
-rw-r--r-- 1 mriedel zam    129 Jul  6 22:32 results.txt
-rw-r--r-- 1 mriedel zam  68836 Dec 14  2015 svm.cpp
-rw-r--r-- 1 mriedel zam    477 Dec 14  2015 svm.def
-rw-r--r-- 1 mriedel zam   3382 Dec 14  2015 svm.h
-rw-r--r-- 1 mriedel zam 180224 Jul  6 20:05 svm.o
-rwxr-xr-x 1 mriedel zam  78270 Jul  6 20:05 svm-predict
-rwxr-xr-x 1 mriedel zam    113 Jul  6 22:31 svm-predict1-3.sh
-rwxr-xr-x 1 mriedel zam    113 Jul  6 22:35 svm-predict2-3.sh
-rw-r--r-- 1 mriedel zam   5536 Dec 14  2015 svm-predict.c
-rwxr-xr-x 1 mriedel zam  18587 Jul  6 20:05 svm-scale
-rw-r--r-- 1 mriedel zam   8539 Dec 14  2015 svm-scale.c
drwxr-xr-x 5 mriedel zam    512 Dec 14  2015 svm-toy
-rwxr-xr-x 1 mriedel zam  79589 Jul  6 20:05 svm-train
-rwxr-xr-x 1 mriedel zam     76 Jul  6 22:24 svm-train1-3.sh
-rwxr-xr-x 1 mriedel zam     76 Jul  6 22:35 svm-train2-3.sh
-rw-r--r-- 1 mriedel zam   8985 Dec 14  2015 svm-train.c
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 tools
drwxr-xr-x 2 mriedel zam    512 Dec 14  2015 windows
```


Model File: non-linearly seperable case (iris-class2and3)

- Many SVs / sample
  - (careful – indicator for problems)
  - In the linear case we know from looking at the data it still be ok in this case
  - Linear SVM worked: PLA instead would not able to stop with this dataset

```
-bash-4.2$ more iris-class2and3-training.model
svm_type c_svc
kernel_type linear
nr_class 2
total_sv 27
rho -1.60294
label 2 3
nr_sv 14 13
SV
1 1:0.5 3:0.254237 4:0.0833333
1 1:0.166667 3:0.186441 4:0.166667
1 1:0.444444 2:-0.0833334 3:0.322034 4:0.166667
1 1:-0.333333 2:-0.75 3:0.8169491 4:-4.83573e-08
1 1:0.222222 2:-0.333333 3:0.220339 4:0.166667
1 1:-0.222222 2:-0.333333 3:0.186441 4:-4.83573e-08
1 1:0.111111 2:0.0833333 3:0.254237 4:0.25
1 1:0.277778 2:-0.25 3:0.220339 4:-4.83573e-08
0.306967888182512 1:-0.5 2:-0.416667 3:-0.8169491 4:0.0833333
1 1:-0.111111 2:-0.166667 3:0.0847457 4:0.166667
1 1:-1.32455e-07 2:-0.25 3:0.254237 4:0.0833333
0.327293858578756 1:0.333333 2:-0.0833334 3:0.152542 4:0.0833333
1 1:-0.277778 2:-0.166667 3:0.186441 4:0.166667
1 1:0.8555554 2:-0.833333 3:0.186441 4:0.166667
-1 1:-0.666667 2:-0.583333 3:0.186441 4:0.333333
-0.634268746701266 1:0.222222 3:0.38983 4:0.583333
-1 1:0.222222 2:-0.166667 3:0.525424 4:0.416667
-1 1:-0.8555556 2:-0.833333 3:0.355932 4:0.166667
-1 1:0.111111 2:-0.416667 3:0.322034 4:0.416667
-1 1:0.8555554 2:-0.333333 3:0.288136 4:0.416667
-1 1:-1.32455e-07 2:-0.166667 3:0.322034 4:0.416667
-1 1:0.611111 2:-0.166667 3:0.627119 4:0.25
-1 1:0.111111 2:-0.333333 3:0.38983 4:0.166667
-1 1:-1.32455e-07 2:-0.5 3:0.559322 4:0.0833333
-1 1:0.166667 2:-0.0833334 3:0.525424 4:0.416667
-1 1:-0.8555556 2:-0.166667 3:0.288136 4:0.416667
-1 1:-0.111111 2:-0.166667 3:0.38983 4:0.416667
```

- **Generalization measure: #SVs as 'in-sample quantity' → 10SVs/1000 samples ok, 500SVs/1000 bad**
- **Reasonsing towards overfitting due to a large number of SVs (fit many, small margin, gives bad E_{out})**

Testing Phase: non-linearly seperable case (iris-class2and3)

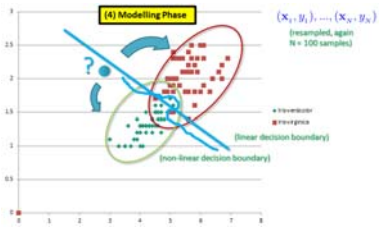
- Use svm-predict (using newly created model file & testing data)

```
-bash-4.2$ more svm-predict2-3.sh
./svm-predict /homeb/zam/mriedel/datasets/iris-class2and3-testing ./iris-class2and3-training.model ./results.txt
```

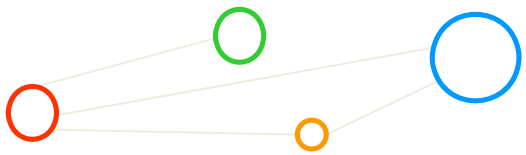
```
-bash-4.2$ ./svm-predict2-3.sh
Accuracy = 93.3333% (56/60) (classification)
```

```
-bash-4.2$ head results.txt
3
2
3
2
2
2
2
3
2
```

(consistent with our graph: ~4 data point will be misclassified by a linear decision boundary)



Lecture Bibliography



Lecture Bibliography

- [1] An Introduction to Statistical Learning with Applications in R, Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [2] LibSVM Webpage, Online: https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- [3] EUDATB2SHARE Iris Dataset LibSVM Format Preprocessing (Record 397), Online: http://hdl.handle.net/11304/10e216d4-0a98-4ab4-86ea-75ed05ee0f46
- [4] 'Simple-SVM question', Online: http://stats.stackexchange.com/questions/108617/simple-svm-question

Lecture 3 – Support Vector Machines

58 / 58