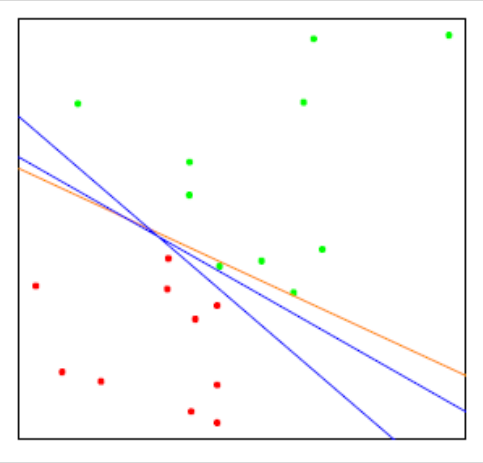


Parallel & Scalable Data Analysis

Introduction to Machine Learning Algorithms

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

LECTURE 5

Regularization and Support Vector Machines

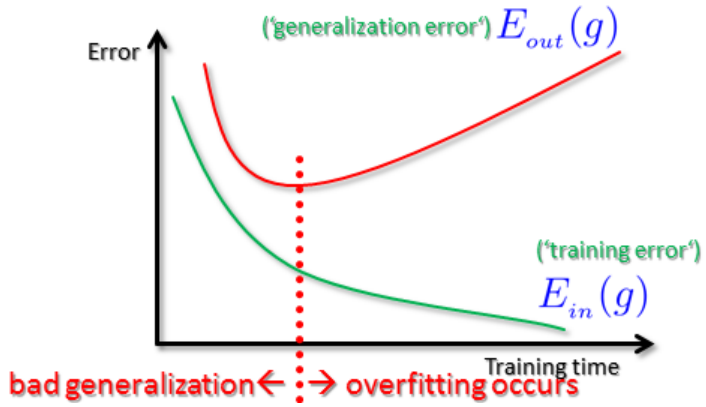
November 24th, 2017
Ghent, Belgium

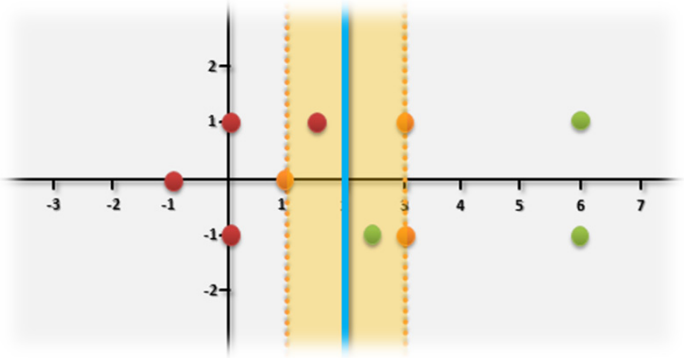
UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH
FORSCHUNGSZENTRUM

Review of Lecture 4

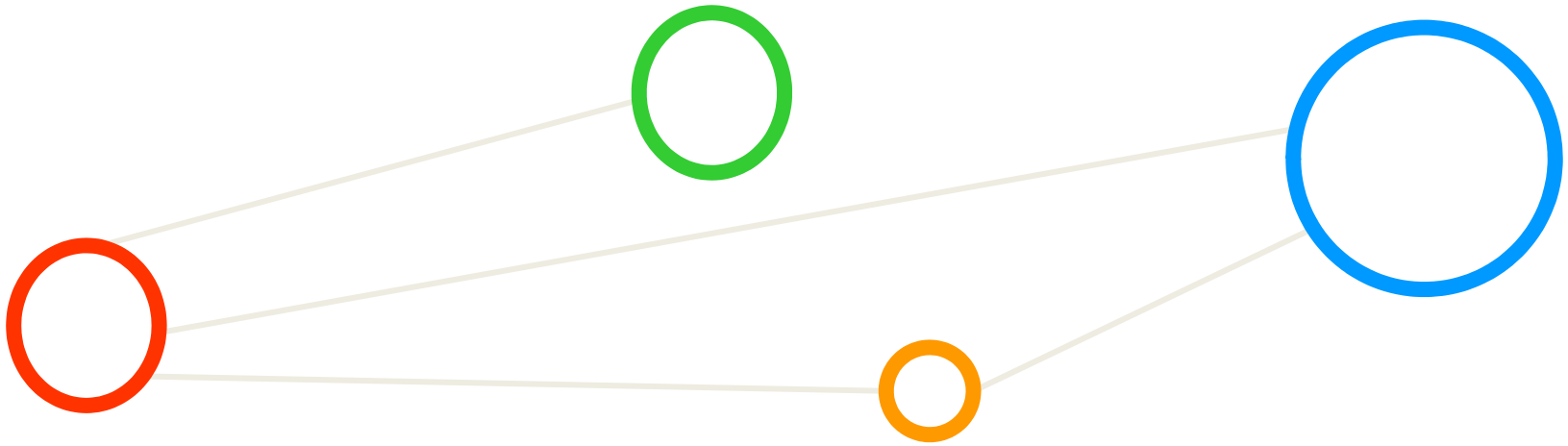
- Classification Challenges
  - Scalability, high dimensionality, etc.
  - Non-linearly seperable data
  - Overfitting
- Maximal Margin Classifier
  - Many possible lines exists
  - Maximal margin decision boundary with best generalization capabilities
  - Constraint optimization problem
  - Hard-margin (no errors allowed)





- LibSVM tool
  - Implements sequential minimal optimization (SMO) used in SVM training
  - svm-train and svm-predict with many parameters, e.g. C regularization

Outline



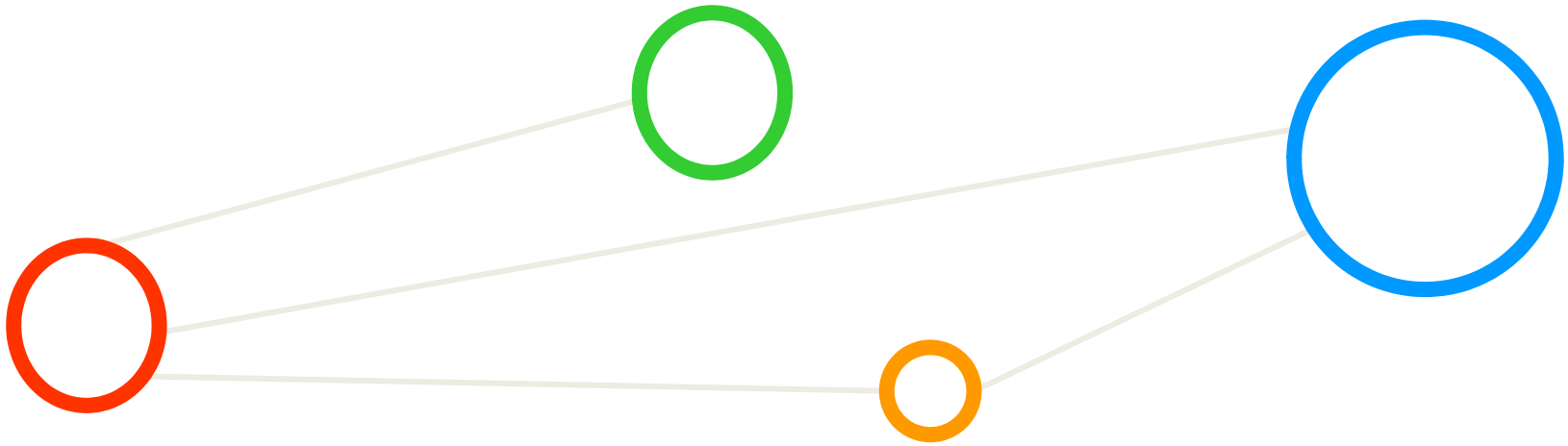
Outline of the Course

1. Machine Learning Fundamentals
2. Unsupervised Clustering and Applications
3. Supervised Classification and Applications
4. Classification Challenges and Solutions
5. Regularization and Support Vector Machines
6. Validation and Parallelization Benefits

Outline

- Regularization & Support Vector Classifier
  - Regularization Methods against Overfitting
  - From Hard-margin to Soft-margin
  - Role of Regularization Parameter C
  - Solving and Limitations of Classifier
  - Apply Classifier to Flower Problem
- Non-linear Transformations
  - Need for Non-Linear Decision Boundaries
  - Mapping Data to higher dimensions
  - Role of Linear Boundary in Feature Space
- Kernel Methods
  - Full Support Vector Machines & Kernel Trick
  - Polynomial and Radial Basis Function Kernel
  - Apply Kernels to Remote Sensing Data
  - Multi-Class Classification Approaches

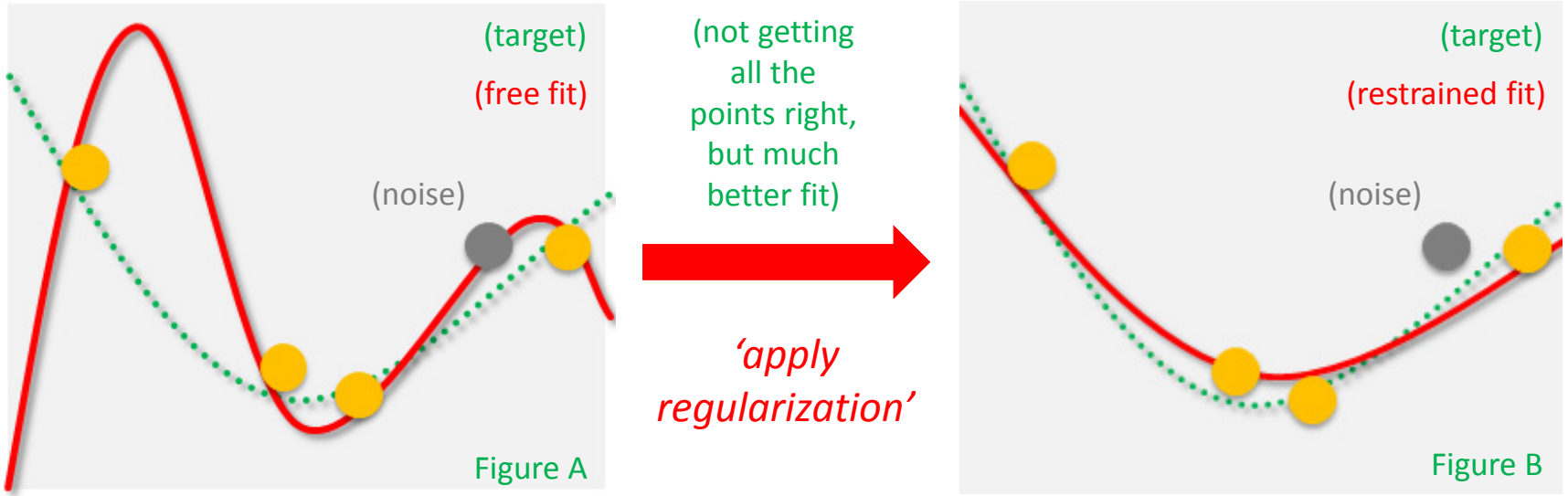
Support Vector Classifier



Regularization – Key Principle

- Initial setup
  - Considered as ‘free fit‘ – ‘fit as far as the model can do‘
  - E.g. use 4th order polynomial model and fit the 5 data points (cf. Figure A)
- ‘Putting the brakes‘ regularization to avoid overfitting
  - Apply a ‘restrained fit‘ – ‘preventing to fit the data perfectly‘
  - E.g. use 4th order polynomial model but use ‘minimal brakes‘ (cf. Figure B)

(equivalent meaning of explicitly forbidding some of the hypothesis to be considered in learning)



Exercises

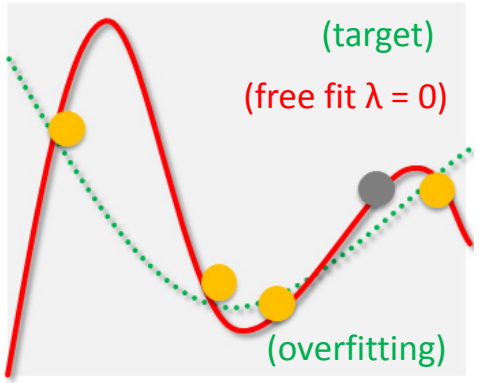
Regularization – Two Approaches

- Used in almost every machine learning situation
  - Two approaches to regularization: mathematics & heuristics
- Mathematics (from ill-posed problems in function approximation)
  - Goal: approximate a function
  - 'Ill-posed Problem': there are many functions that fit this function
  - Approach: Impose 'smoothness constraints' to solve the problem
  - Learning practice: assumptions in mathematical approach not realistic
  - Solution: use mathematical approach intuitively to create 'rules of thumb'
- Heuristics
  - Handy-capping the minimization of the in-sample-error
  - Idea of 'putting the brakes' – partly based on mathemat $E_{in}(g)$
  - Not a random process and oriented towards preventing overfitting

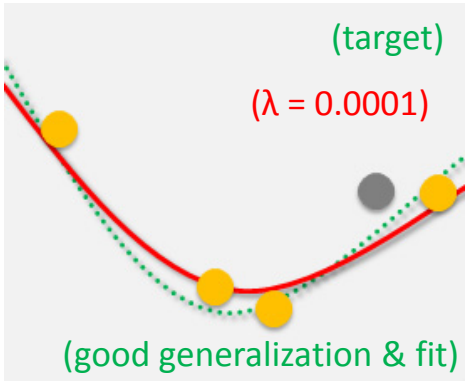
- **Regularization creates simpler models & thus in-line with 'Occams Razor' (simple model better)**

Regularization – Regularization Example & Impact

- Example: Minimize $E_{in}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$ (penalizing models with extreme parameter values)

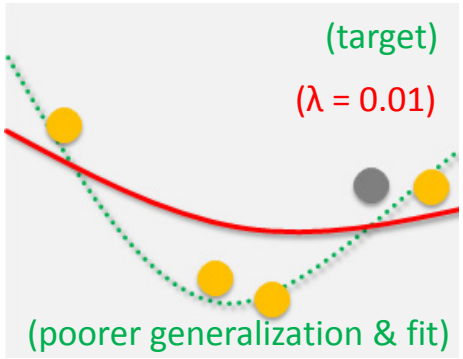


'apply little regularization'

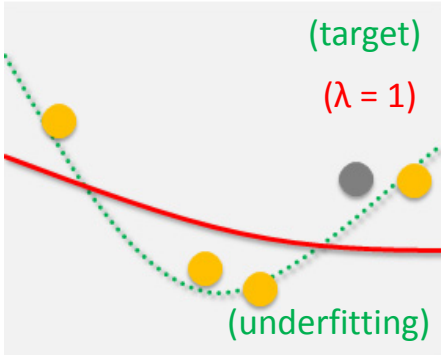


(regularization coefficient λ controls tradeoff: how much it is important to fit the data well vs. simple hypothesis)

'apply more regularization'



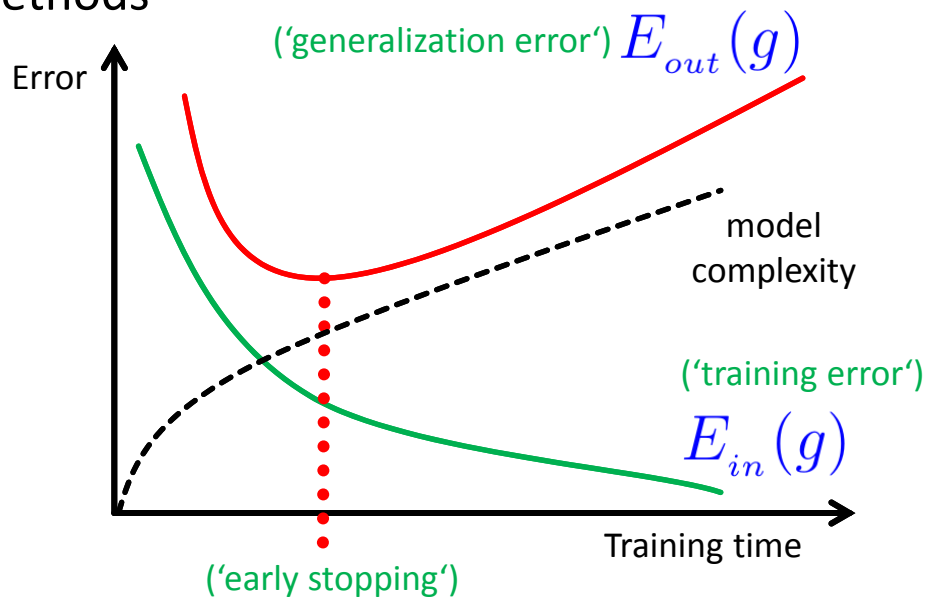
'apply too much regularization'



- **Choice of λ is extremely critical and can lead with high values to underfitting (as bad as overfitting)**
- **Choice of type of regularizers is heuristic, but choice of λ will be extremely principled (validation)**

Regularizer Methods – Early Stopping & Big Data

- Idea: regularization through the optimizer
  - Apply early stopping algorithm as part of the 'training time'
  - Based on practical validation methods
  - Point in time to stop will be choosen by validation



- **Big data has a bigger dataset N and more likely also 'more stochastic noise'**
- **Preventing overfitting with big data tends to require stricter regularization**
- **Early stopping offers a pragmatic way for big data to prevent overfitting**
- **Given large quantities of dataset N, the training time is massively reduced too**

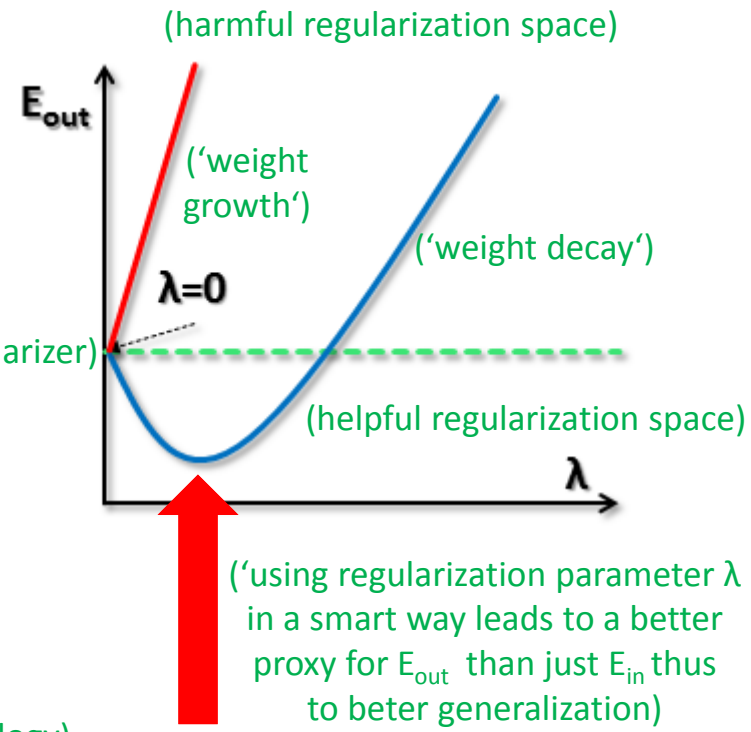
➢ **Lecture 6 provides details on how validation is used to determine the early stopping point in time**

Regularization Methods – Choosing Guidelines

- **Choosing a regularizer in learning from data is a must have & leads to better generalization**
- **Choosing not a regularizer will definitely lead to overfitting the data in practical situations**

- Selected ‘rules of thumb‘
  - Choosing constraints with ‘weight growth‘ gives terrible results
  - Choosing constraints towards ‘smoother hypothesis‘ gives good results
- Reasoning noise is not smooth
  - Stochastic noise → ‘high frequency‘
  - Determinisic noise → ‘non-smooth‘
  - Smaller weights correspond to smoother hypothesis

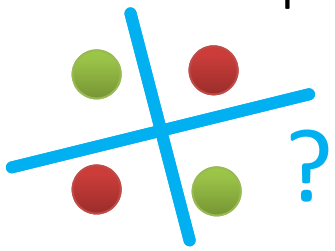
(‘hectic dog & calm man have a walk‘ analogy)

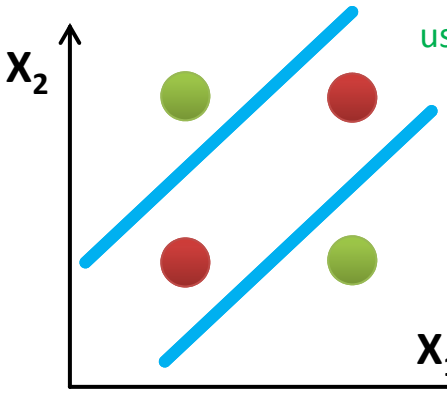


- **The rule of thumb in regularization is to constrain the learning with smoother/simpler hypothesis**
- **Regularization forbids some hypothesis & thus reduces the VC dimension: improving generalization**

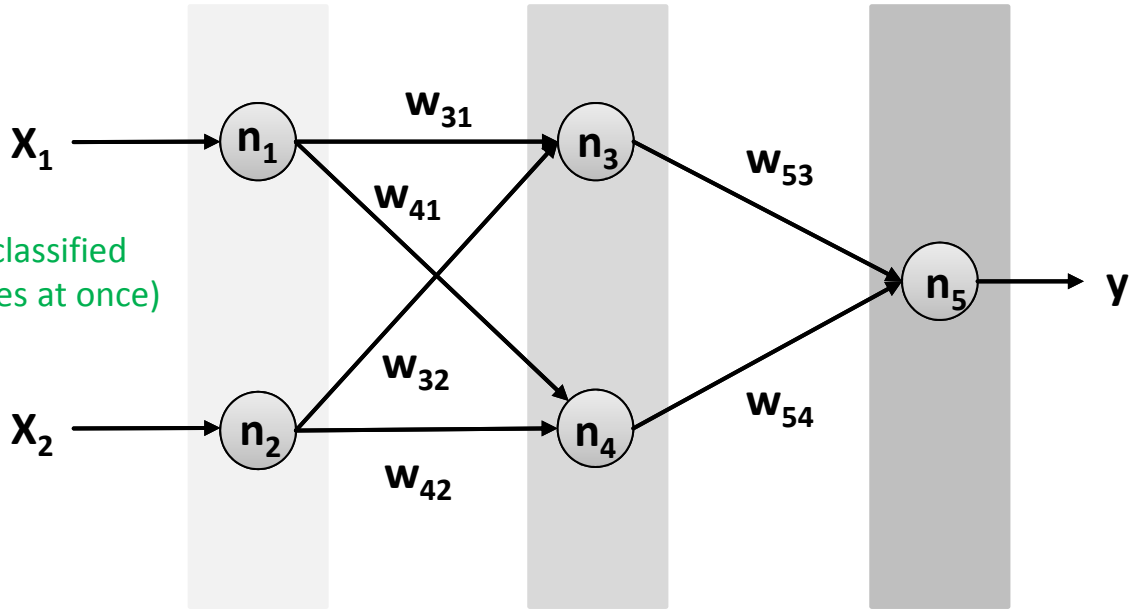
Other Models – Artificial Neural Networks (ANNs)

- ANNs are one of the most powerful classification methods
  - Often very much optimized for domain-specific solutions
  - Idea: Linear models work but are limited (e.g. simple XOR example)
  - Example: 'how we can combine layers of perceptrons to learn from data?'





(instances can be classified using two hyperplanes at once)

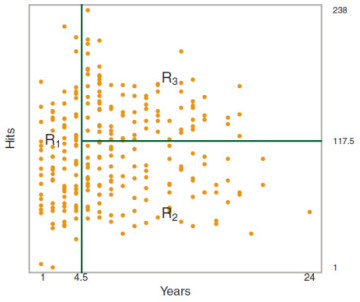


(regularization method 'weight decay is used in neural networks')

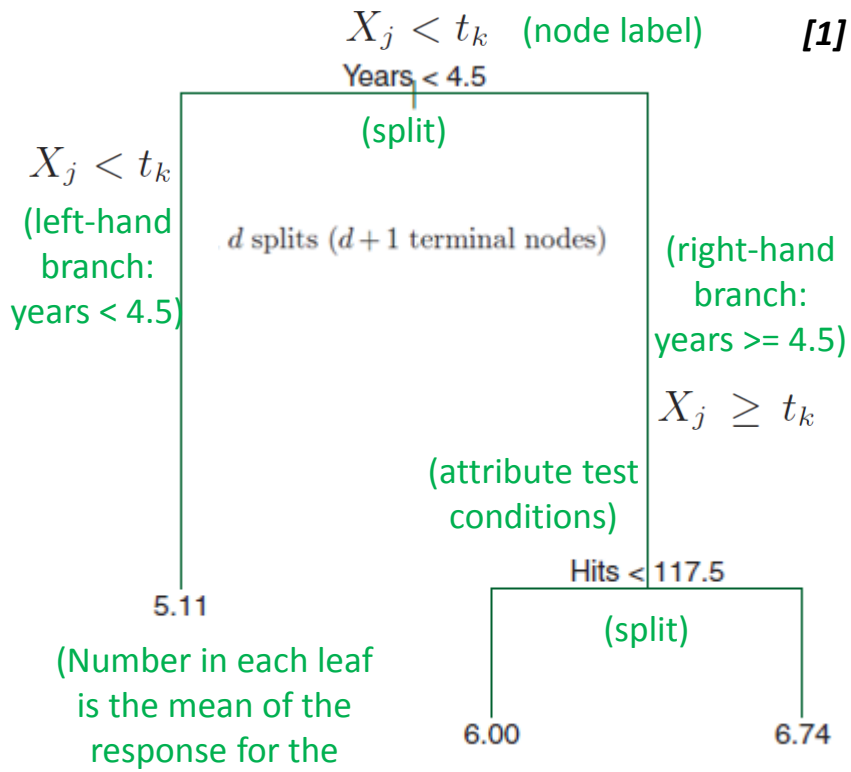
➢ **Artificial Neural Networks can be considered as Multi-Layered Perceptron: tutorial next week!**

Other Models – Random Forests & Decision Trees

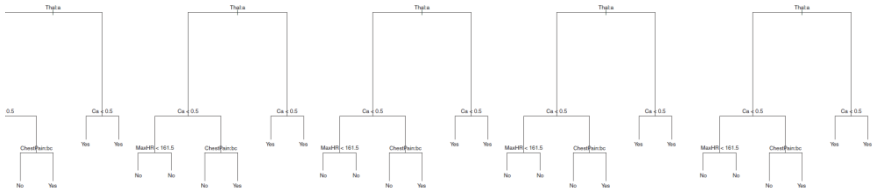
- Decision Trees and Random Forests often used
  - Decision boundaries can be interpreted as regions



[1] An Introduction to Statistical Learning

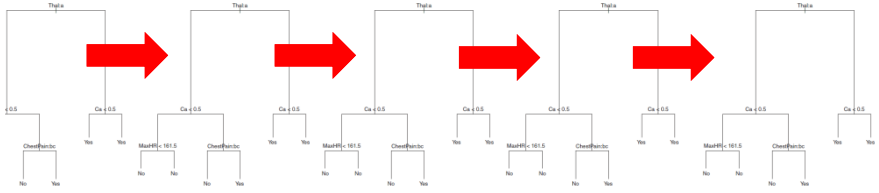


(Random Forest with bagging)



nstruct multiple trees in parallel, each on a sample of the data)

(Random Forest with boosting)



(construct multiple trees in a series, each on a sample of the data)

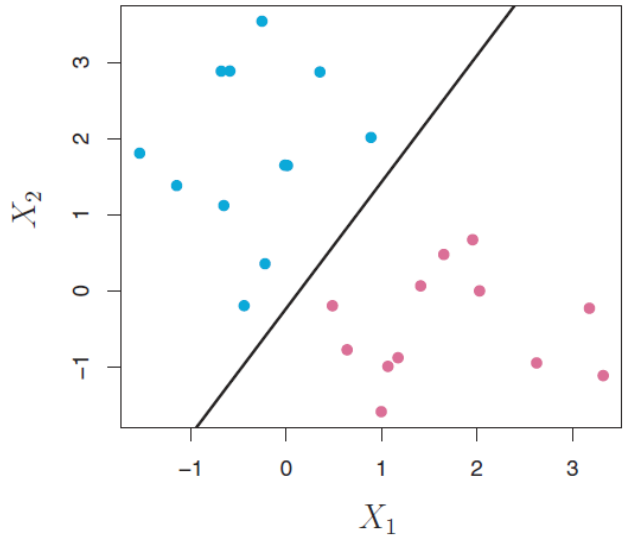
➢ **Random Forests are ensemble methods of Decision Trees and also often used in practice today**

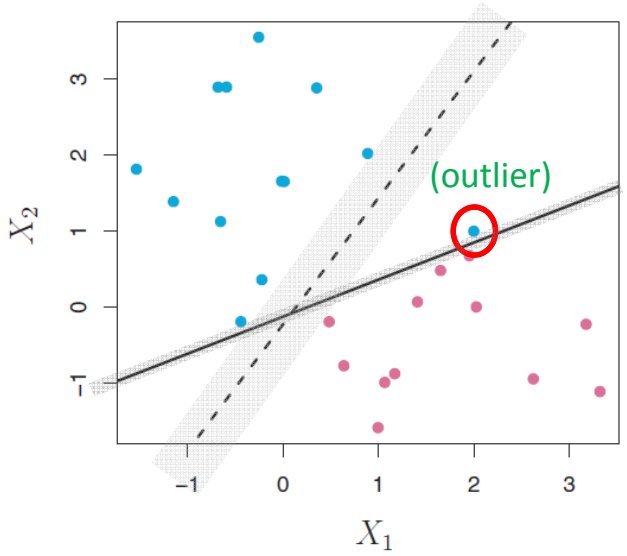
Support Vector Classifiers – Motivation

- **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





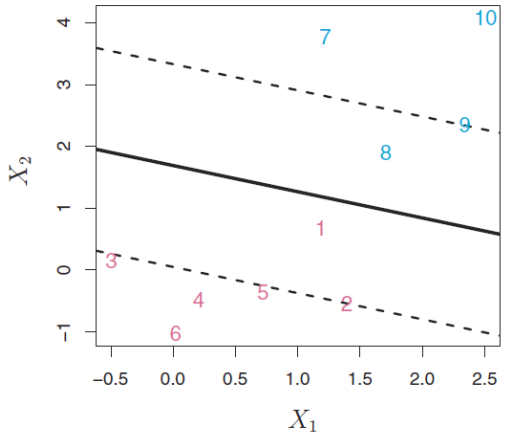
(significant reduction of the maximal margin)

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

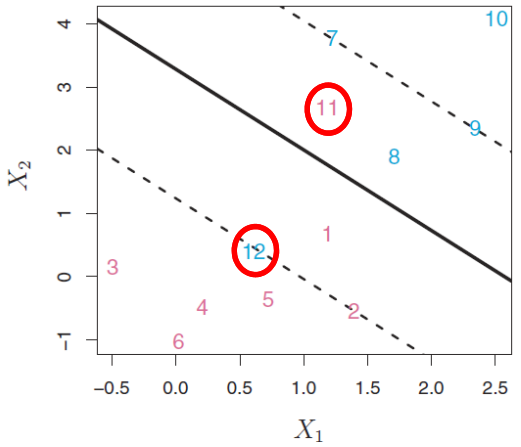
Support Vector Classifiers – Modified Technique Required

- 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)

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

(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 $\epsilon_1, \ldots, \epsilon_n$

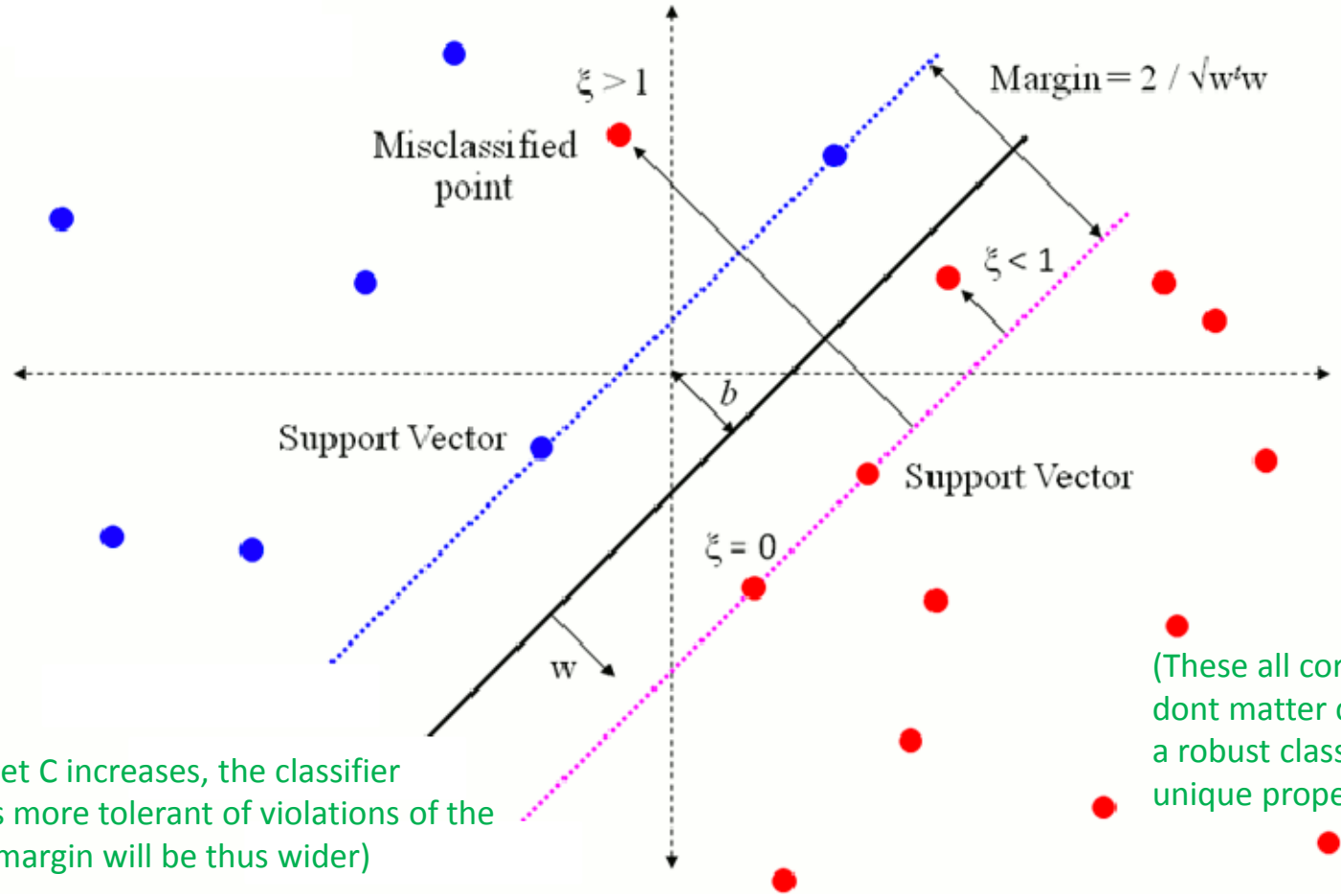
[1] An Introduction to Statistical Learning

- **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 6 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)

Support Vector Classifier – Impact of Regularization with C

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

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

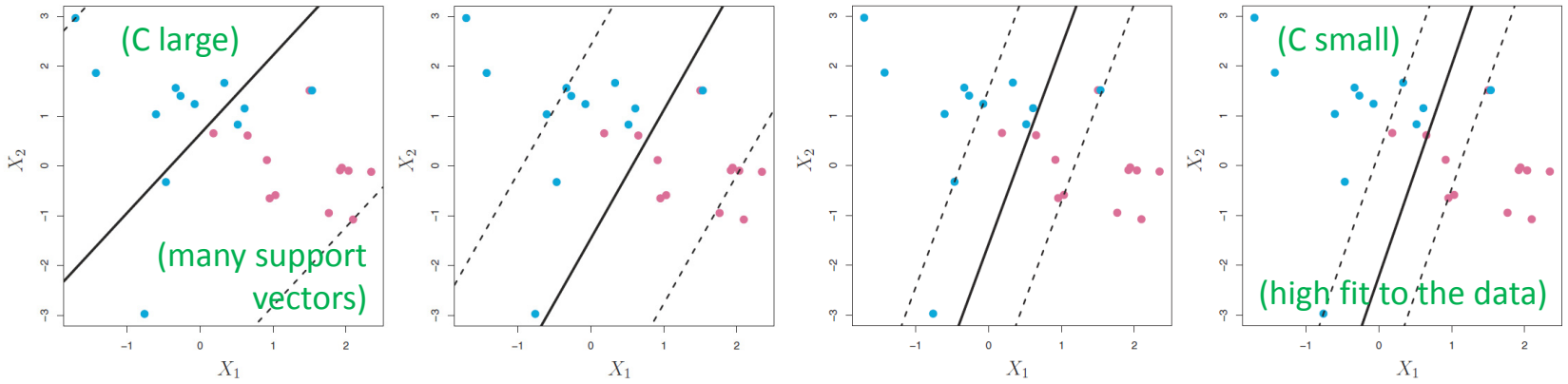
(maximal margin classifier)

- 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)

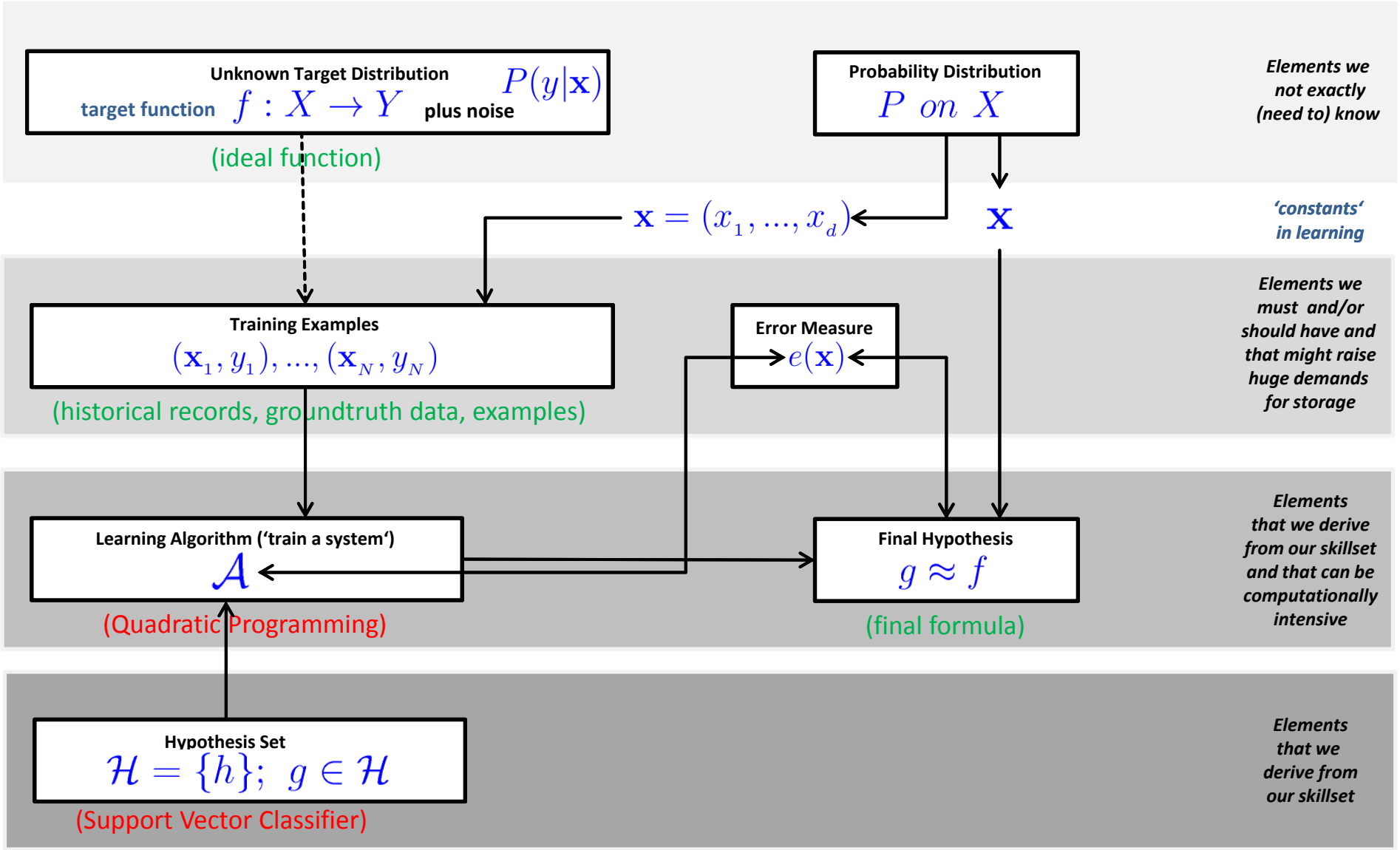
[1] An Introduction to Statistical Learning



(fewer support Vectors: just #8)

➢ **Lecture 6 provides details on validation methods that provides a value for C in a principled way**

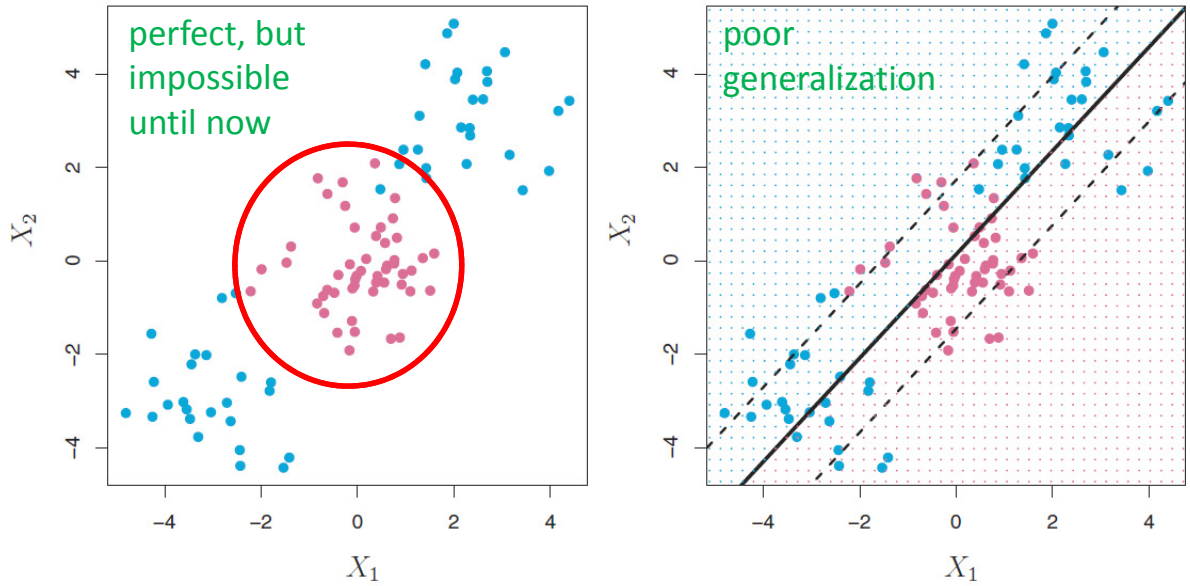
Solution Tools: Support Vector Classifier & QP Algorithm



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
```

- Check model file
  - Next page, because many support vectors!

```
-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  83089 Dec 14  2015 FAQ.html
-rw-r--r-- 1 mriedel zam  27670 Dec 14  2015 heart_scale
-rw-r--r-- 1 mriedel zam    354 Jul  6 22:25 iris-class1and3-training.model
-rw-r--r-- 1 mriedel zam   1430 Jul  6 22:36 iris-class2and3-training.model
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    120 Jul  6 22:32 results.txt
-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 100224 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  78509 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   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
```

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 -1.60284
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 3:0.0169491 4:-4.03573e-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:-4.03573e-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:-4.03573e-08
0.3069670881822512 1:-0.5 2:-0.416667 3:-0.0169491 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
0.3272936585778756 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 1:0.0555554 2:-0.833333 3:0.186441 4:0.166667
-1 1:-0.666667 2:-0.583333 3:0.186441 4:0.333333
-0.6342607467601266 1:0.222222 3:0.38983 4:0.583333
-1 1:0.222222 2:-0.166667 3:0.525424 4:0.416667
-1 1:-0.0555556 2:-0.833333 3:0.355932 4:0.166667
-1 1:0.111111 2:-0.416667 3:0.322034 4:0.416667
-1 1:0.0555554 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:-0.0555556 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)

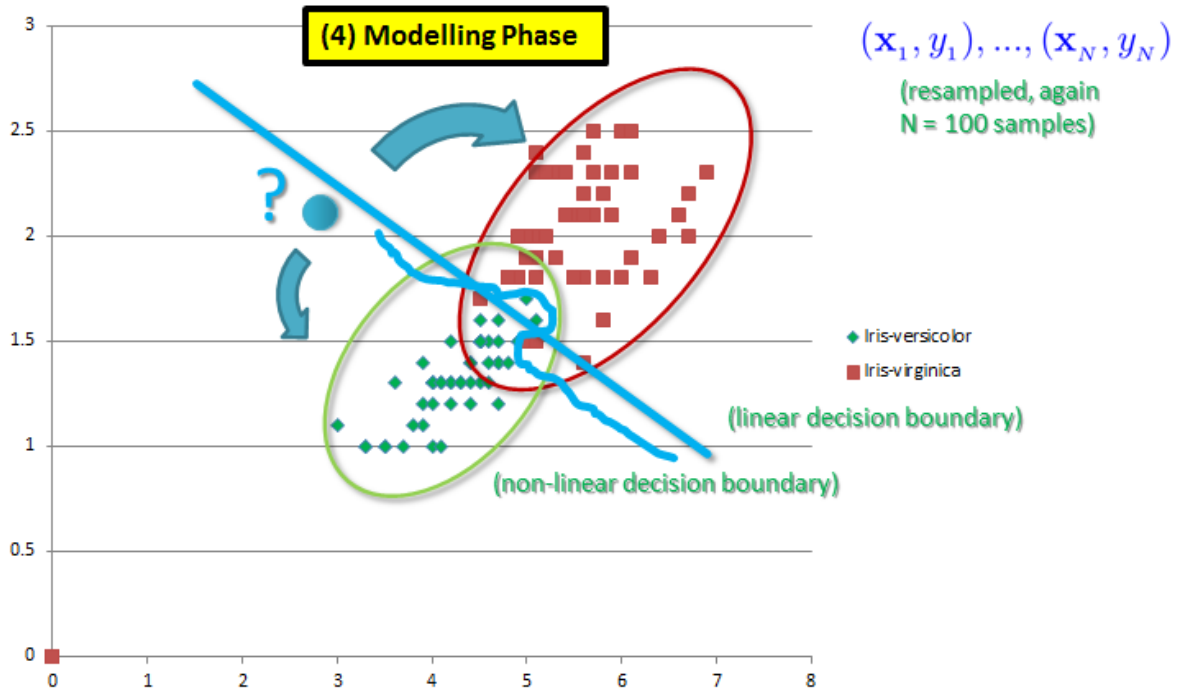
- Use svm-predict (using newly created model file & testing data)

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

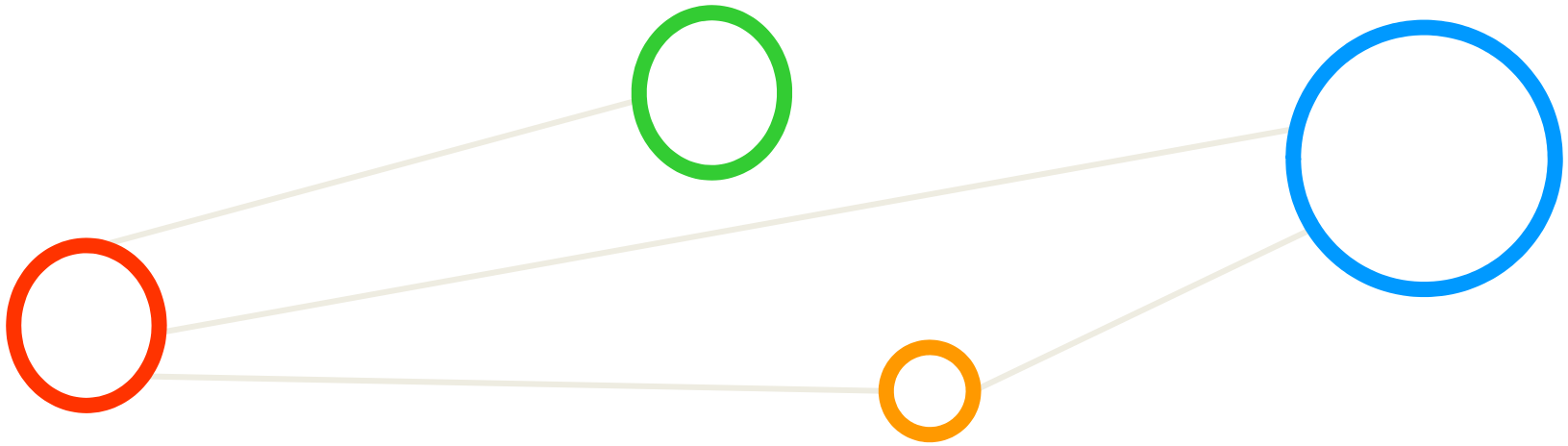
```
-bash-4.2$ ./svm-predict2-3.sh
Accuracy = 93.3333% (56/60) (classification)
```

```
-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)

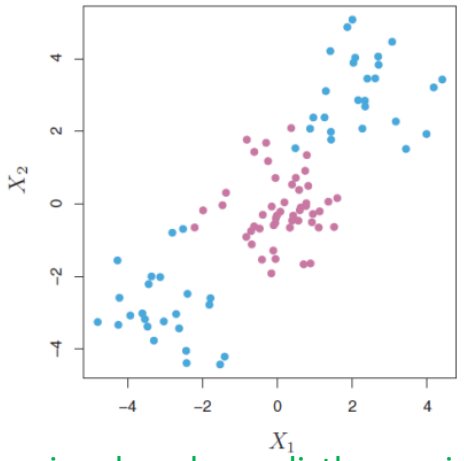


Non-Linear Transformations



Need for Non-linear Decision Boundaries

- Lessons learned from practice
  - Scientists and engineers are often faced with non-linear class boundaries
- Non-linear transformations approach
  - Enlarge feature space (computationally intensive)
  - Use quadratic, cubic, or higher-order polynomial functions of the predictors
- Example with Support Vector Classifier



(time invest: mapping done by explictly carrying out the map into the feature space)

$X_1, X_2, \ldots, X_p$ (previously used p features)

$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$ (new 2p features)

(decision boundary is linear in the enlarged feature space)

(decision boundary is non-linear in the original feature space with q(x) = 0 where q is a quadratic polynomial)

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

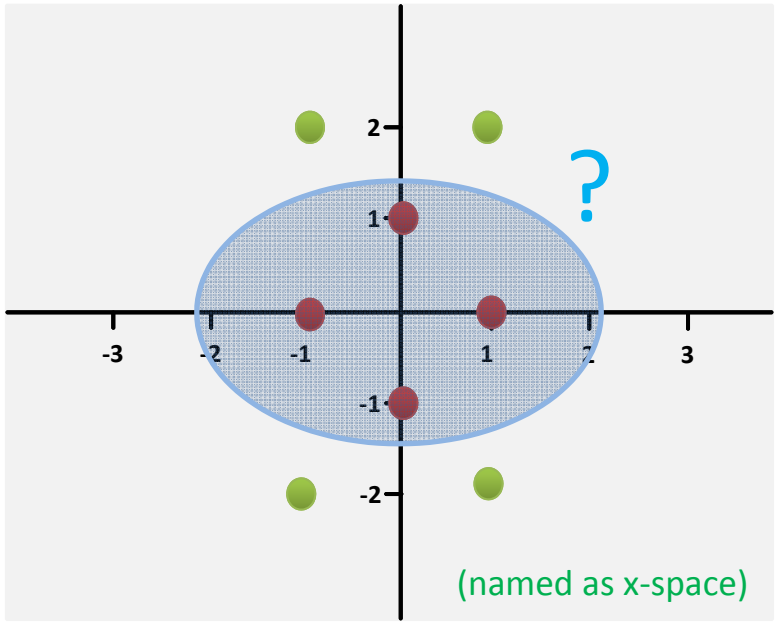
$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \geq M(1-\epsilon_i)$$

$$\sum_{i=1}^{n}\epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1$$

[1] An Introduction to Statistical Learning

Understanding Non-Linear Transformations (1)

- Example: ‘Use measure of distances from the origin/centre‘
- Classification (still linear models applicable)
  - (1) new point; (2) transform to z-space; (3) classify it with e.g. perceptron

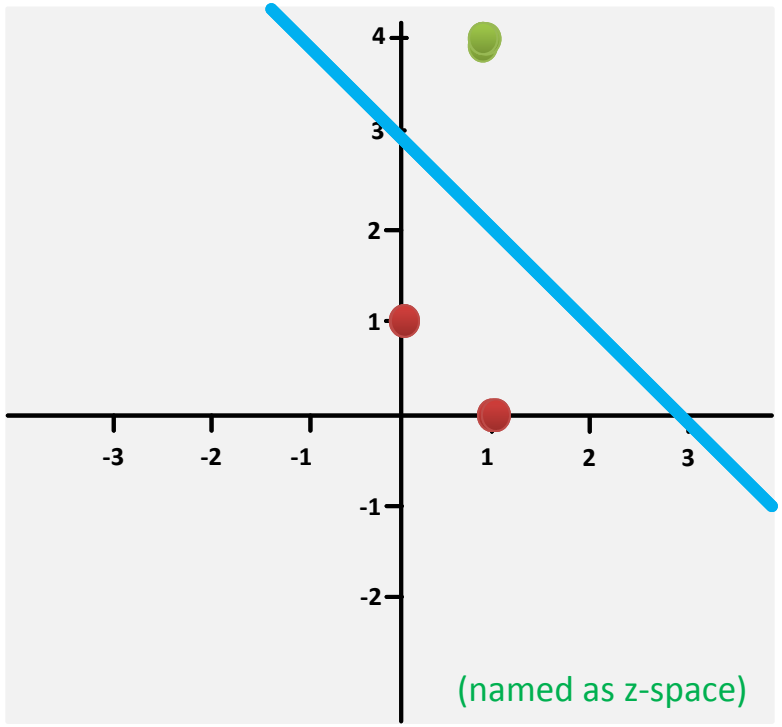


(named as x-space)

(also called input space)

Φ

(‘changing constants‘)

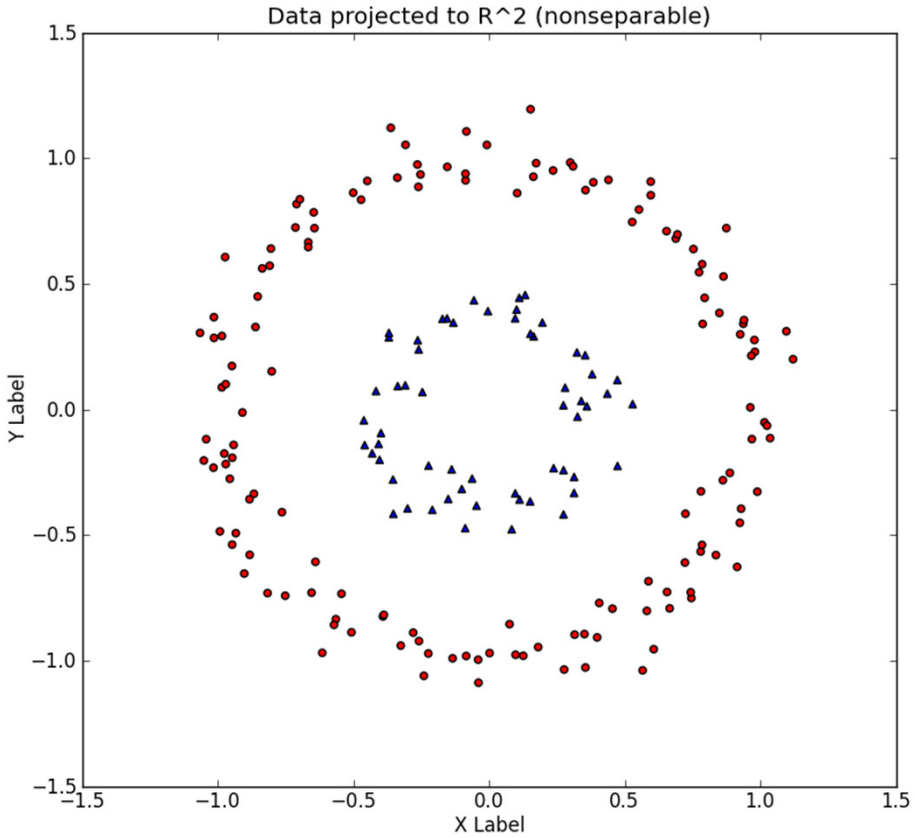


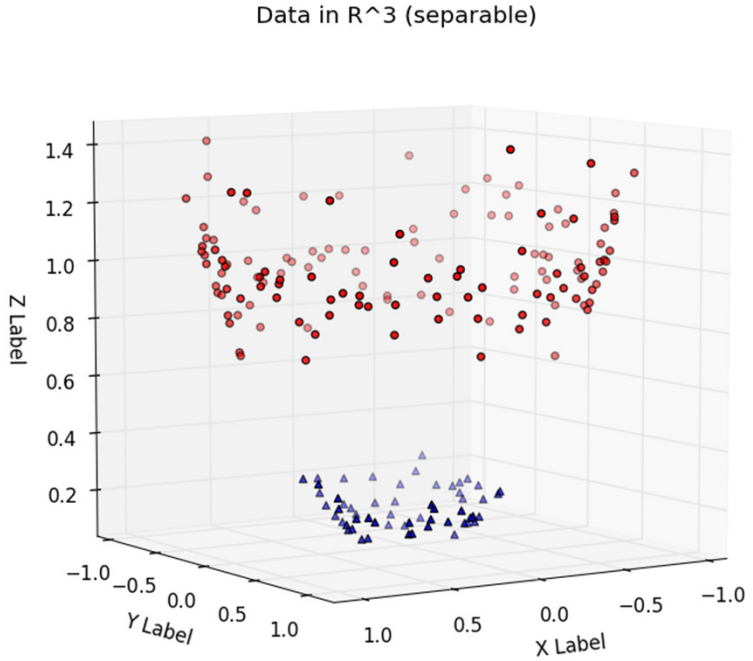
(named as z-space)

(also called feature space)

Understanding Non-Linear Transformations (2)

- Example: From 2 dimensional to 3 dimensional: $[x_1, x_2] = [x_1, x_2, {x_1}^2 + {x_2}^2]$
  - Much higher dimensional can cause memory and computing problems



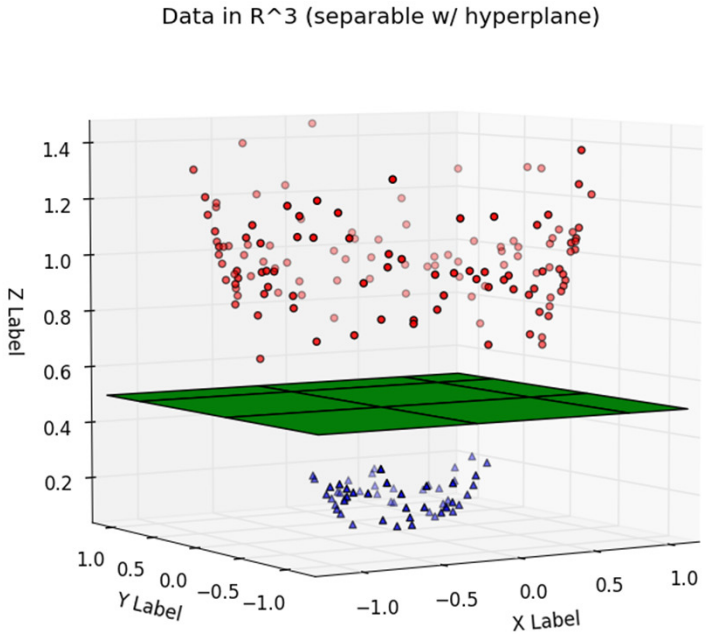


[2] E. Kim

- **Problems: Not clear which type of mapping (search); optimization is computationally expensive task**

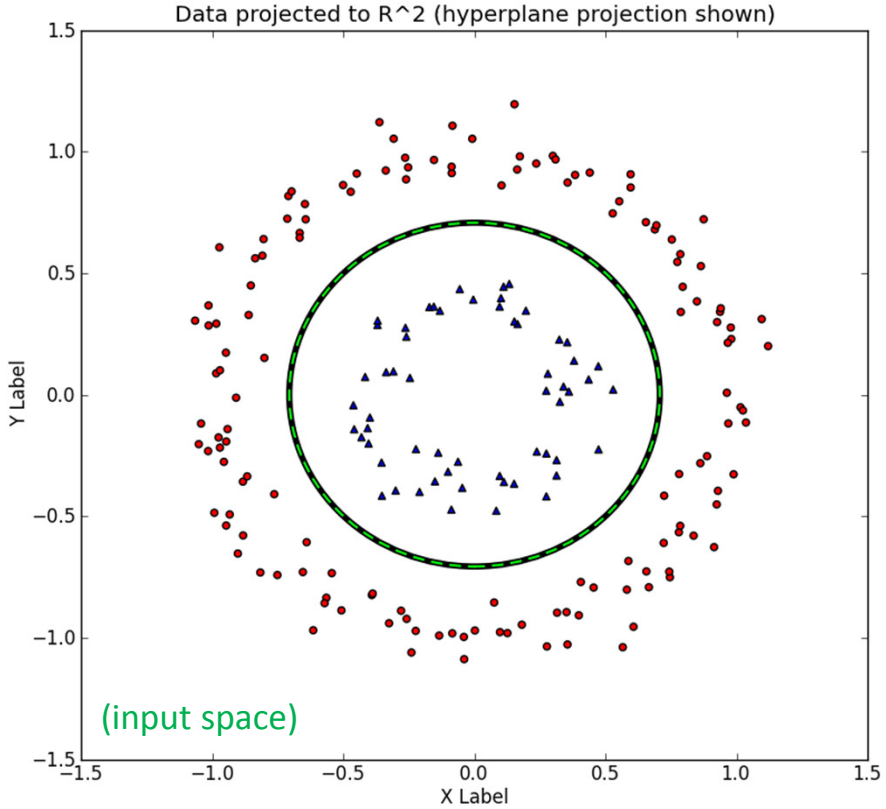
Understanding Non-linear Transformations (3)

- Example: From 2 dimensional to 3 dimensional: $[x_1, x_2] = [x_1, x_2, {x_1}^2 + {x_2}^2]$
  - Separating hyperplane can be found and 'mapped back' to input space



(feature space)

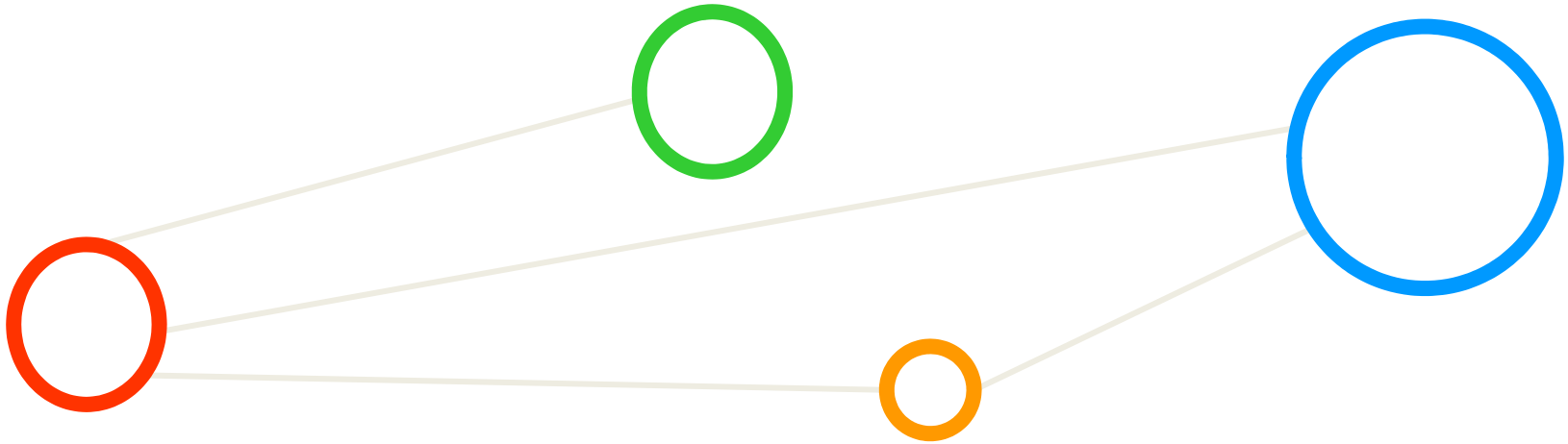
[2] E. Kim



(input space)

- **Problem: 'curse of dimensionality' – As dimensionality increases & volume of space too: sparse data!**

Kernel Methods



Term Support Vector Machines – Revisited

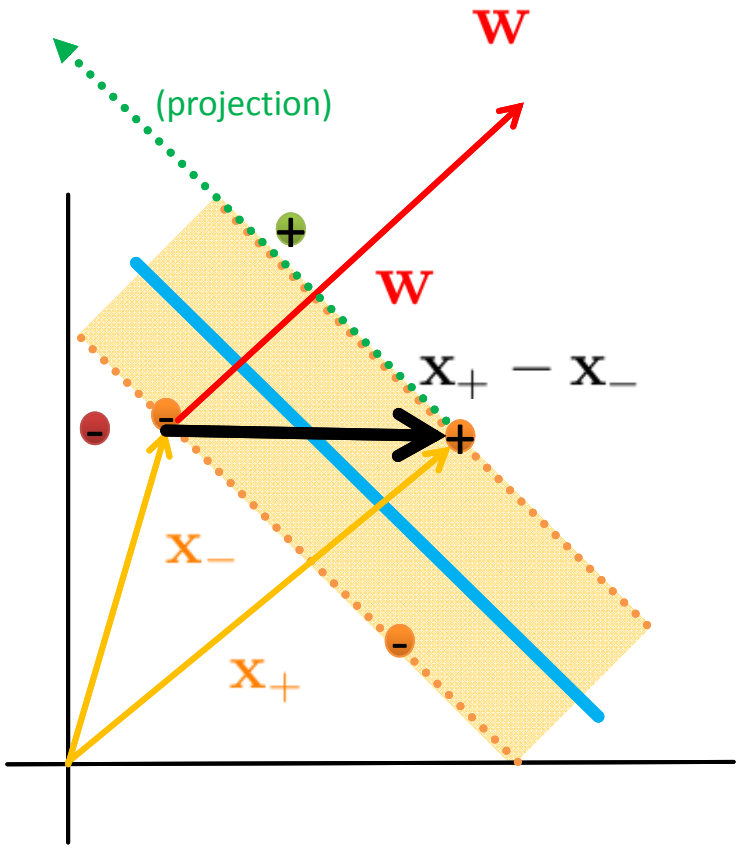
- **Support Vector Machines (SVMs) are a classification technique developed ~1990**
- **SVMs perform well in many settings & are considered as one of the best ‘out of the box classifiers‘**

[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;

Constrained Optimization Steps SVM & Dot Product

- 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)



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

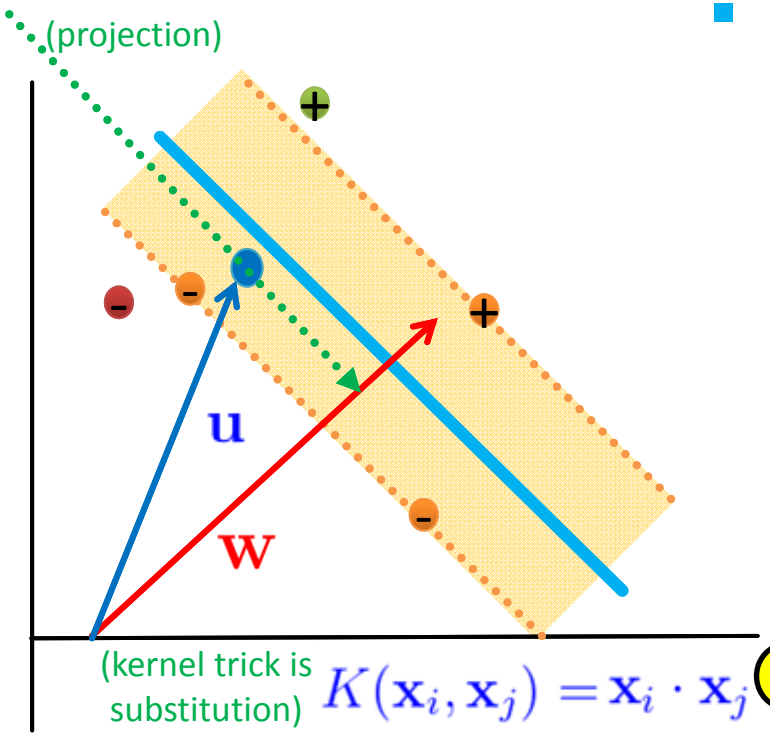
Kernel Methods & Dot Product Dependency

- 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)



- Dotproduct enables nice more elements
  - E.g. consider non linearly seperable data
  - Perform non-linear transformation Φ of the samples into another space (work on features)

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

(optimization depends only on dot product of samples)

$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ (in optimization)

$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{u}_i)$ (for decision rule above too)

(kernel trick is substitution) $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$

(7) $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$

(trusted Kernel avoids to know Phi)

Support Vector Machines & Kernel Methods

- **Support Vector Machines are extensions of the support vector classifier using kernel methods**
- **Support Vector Machines enable non-linear decision boundaries that can be efficiently computed**
- **Support Vector Machines avoids 'curse of dimensionality' and mapping search using a 'kernel trick'**

- Non-linear transformations

[1] An Introduction to Statistical Learning

  - Lead to high number of features → Computations become unmanageable
    (including the danger to run into 'curse of dimensionality')
- Benefits with SVMs
  - Enlarge feature space using 'kernel trick' → ensures efficient computations
  - Map training data into a higher-dimensional feature space using Φ
  - Create a seperating 'hyperplane' with maximum margin in feature space
- Solve constraint optimization problem
  - Using Lagrange multipliers & quadratic programming (cf. earlier classifiers)
  - Solution involves the inner products of the data points (dot products)
  - Inner product of two r-vectors a and b is defined as $\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$
  - Inner product of two data points: $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$

Linear SV Classifier Refined & Role of SVs

- Linear support vector classifier
  - Details w.r.t. inner products
  - With n parameters $\alpha_i, \quad i = 1, \ldots, n$ (Lagrange multipliers)
  - Use training set to estimate parameters
  - Estimate $\alpha_1, \ldots, \alpha_n$ and β_0 using $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

(between all pairs of training data points)

n (n – 1) / 2 number of pairs

- Evaluate $f(x)$ with a new point
  - Compute the inner product between new point x and each of the training points x_i
- Identify support vectors → Quadratic programming
  - α_i is zero most of the times (identified as not support vectors)
  - α_i is nonzero several times (identified as the support vectors)

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

'big data' reduction & less computing

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

(S with indices of support vectors)

[1] An Introduction to Statistical Learning

The ('Trusted') Kernel Trick

- Summary for computation
  - All that is needed to compute coefficients are inner products
- Kernel Trick
  - Replace the inner product with a generalization of the inner product
  - K is some kernel function
- Kernel types
  - Linear kernel
  - Polynomial kernel

(sompute the hyperplane without explictly carrying out the map into the feature space)

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

(inner product used before)

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

(kernel ~ distance measure)

$$K(x_i, x_{i'})$$

(choosing a specific kernel type)

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$ (linear in features)

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$ (polynomial of degree d)

[1] An Introduction to Statistical Learning

- **Kernel trick refers to a mechanism of using different kernel functions (e.g. polynomial)**
- **A kernel is a function that quantifies the similarity of two data points (e.g. close to each other)**

Kernel Trick – Example

- Consider again a simple two dimensional dataset
  - We found an ideal mapping Φ after long search
  - Then we need to transform the whole dataset according to Φ

$$\Phi : (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

  - Instead, with the 'kernel trick' we 'wait' and 'let the kernel do the job':

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \Rightarrow min_\alpha \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum \alpha_i$$

(no need to compute the mapping already)

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1)$$

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1$$

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2 = K(\mathbf{u}, \mathbf{v})$$

(we can save computing time by do not perform the mapping)

(in transformed space still a dot product in the original space → no mapping needed)

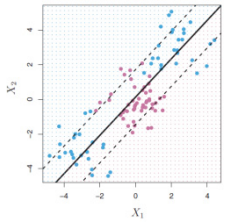
- **Example shows that the dot product in the transformed space can be expressed in terms of a similarity function in the original space (here dot product is a similiarity between two vectors)**

Linear vs. Polynomial Kernel Example

- Linear kernel
  - Enables linear decision boundaries (i.e. like linear support vector classifier)

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$ (linear in features)

(observed useless for non-linear data)

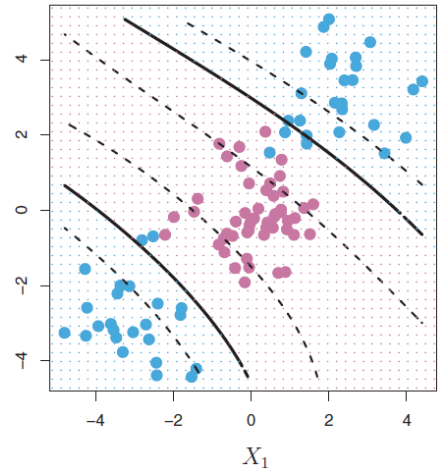


- Polynomial kernel
  - Satisfy Mercer's theorem = trusted kernel
  - Enables non-linear decision boundaries (when choosing degree d > 1)
  - Amounts to fit a support vector classifier in a higher-dimensional space
  - Using polynomials of degree d (d=1 linear support vector classifier)

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$ (polynomial of degree d)

(SVM with polynomial kernel of degree 3)

(significantly improved decision rule due to much more flexible decision boundary)

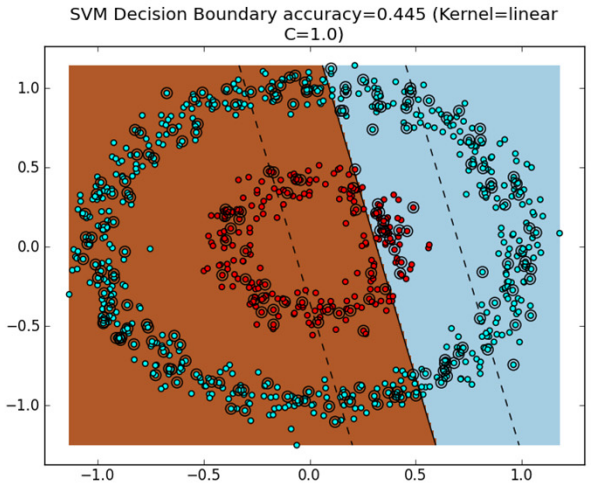


[1] An Introduction to Statistical Learning

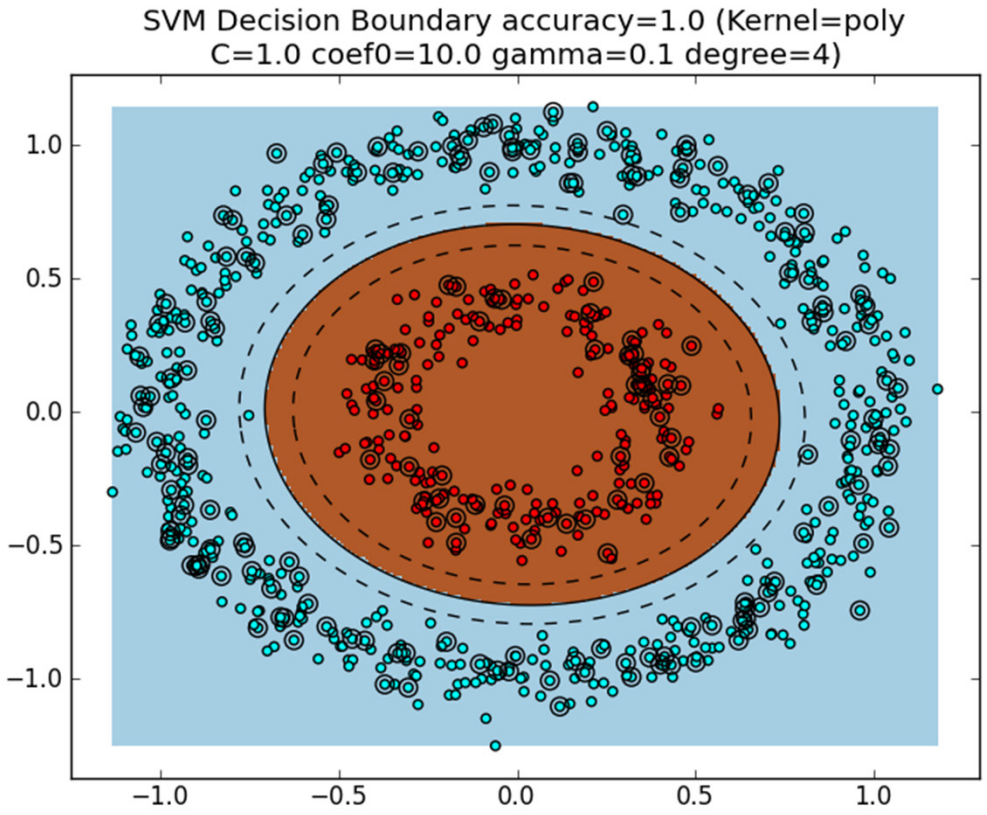
- **Polynomial kernel applied to non-linear data is an improvement over linear support vector classifiers**

Polynomial Kernel Example

- Circled data points are from the test set



$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$



[2] E. Kim

RBF Kernel

- Radial Basis Function (RBF) kernel (also known as radial kernel)
  - One of the mostly used kernel function
  - Uses parameter γ as positive constant

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

- 'Local Behaviour functionality'
  - Related to Euclidean distance measure (ruler distance)

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

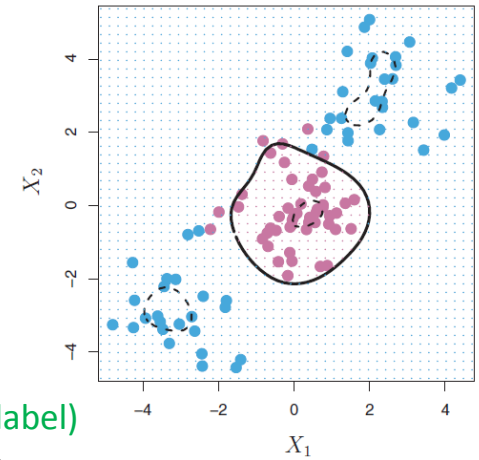
- Example
  - Use test data $x^* = (x_1^* \ldots x_p^*)^T$
  - Euclidean distance gives x^* far from x_i

$\sum_{j=1}^{p} (x_j^* - x_{ij})^2$ (large value with large distance)

$K(x^*, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_j^* - x_{ij})^2)$ (tiny value)

$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$ (training data x_i plays no role for x* & its class label)

(SVM with radial kernel)



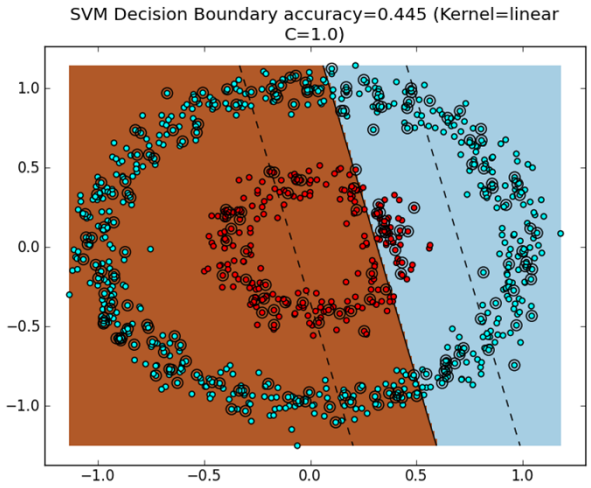
[1] An Introduction to Statistical Learning

- **RBF kernel have local behaviour (only nearby training data points have an effect on the class label**

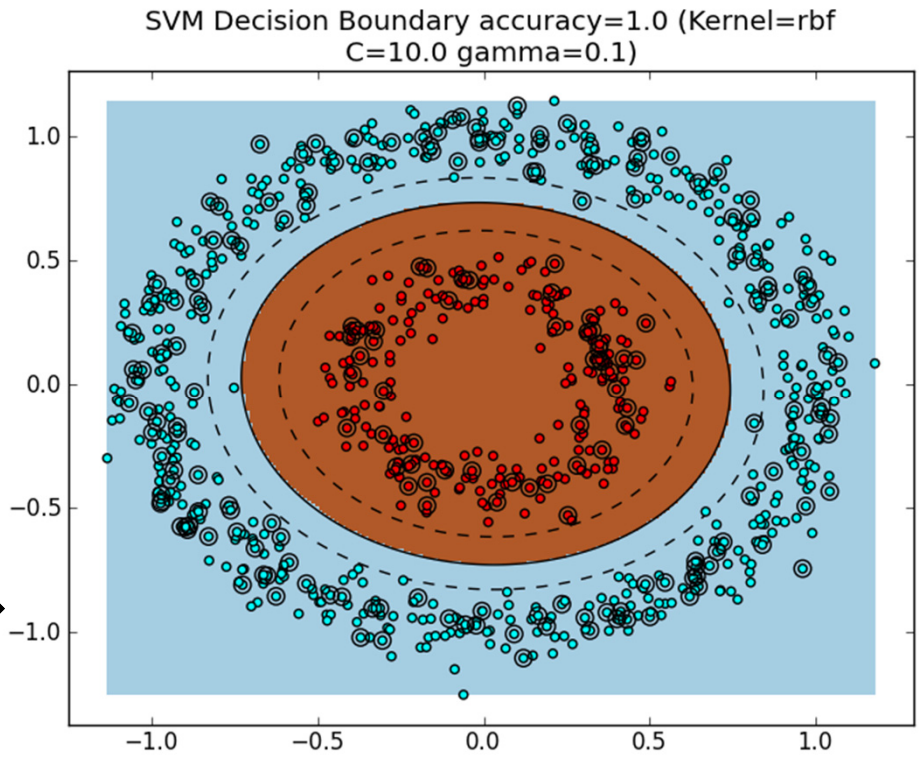
RBF Kernel Example

- Circled data points are from the test set

(similiar decision boundary as polynomial kernel)



$$K(x^*, x_i) = \exp(-\gamma \sum_{j=1}^{p} (x_j^* - x_{ij})^2)$$



[2] E. Kim

Exact SVM Definition using non-linear Kernels

- **True Support Vector Machines are Support Vector Classifiers combined with a non-linear kernel**
- **There are many non-linear kernels, but mostly known are polynomial and RBF kernels**

[1] An Introduction to Statistical Learning

- General form of SVM classifier
  - Assuming non-linear kernel function K
  - Based on 'smaller' collection S of SVs

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

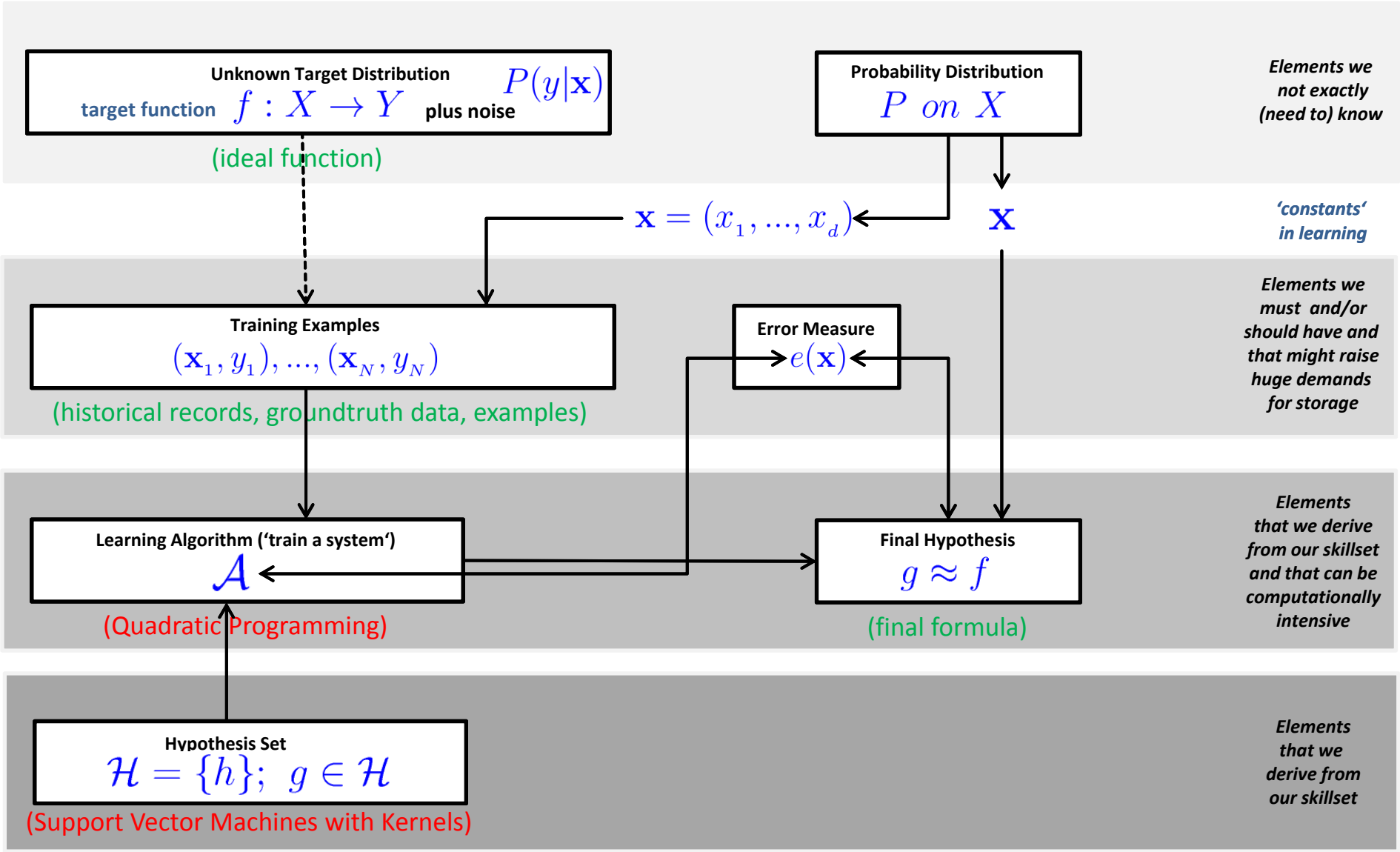
- Major benefit of Kernels: Computing done in original space

(independent from transformed space)

- Linear Kernel $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$ (linear in features)
- Polynomial Kernel $K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$ (polynomial of degree d)
- RBF Kernel $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$ (large distance, small impact)

(the win: kernel can compute this without ever computing the coordinates of the data in that space, next slides)

Solution Tools: Support Vector Classifier & QP Algorithm



Non-Linear Transformations with Support Vector Machines

- Same idea: work in z space instead of x space with SVMs
  - Understanding effect on solving (labels remain same)
  - SVMs will give the 'best seperator'

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

(replace this simply with z's obtained by Φ)

(result from this new inner product is given to quadratic programming optimization as input as before)

  - Value: inner product is done with z instead of x – the only change
  - Result after quadratic programming is hyperplane in z space using the value
- Impacts of Φ to optimization
  - From linear to 2D → probably no drastic change
  - From 2D to million-D → sounds like a drastic change but just inner product
  - Input for $K(x_i, x_i')$ remains the number of data points (nothing to do with million-D)
  - Computing longer million-D vectors is 'easy' – optimization steps 'difficult'

- **Infinite-D Z spaces are possible since the non-linear transformation does not affect the optimization**

Kernels & Infinite Z spaces

- Understanding advantage of using a kernel
  - Better than simply enlarging the feature space
  - E.g. using functions of the original features like $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2.$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \emptyset(\boldsymbol{x}_i)^T \emptyset(\boldsymbol{x}_j)$$

(maps data to higher-dimensional feature spaces)

- Computational advantages
  - By using kernels only compute $K(x_i, x_i')$
  - Limited to just all $\binom{n}{2}$ distinct pairs $i,\ i'$

(number of 2 element sets from n element set)

  - Computing without explicitly working in the enlarged feature space
  - Important because in many applications the enlarged feature space is large

(computing would be infeasible then w/o kernels)

- Infinite-D Z spaces $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \emptyset(\boldsymbol{x}_i)^T \emptyset(\boldsymbol{x}_j)$
  - Possible since all that is needed to compute coefficients are inner products

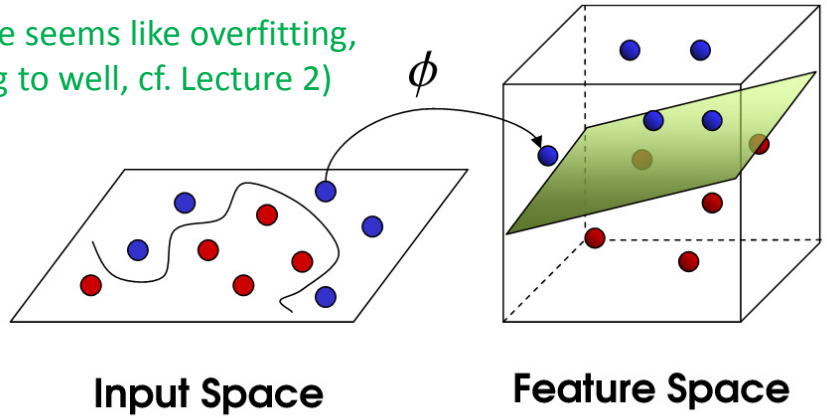
[1] An Introduction to Statistical Learning

- **Kernel methods like RBF have an implicit and infinite-dimensional features space that is not 'visited'**

Visualization of SVs

- Problem: z-Space is infinite (unknown)
  - How can the Support Vectors (from existing points) be visualized?
  - Solution: non-zero alphas have been the identified support vectors
    (solution of quadratic programming optimization will be a set of alphas we can visualize)
  - Support vectors exist in Z – space (just transformed original data points)
  - Example: million-D means a million-D vector for $\mathbf{w}$
  - But number of support vector is very low, expected E_{out} is related to #SVs
    (generalization behaviour despite million-D & snake-like overfitting)

(snake seems like overfitting, fitting to well, cf. Lecture 2)



[3] Visualization of high-dimensional space

- **Counting the number of support vectors remains to be a good indicator for generalization behaviour even when performing non-linear transforms and kernel methods that can lead to infinite-D spaces**

(rule of thumb)

LibSVM – svm-train Parameters – Supported Kernels

- 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)
```

(choose a kernel, but it may need additional parameters, e.g. 2 here needs gamma as additional parameter)

(Regularization Parameter)

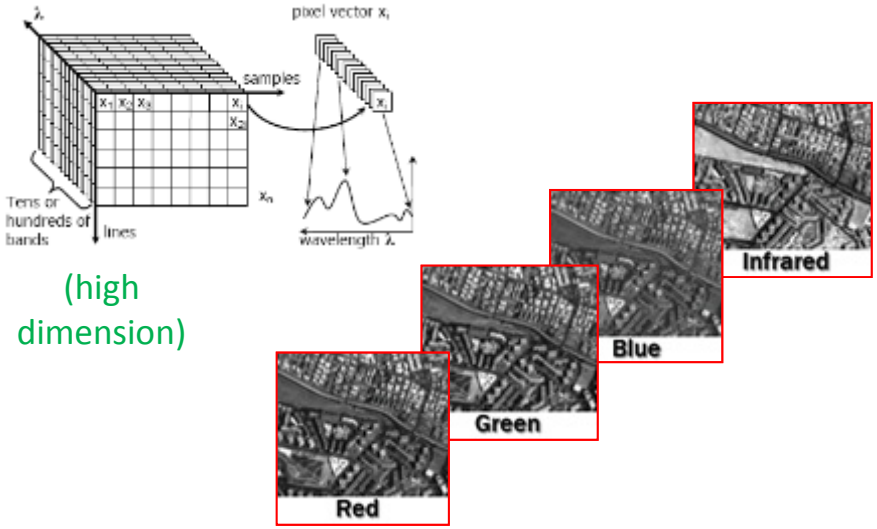
[4] LibSVM Webpage

Moderate Complexity – Rome Dataset (B2SHARE Record 86)

- Example dataset: Geographical location: Image of Rome, Italy
  - Remote sensor data obtained by Quickbird satellite

- High-resolution (0.6m) panchromatic image

Pansharpened (UDWT) low-resolution (2.4m) multispectral images



(high dimension)

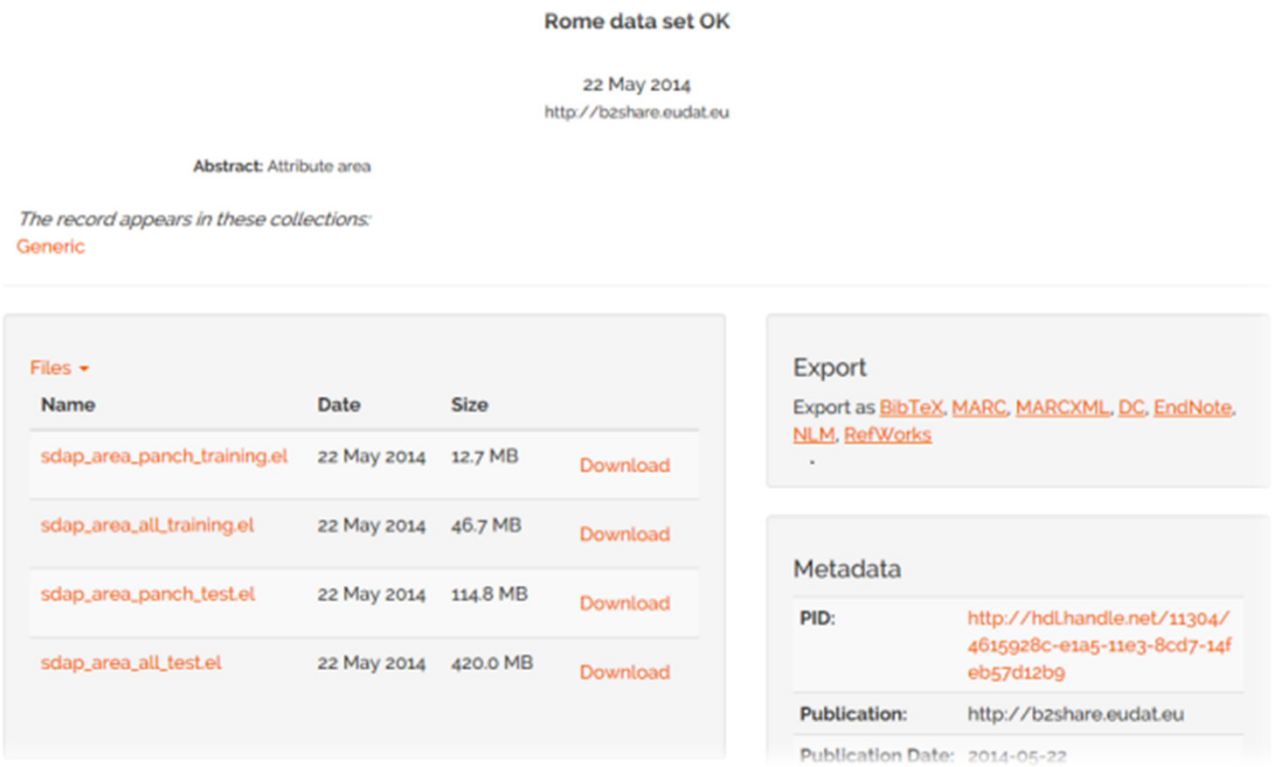
(Reasoning for picking SVM: Good classification accuracies on high dimensional datasets, even with a small ‚rare' number of training samples)

[5] Rome Image dataset

Moderate Complexity – Rome Dataset (B2SHARE Record 86)

- Data is publicly available in EUDAT B2SHARE tool

[5] Rome Image dataset



(persistent handle link for publication into papers)

Exercises

High Complexity – Indian Pines Dataset

- *Indian Pines Dataset Raw and Processed*

Abstract: 1) Indian raw: 1417x614x200 (training 10% and test)
2) Indian processed:1417x614x30 (training 10% and test)

[6] Indian Pine Image dataset

B2SHARE
Store and Share Research Data

Files

Name	Date	Size	
indian_processed_training.el	05 Feb 2015	11.7 MB	Download
indian_raw_test.el	05 Feb 2015	747.1 MB	Download
indian_raw_training.el	05 Feb 2015	83.0 MB	Download
indian_processed_test.el	05 Feb 2015	105.6 MB	Download

```
[vsc42544@gligar02 Rome]$ pwd
/apps/gent/tutorials/machine_learning/classification/Rome
[vsc42544@gligar02 Rome]$ ls -al
total 1160512
drwxr-xr-x 2 vsc40003 vsc40003      4096 Nov 22 15:43 .
drwxr-xr-x 6 vsc40003 vsc40003      4096 Nov 22 15:44 ..
-rw-r--r-- 1 vsc40003 vsc40003 419974873 Nov 22 15:39 sdap_area_all_test.el
-rw-r--r-- 1 vsc40003 vsc40003  46652874 Nov 22 15:40 sdap_area_all_training.el
-rw-r--r-- 1 vsc40003 vsc40003 114763982 Nov 22 15:42 sdap_area_panch_test.el
-rw-r--r-- 1 vsc40003 vsc40003  12745692 Nov 22 15:42 sdap_area_panch_training.el
```

SVM Multi-class Classification (1)

- Multi-class classification common in science & engineering
  - Requires different approach as previous 'binary classification' (2 classes)
  - Cf. associated remote sensing SVM application (e.g. 52 land cover classes)

(advanced topic – required much more study – here just the two most popular approaches)

- One vs. One (all pairs) classification
  - Given K > 2 classes, this approach creates $\binom{K}{2}$ different SVMs
  - Each of the different SVMs compares a pair of classes (i.e. binary classifier)
  - Classification is done by using test data points with each of the classifiers
  - Count number of times that each point is assigned to each of the k classes
  - Class is which it was most frequently assigned in $\binom{K}{2}$ pairwise classification

(the more classes – the more SVMs are created to perform pairwise classification – the more computational complexity)

[1] An Introduction to Statistical Learning

- **One vs. one multi-class classification creates different SVMs that compare each a pair of k classes**

SVM Multi-class Classification (2)

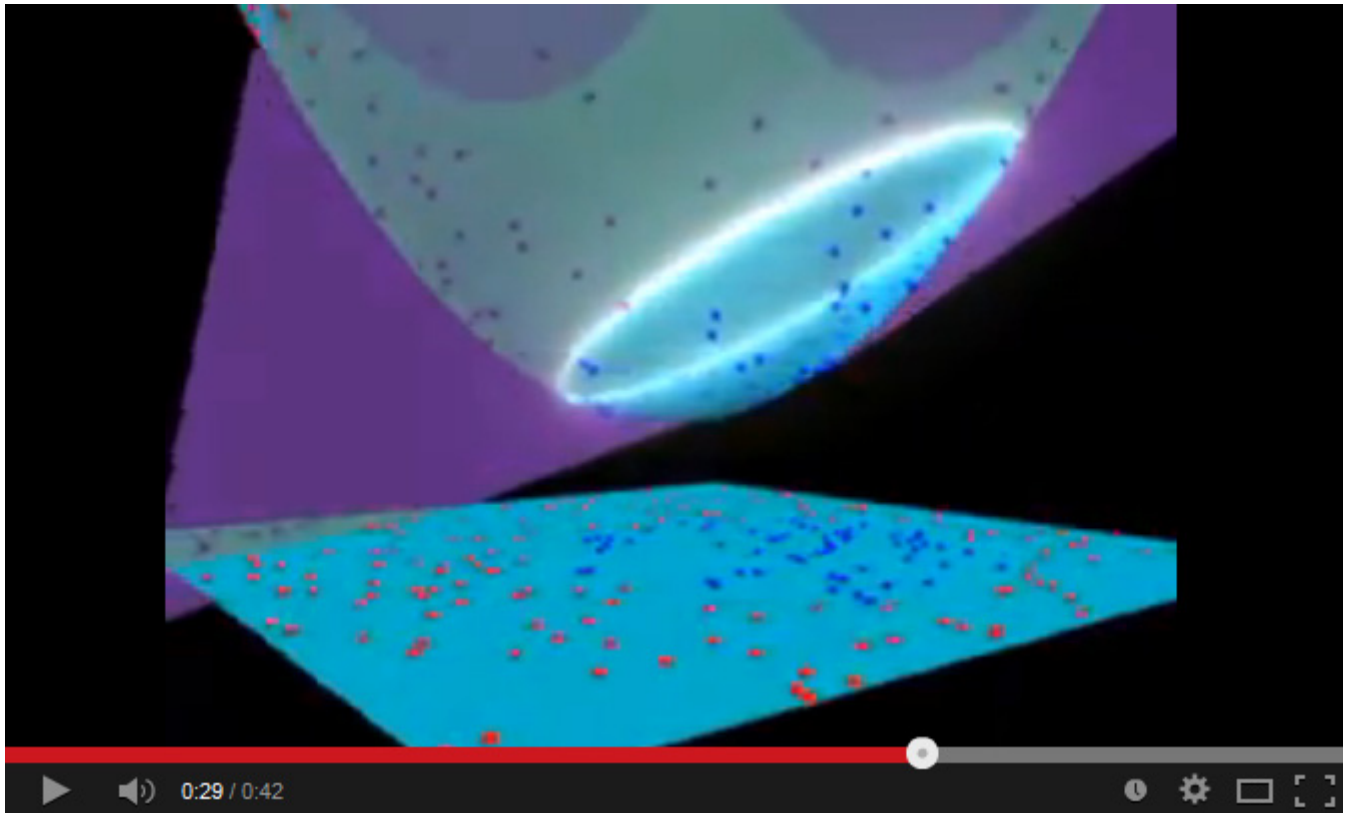
- One vs. All classification
  - Given K > 2 classes, this approach fits only K SVMs
  - Each time one of the K classes is compaired to the remaining K-1 classes
  - Coefficients that result from fitting an SVM comparing the *k*th class (coded as +1) to the others (coded as −1) are $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$
  - Classification with testset data
  - Assign the testset data to the class for which the following is largest:
  - Reasoning: high level of confidence that the test data points belong to the kth class rather than to any of the other classes

(less SVMs are created – but more comparisons are done while creating the classifiers – can be computationally intensive)

[1] An Introduction to Statistical Learning

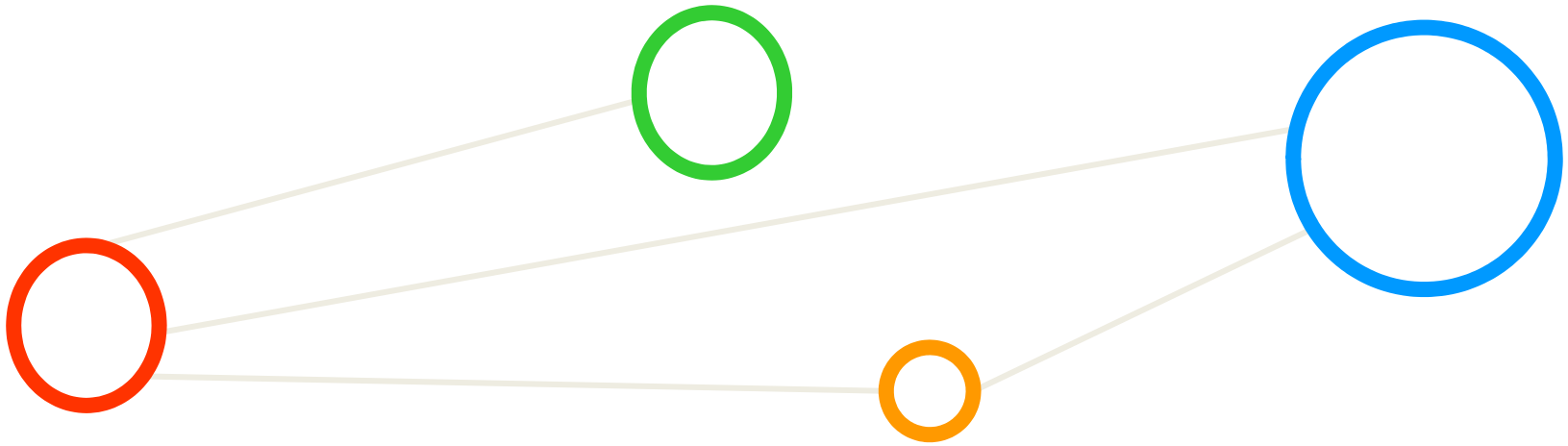
- **One vs. all multi-class classification creates K SVMs compairing it with to the remaining K-1 classes**

[Video] SVM with Polynomial Kernel Example



[7] YouTube, SVM with Polynomial Kernel

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] E. Kim, 'Everything You Wanted to Know about the Kernel Trick',
  Online: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html
- [3] Visualization of high-dimensional space,
  Online: http://i.imgur.com/WuxyO.png
- [4] LibSVM Webpage,
  Online: https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- [5] Rome Dataset, B2SHARE,
  Online: http://hdl.handle.net/11304/4615928c-e1a5-11e3-8cd7-14feb57d12b9
- [6] Indian Pine Image Dataset, B2SHARE,
  Online: http://hdl.handle.net/11304/7e8eec8e-ad61-11e4-ac7e-860aa0063d1f
- [7] YouTube, 'SVM with Polynomial Kernel',
  Online: https://www.youtube.com/watch?v=3liCbRZPrZA

Acknowledgements and more Information: Yaser Abu-Mostafa, Caltech Lecture series, YouTube


Data
Science
Big
Research
science
analysis
methods
simulations
big
processing
technologies
TB
computing
European
Supercomputing
Scientific
computational
research
HPC
Juelich
datasets
Climate
modelling
structures
Computational
resources
services
E.g
Centre
climate
Computing
nodes
centers
community
Environment
storage
ENES
Infrastructure
Enable
Modelling
important
different
databases
brain
Project
EUDAT
increasing
DLCL
Level
images
performance
management
data-intensive
Service
distributed
large-scale
access
System
DLCLs
Understanding
Germany
Provide
NASA
Energy
systems
Simulation
day
Processing
Health
directory
Earth
Technologies
Hardware
AR5
project
Structure
General
current
hundreds
Services
network
Nutshell
Landscape
Concrete
Matters
system
compute
cores
Summary
Large
stored
Resources
analyze
external
levels
often
Work
Using
GB
Methods
Common
IPCC
physical
DKRZ
Use
etc
High
PB
Key
scientific
basis
files
Approaches
construction
Fusion
results
disciplines
expertise
Computer
Analysis
device
concepts
funding
new
fusion
within
measurement
Policy-based
forms
cross-disciplinary
year
like
Test
Cross-Disciplinary
EU
formats
SMAQ
international
model
engineering
References
codes
use
JSC
e.g
acquisition
create
MapReduce
via