To be specialized in one specific subject is also a good thing; it will make you a professional in a specific subject, but when you come to society they will expect you to use your knowledge in practical situations.

Changing your habits is not an easy thing, but one is urged to do it for a number of reasons. A successful teacher should renew his lectures so that students are not bored.

Changing your view, clothes or even your hair cut.

This indicates that he is up to date and also his mind constant.

Also, you might think of changing your way of making, or synthesizing, chemical compounds. You might come out with less expensive methods.

It is troublesome, as it seems, but it keeps me fresh in information and with a good status among my colleagues.

Changing your glasses color, for example, would be attractive to students and colleagues, and make me feel better.

Lastly, change is a difficult decision in the human mind, but it is important for many good reasons, and gets back on the human with good benefits.

Alternating work and dormancy in your life pace, with activities and exercise of great benefit to the body and the mind.

In contrast, many people believe that changing is really important in people's lives.

Lastly, change is a difficult decision in the human mind, but it is important for many good reasons, and gets back on the human with good benefits.

Lastly, change is a difficult decision in the human mind, but it is important for many good reasons, and gets back on the human with good benefits.

Lastly, change is a difficult decision in the human mind, but it is important for many good reasons, and gets back on the human with good benefits.

Lastly, change is a difficult decision in the human mind, but it is important for many good reasons, and gets back on the human with good benefits.
Identifying the native language (L1) of a writer based on a sample of their writing in a second language (L2)

**Our data**

- **L1s**: Arabic (ARA), Chinese (ZHO), French (FRA), German (DEU) Hindi (HIN), Italian (ITA), Japanese (JPN), Korean (KOR), Spanish (SPA), Telugu (TEL), Turkish (TUR)
- **L2**: English
- **Real-world objects**: For each L1, 1,000 texts in L2 from The ETS Corpus of Non-Native Written English (former TOEFL11), i.e. $Train \cup DevTest$
- **Target class**: L1

*More detailed info is available at the course website.*
96 numerical features = relative character frequencies

Example

"Finally having people with many academic broad know"

|       | a     | b     | c     | d     | e     | m     | n     | o     | F     | g     | h     | i     | k     | l     | p     | r     | t     | v     | w     | y     |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| <SPACE> | 0.17073171 | 0.14634146 | 0.02439024 | 0.04878049 | 0.04878049 | 0.07317073 | 0.04878049 | 0.09756098 | 0.07317073 | 0.02439024 | 0.02439024 | 0.04878049 | 0.09756098 | 0.02439024 | 0.07317073 | 0.04878049 | 0.02439024 | 0.04878049 | 0.04878049 |
NLI
Load training and test data

```r
> data <- read.table(file =
"https://ufal.mff.cuni.cz/~hladka/2015/docs/fv.c.1.gram.rel.traindev.csv",
  sep = ",",
  header = T)

> train.indexes <- read.table(file =
"https://ufal.mff.cuni.cz/~hladka/2015/docs/train.txt")

> test.indexes <- read.table(file =
"https://ufal.mff.cuni.cz/~hladka/2015/docs/dev.txt")
```

- **Task #1** Look at the top 5 rows of the train set
- **Task #2** Find out the target class distribution in the train set
**NLI**

**Load training and test data**

---

**Task #1**

```r
> data[1:5,]

<table>
<thead>
<tr>
<th>file</th>
<th>c.1.1</th>
<th>c.1.2</th>
<th>...</th>
<th>c.1.96</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>88.txt</td>
<td>0.001524390</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>KOR</td>
</tr>
<tr>
<td>5301.txt</td>
<td>0.000000000</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>KOR</td>
</tr>
<tr>
<td>15340.txt</td>
<td>0.000559597</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>KOR</td>
</tr>
<tr>
<td>23684.txt</td>
<td>0.001183430</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>KOR</td>
</tr>
<tr>
<td>38991.txt</td>
<td>0.001741150</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>KOR</td>
</tr>
</tbody>
</table>
```

---

**Task #2**

```r
> table(train$class)

ARA  DEU  FRA  HIN  ITA  JPN  KOR  SPA  TEL  TUR  ZHO
900  900  900  900  900  900  900  900  900  900  900
```
Online demo
• Java applet at http://svm.dcs.rhbnc.ac.uk/

The implementation of SVMs in R
• library(e1071), but there are also other libraries (kernlab, shogun ...)
• training: function svm()
• prediction: function predict()
• svm() can work in both classification and regression mode
• if response variable is categorical (factor) the engine switches to classification
SVM in R

\[
\text{model} = \text{svm(formula, data=, kernel=, cost=, cross=, \ldots)}
\]

- `?svm`
- `kernel` defines the kernel used in training and prediction. The options are: linear, polynomial, radial basis and sigmoid (default: radial)
- `cost` – cost of constraint violation (default: 1)
- `cross` – optional, with the value k the k-fold cross-validation is performed

Kernel parameters - see later
Q: What is the purpose of SVM kernels?

Recall
We want to map non-linearly separable instances to higher-dimensional place where we can apply large/soft margin classifier.

Kernel trick
We don’t need to compute the coordinates of the data in that space, but just simply replace the dot product by the kernel function $K(x_i, x_j)$
### SVM kernels in e1071

<table>
<thead>
<tr>
<th>Kernel name</th>
<th>Formula</th>
<th>Learning parameters and their default values</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>$x_i^T x_j$</td>
<td></td>
</tr>
<tr>
<td>polynomial</td>
<td>$(\gamma x_i^T x_j + c_0)^d$</td>
<td>$\gamma$, gamma=1/(data dimension) $c_0$, coef0=0 $d$, degree=3</td>
</tr>
<tr>
<td>radial</td>
<td>$\exp(-\gamma(|x_i - x_j|^2))$</td>
<td>$\gamma$, gamma=1</td>
</tr>
<tr>
<td>sigmoid</td>
<td>$\tanh(\gamma x_i^T x_j + c_0)$</td>
<td>$\gamma$, gamma=1/(data dimension) $c_0$, coef0=0</td>
</tr>
</tbody>
</table>
Non-linear kernel functions

- polynomial kernel
  - smaller degree can generalize better
  - higher degree can fit (only) training data better

- radial basis
  - very robust
  - you should try and use it when polynomial kernel is weak to fit your data
SVM – parameter cost

Linearly separable vs. NOT linearly separable data

- possible remedy for overlapping classes: “soft margins”
- parameter: cost
- higher cost
  → misclassifications are penalized strongly
  → the model will not generalize much
- lower cost
  → relaxed model; misclassifications are not penalized
**Task #3** Train a model with the linear kernel and cost=1

```r
model <- svm(class ~ ., train, kernel = "linear", cost = 1)
> model

Call:
svm(formula = class ~ ., data = train, kernel = "linear",
    cost = 1)

Parameters:
  SVM-Type:  C-classification
  SVM-Kernel:  linear
  cost:  1
  gamma:  0.01041667

Number of Support Vectors:  8840
```
Task #4 Find out accuracy of the model on the test data

```r
> prediction <- predict(model, test, type="class")
> mean(prediction==test$class)
0.3804545
```
Confusion matrix of the predicted classes

Task #5 Compute the confusion matrix for the true target classes and the predicted classes

```r
> table(pred = prediction, true = test$class)

<table>
<thead>
<tr>
<th>true</th>
<th>ARA</th>
<th>DEU</th>
<th>FRA</th>
<th>HIN</th>
<th>ITA</th>
<th>JPN</th>
<th>KOR</th>
<th>SPA</th>
<th>TEL</th>
<th>TUR</th>
<th>ZHO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARA</td>
<td>28</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>DEU</td>
<td>3</td>
<td>34</td>
<td>11</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>FRA</td>
<td>10</td>
<td>7</td>
<td>33</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>HIN</td>
<td>13</td>
<td>10</td>
<td>2</td>
<td>29</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>27</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>ITA</td>
<td>8</td>
<td>9</td>
<td>11</td>
<td>4</td>
<td>58</td>
<td>0</td>
<td>1</td>
<td>17</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>JPN</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>56</td>
<td>14</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>KOR</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>14</td>
<td>43</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>SPA</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>8</td>
<td>34</td>
<td>2</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>TEL</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>34</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>52</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>TUR</td>
<td>5</td>
<td>10</td>
<td>11</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>ZHO</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>46</td>
</tr>
</tbody>
</table>
```
Task #6 Train a model with the polynomial kernel of degree 2

```r
> model2 <- svm(class ~ ., train, kernel = "polynomial", degree = 2,
  cost = 1)
> prediction2 <- predict(model2, test, type="class")
> mean(prediction2 == test$class)
[1] 0.3172727
```

Task #7 Train a model with the polynomial kernel of degree 3

```r
> model3 <- svm(class ~ ., train, kernel = "polynomial", degree = 3,
  cost = 1)
> prediction3 <- predict(model3, test, type="class")
> mean(prediction3 == test$class)
[1] 0.2490909
```
Task #8 Train a model with the radial basis kernel

```r
> model4 <- svm(class ~ ., train, kernel = "radial")
> prediction4 <- predict(model4, test, type="class")
> mean(prediction4 == test$class)
[1] 0.4081818
```
**NLI**

**Training data scaling**

- `svm()` by default normalizes the training data, so each feature has zero mean and unit variance
- **scaling of the data usually drastically improves the results**
- After scaling, the decision boundary will not depend on the range of feature values, but only on the distribution of instances.
The training data should not contain constant columns (features with constant values)!

```r
> set.seed(99)
> s <- sample(nrow(train)); indices.train <- s[1:2000]
> train.small <- train[indices.train,]
> # check constant columns
> names(train.small[, sapply(train.small,
        function(v) var(v, na.rm=TRUE)==0)])
# [1] "c.1.11" "c.1.13" "c.1.25"
# Warning message:
# In svm.default(x, y, scale = scale, ..., na.action = na.action):
# Variable(s) ‘c.1.11’ and ‘c.1.13’ and ‘c.1.25’ constant.
# Cannot scale data.
> train.small$c.1.11=NULL; train.small$c.1.13=NULL
> train.small$c.1.25=NULL

• model accuracy with scaling: 35.00%
• model accuracy without scaling: 9.09%
```
• SVM is a more complicated method in comparison with the previous and usually requires parameter tuning!
• parameter tuning can take a very long time on big data, use a reasonably smaller part is often recommended

```r
> model.tune= tune.svm(class ~ ., data=train.small,
>                        kernel = "radial",
>                        gamma = c(0.001, 0.005, 0.01, 0.015, 0.02),
>                        cost = c(0.5, 1, 5, 10))
```

```
> model.tune
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
  gamma  cost
  0.01   1

- best performance: 0.739
```
K-fold cross-validation

- parameter cross

```r
> model.best <- svm(class ~ ., train.small,
>                   kernel = "radial",
>                   gamma = 0.01,
>                   cost = 1,
>                   cross = 10)

> model.best$accuracies
[1] 33.0 27.5 31.0 33.5 28.0 29.0 29.0 33.5 33.0 34.5
> model.best$tot.accuracy
# [1] 31.2
> prediction.best <- predict(model.best, test, type="class")
> mean(prediction.best==test$class)
[1] 0.3472727
```
Class weighting

- `class.weights` parameter
  In case of asymmetric class sizes you may want to avoid possibly overproportional influence of bigger classes. Weights may be specified in a vector with named components, like:
  
  ```r
  m <- svm(x, y, class.weights = c(A = 0.3, B = 0.7))
  ```
• Note that SVMs may be very sensible to the proper choice of parameters, so always check a range of parameter combinations, at least on a reasonable subset of your data.

• Be careful with large datasets as training times may increase rather fast.

• C-classification with the RBF kernel (default) can often be a good choice because of its good general performance and the few number of parameters (only two: cost and gamma).

• When you use C-classification with the RBF kernel: try small and large values for cost first, then decide which are better for the data by cross-validation, and finally try several gamma values for the better cost.