Lecture 12: Classification: SVM

Course: Biomedical Data Science
Parisa Rashidi
Fall 2018

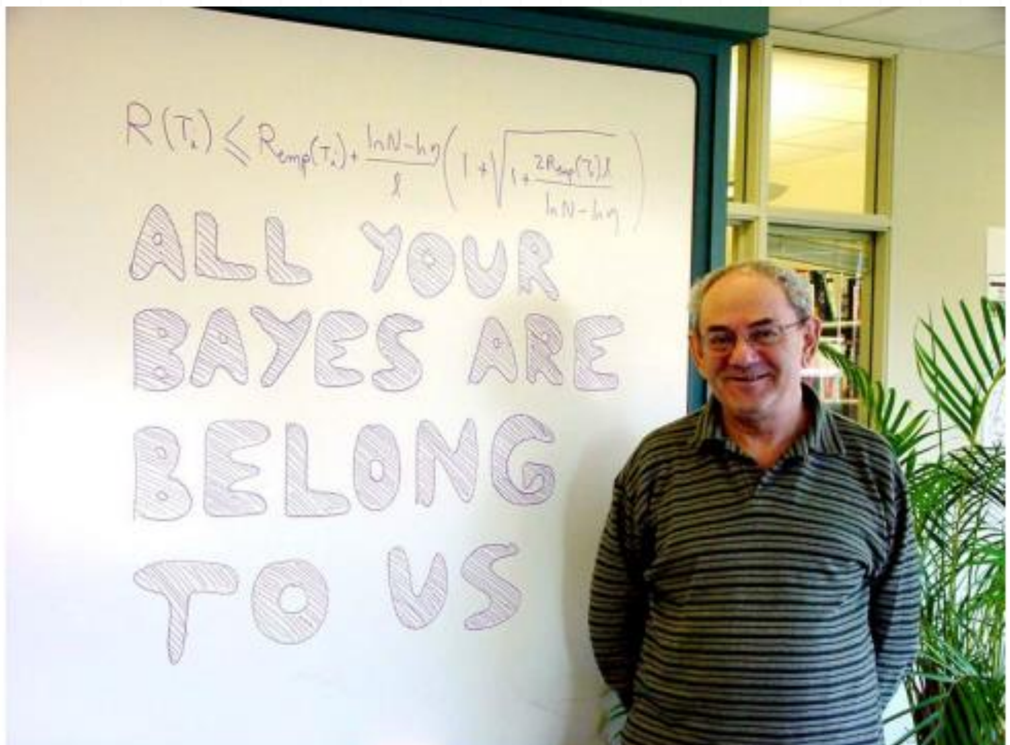
Administrative Items

- Grad Survey proposal topic due in two weeks
  - Provide a link to the survey paper on canvas
  - More comprehensive understanding is expected

Agenda

- k-NN
- Decision Tree based Methods
- Support Vector Machines
- Neural Networks
- Deep learning
- NLP

Support Vector Machines (SVM)



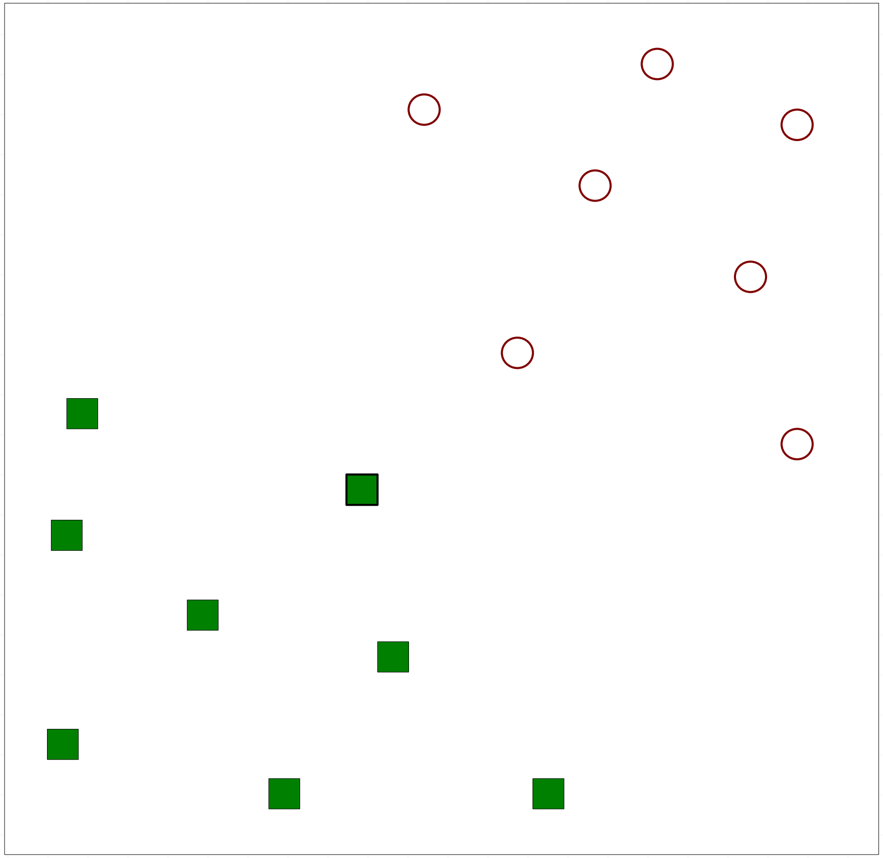
Vladimir Vapnik

Support Vector Machine

- Represents the decision boundary using only a subset of training examples known as support vectors
- Convex optimization problems with a unique solution
- Normally, a linear classifier
  - But, we can also use non-linear kernels
    - Kernels: application-specific measures of similarity
    - Leading to “kernel machines”

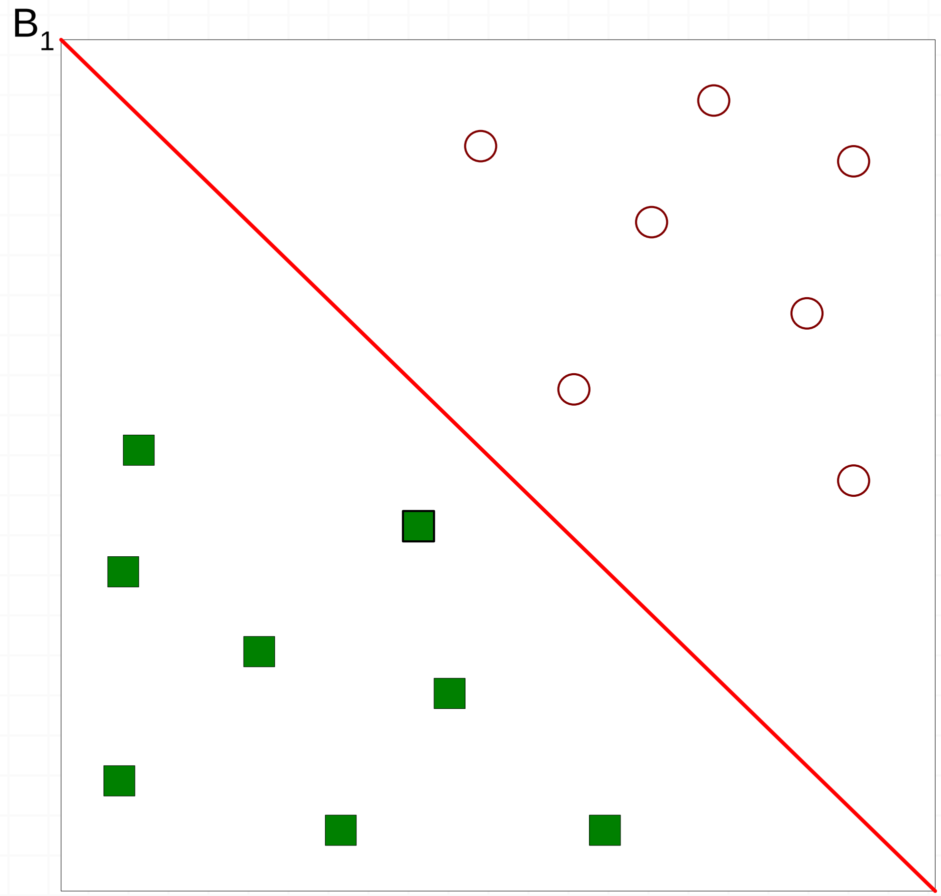
SVM Decision Boundary

- Question
  - Find a linear hyperplane to separate the data



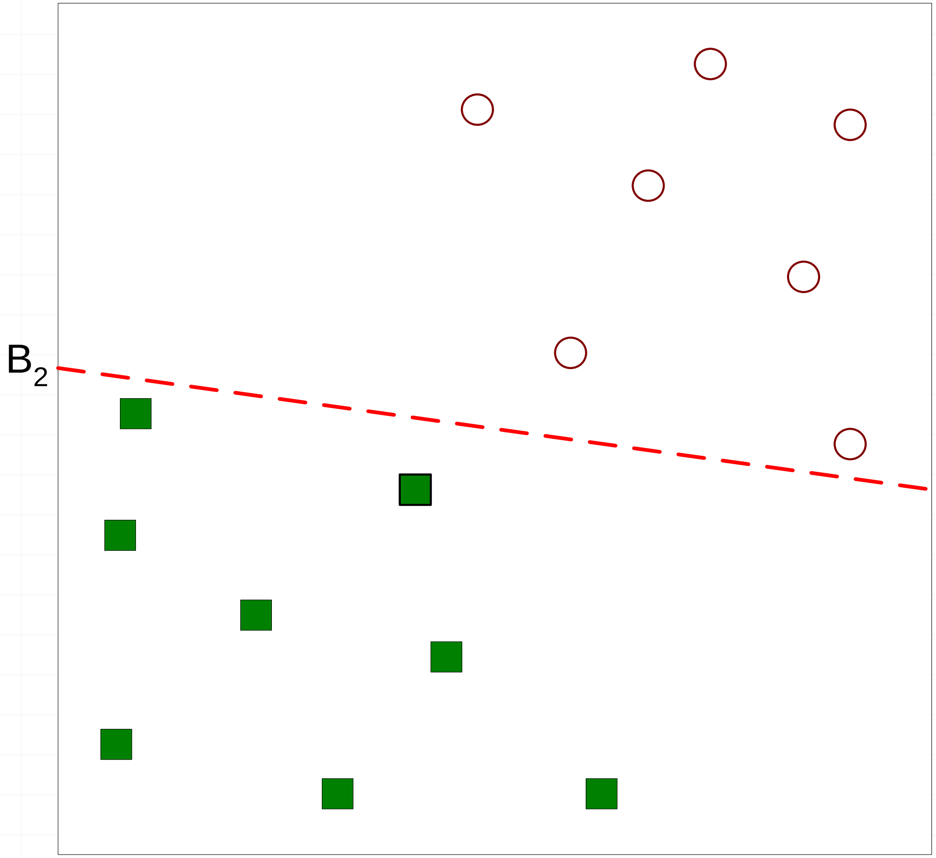
SVM Decision Boundary

- One possible solution



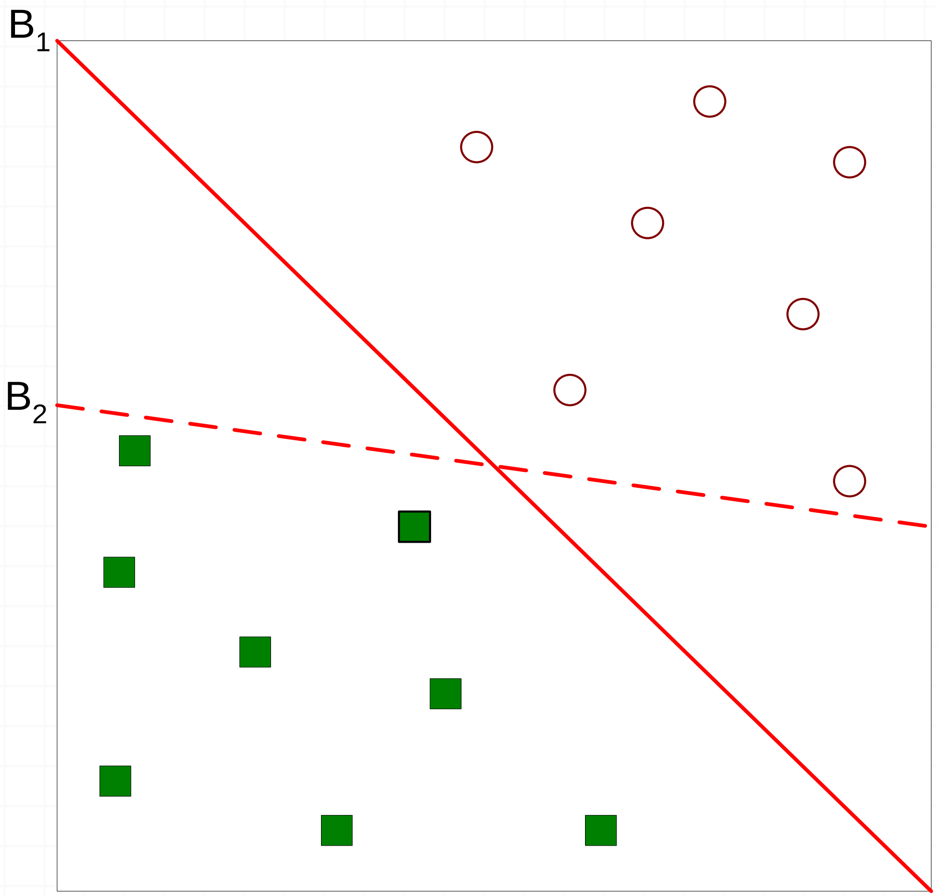
SVM Decision Boundary

- Another possible solution

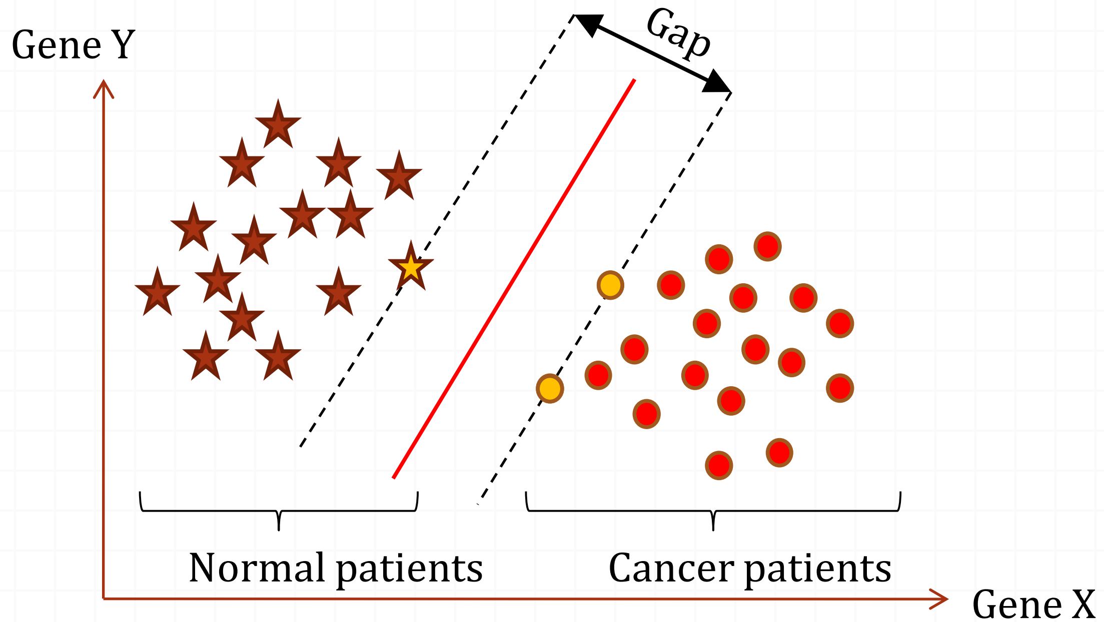


SVM Decision Boundary

- Which one is better?



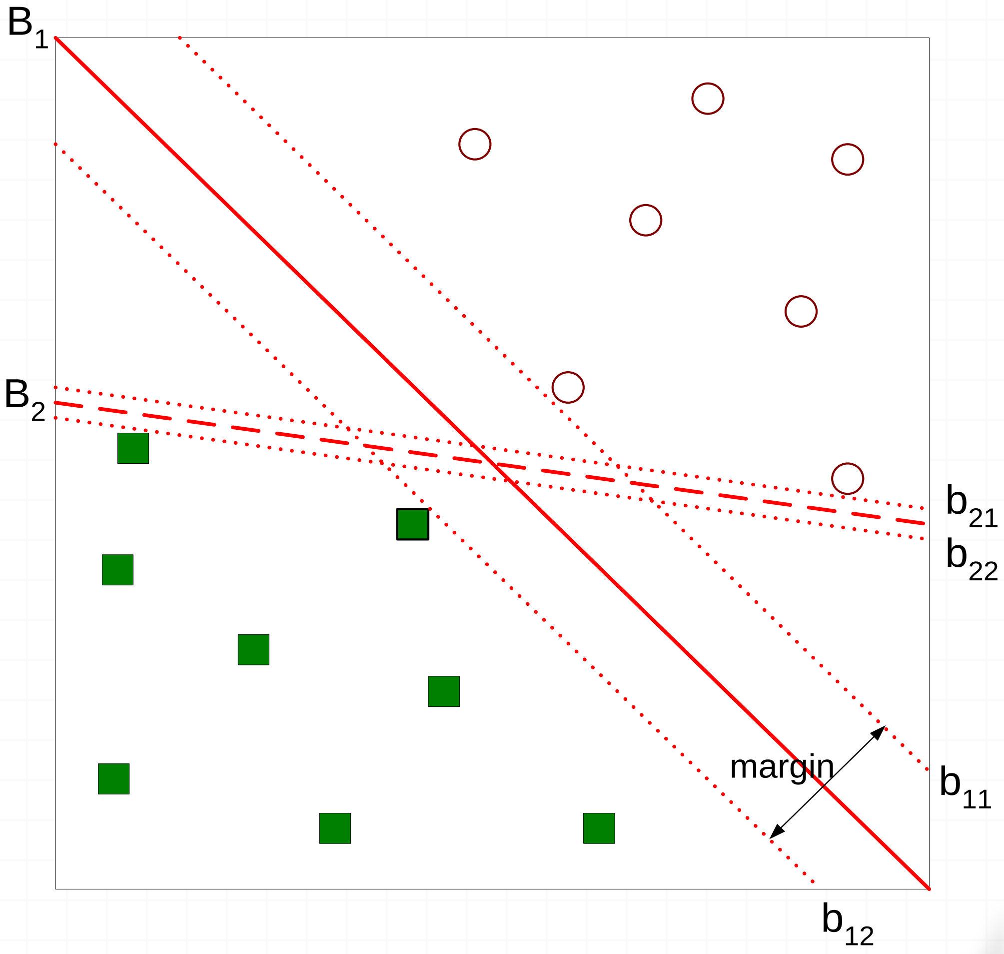
Main ideas of SVMs



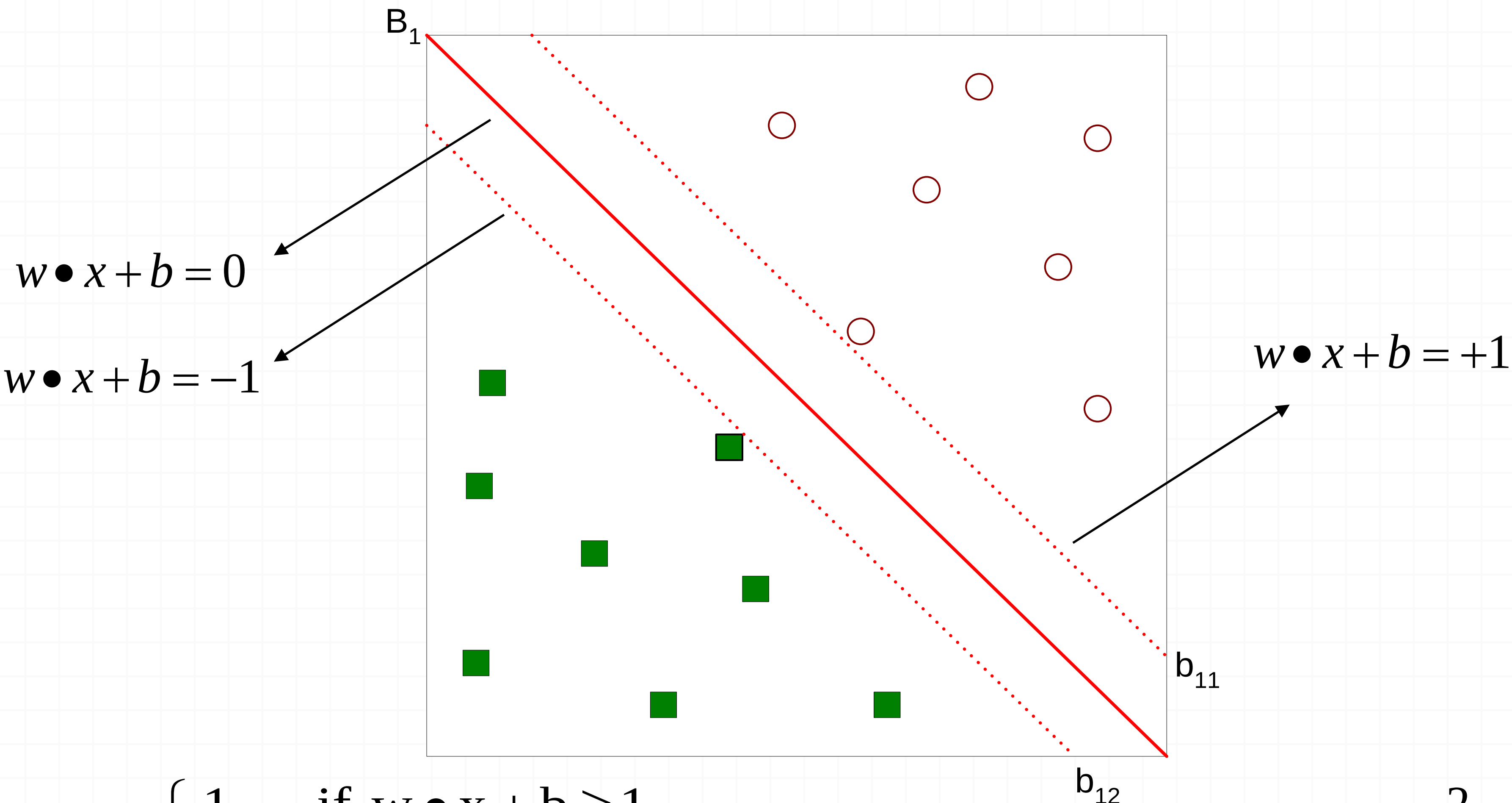
- Represent patients geometrically (by “vectors”);
- Find a linear decision surface (“hyperplane”) that can separate patient classes <u>and</u> has the largest distance (i.e., largest “gap” or “margin”) between border-line patients (i.e., “support vectors”);

Better Decision Boundary

- Margin definition:
  - Distance from the hyperplane to the closest instances on either side
- Find hyperplane that maximizes the margin
- B1 is better than B2
  - Wider margin, and better generalization



Support Vector Machine



$$f(x) = \begin{cases} 1 & \text{if } \mathrm{w} \bullet \mathrm{x} + \mathrm{b} \geq 1 \\ -1 & \text{if } \mathrm{w} \bullet \mathrm{x} + \mathrm{b} \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| w \|^2}$$

Let's Formulate this ...

- Our simple rule

$$f(x) = \begin{cases} 1 & \text{if } \mathrm{w} \bullet \mathrm{x} + \mathrm{b} \geq 1 \\ -1 & \text{if } \mathrm{w} \bullet \mathrm{x} + \mathrm{b} \leq -1 \end{cases}$$

It can be re-written as

$$y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1$$

- But, we also want to maximize the margin (or minimize the inverse of the margin), so we get:

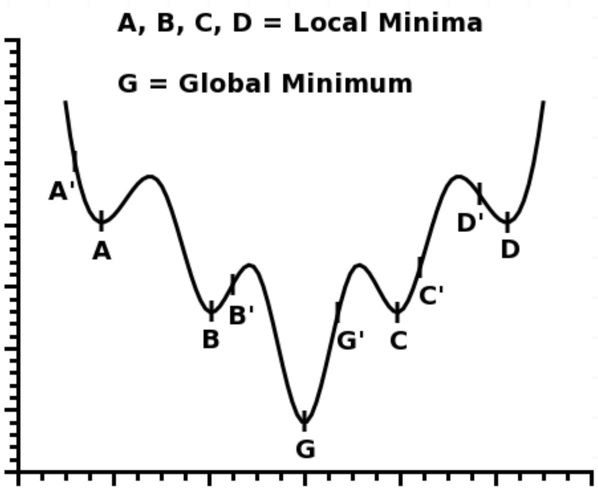
$$\min(|\frac{||\boldsymbol{w}||^2}{2})$$

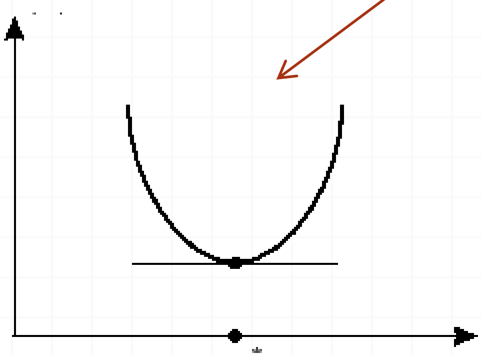
Subject to $y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1 \qquad i = 1,2, \ldots N$

A Convex Optimization Problem

- This is a convex optimization problem (quadratic programming or QP)
  - Therefore, it has only a single optimum (great!)
  - Therefore no worries about convergence, learning rates, etc.

Non-convex





Convex

SVM Training

- Involves estimating parameters **w** and b
- This is a quadratic optimization programming (QP) problem
  - Can be solved using Lagrange multiplier technique

$$\min(|\frac{||\boldsymbol{w}||^2}{2})$$

Subject to $y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1 \qquad i = 1,2, \ldots N$

Lagrange Multiplier

- Write its Lagrangian:

$$(0) \quad L_p = \frac{||\boldsymbol{w}||^2}{2} - \sum_{i=1}^{N} \lambda_i (y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) - 1) \qquad where\ \lambda_i \geq 0$$

- Here λ_i are called the Lagrange multipliers
- We will minimize L_p with respect to **w** and b:

$$(1) \quad \frac{\partial L_p}{\partial \boldsymbol{w}} = 0 \quad \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$(2) \quad \frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

KKT Conditions

- Additionally, the following should hold (KKT condition, Karush-Kuhn-Tucker):

(3) $\lambda_i \geq 0$

(4) $\lambda_i [y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) - 1] = 0$

Support Vectors

- In equation (4) we had,

$$(4) \qquad \lambda_i[y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) - 1] = 0$$

To be zero, either λ_i must be zero, or

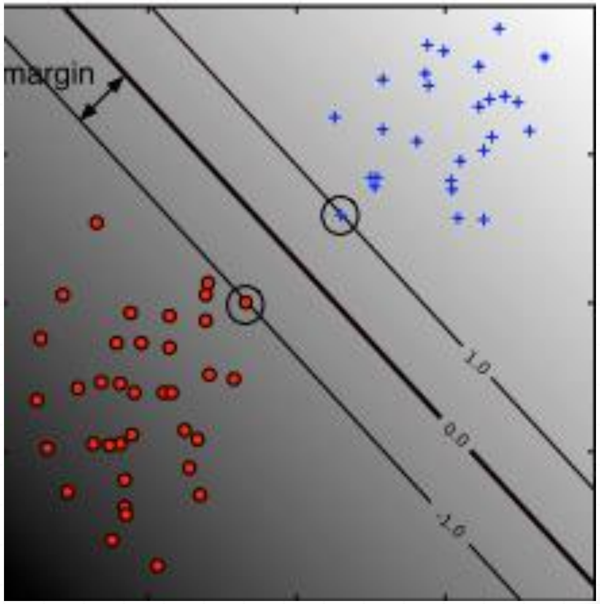
$$y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) = 1$$

which means that such a training instance $\mathbf{x}_i$ lies along the hyperplanes b_1 or b_2

  - Such a training instance is known as a support vector

Support Vectors

- A subset of training set, which are on the decision boundary
  - i.e. those for which $\lambda_i > 0$
- For the other training instances (i.e. those far away from the boundary), $\lambda_i = 0$
  - So, they have no effect on the hyperplane.



*A User's Guide to Support Vector Machines, Asa Ben-Hur, Jason Weston

Final Solution

- We can obtain b and λ using numerical solutions from equations (0)-(4) (we won't go into details)

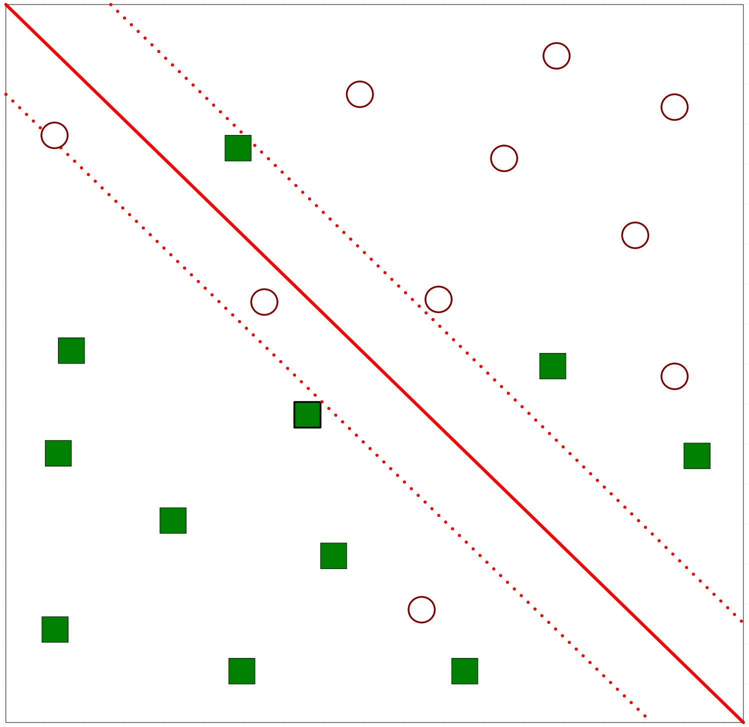
- The final solution will be:

We use this equation in order to classify a new data point z.

$$f(z) = sign(\boldsymbol{w}.\boldsymbol{z} + b) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i.\boldsymbol{z} + b\right)$$

Non-separable case

- How decision boundary can be modified to tolerate small training error?
  1. Instance may lie on the wrong side
  2. Instance may be inside the margin



Soft Margin

- We should consider a decision boundary that is tolerable to small training errors
  - This approach is known as soft margin
  - But SVM must consider the trade-off between the width of the margin and the number of training errors committed by the boundary

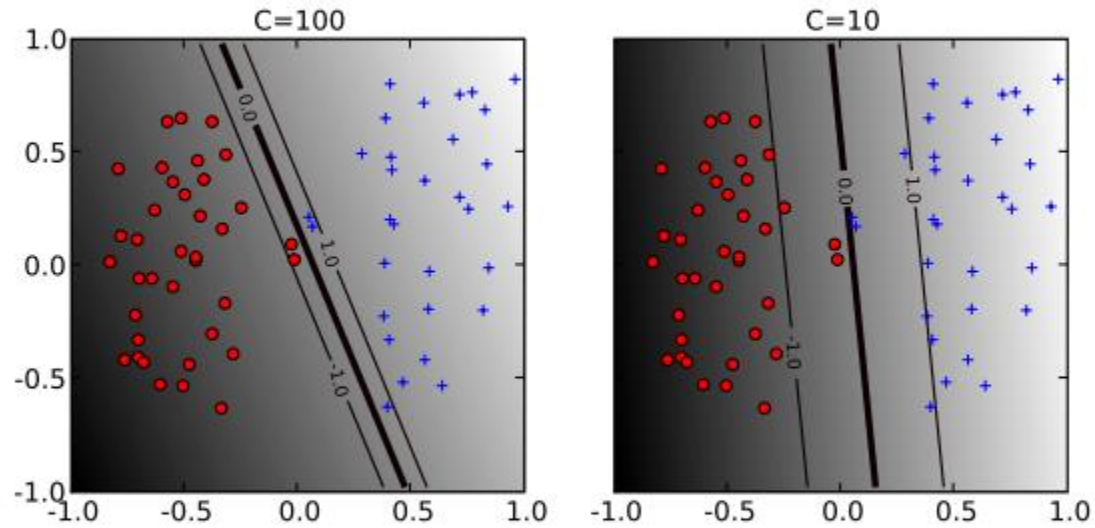
Soft Margin

- The inequality constrained should be relaxed
  - Introduce the slack variables ξ
- The objective function must be modified to penalize a decision boundary with large values of slack variables

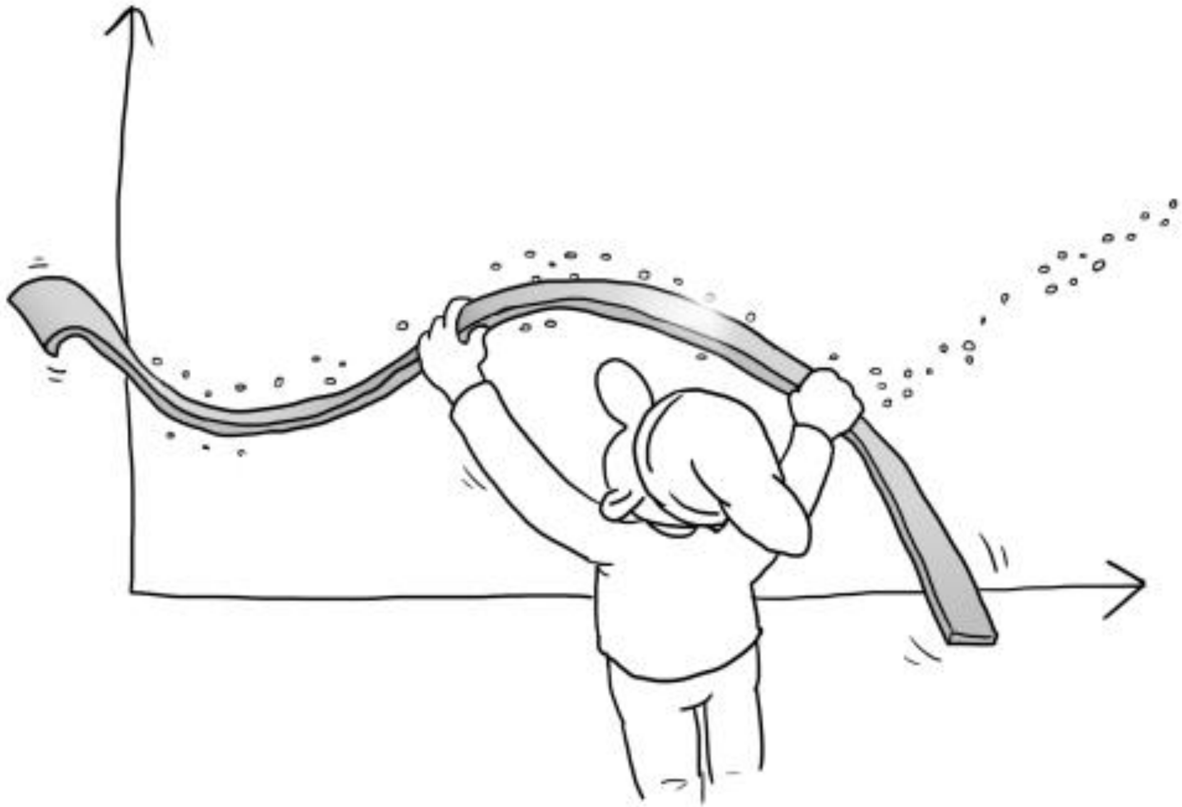
$$\min\left(\frac{||w||^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)\right)$$

$$f(x_i) = \begin{cases} 1 & \text{if } \mathrm{w} \bullet \mathrm{x_i} + \mathrm{b} \geq 1 - \xi_{\mathrm{i}} \\ -1 & \text{if } \mathrm{w} \bullet \mathrm{x_i} + \mathrm{b} \leq -1 + \xi_{\mathrm{i}} \end{cases}$$

Effect of C



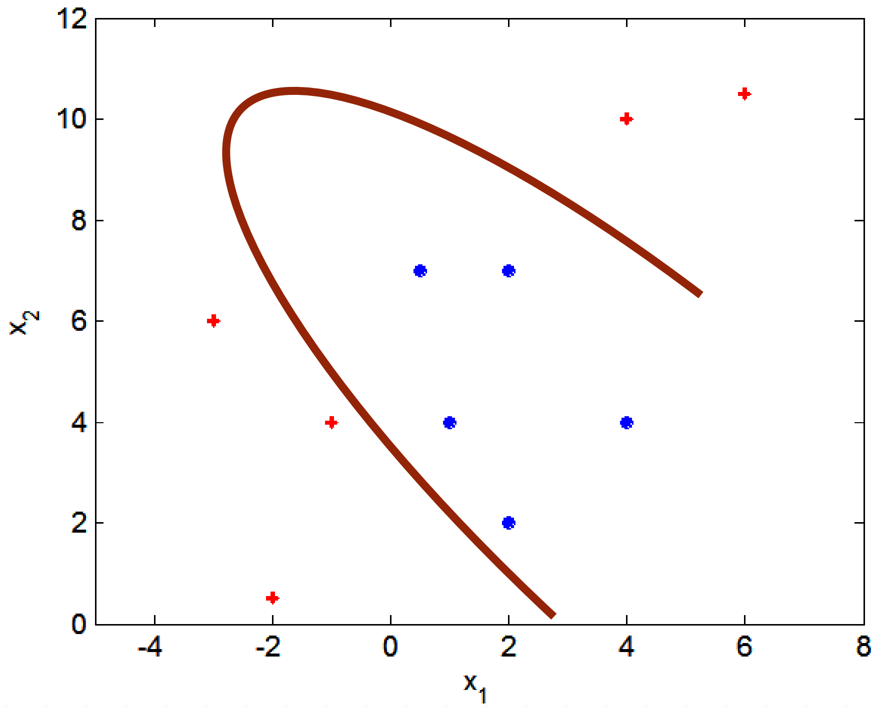
*A User's Guide to Support Vector Machines, Asa Ben-Hur, Jason Weston



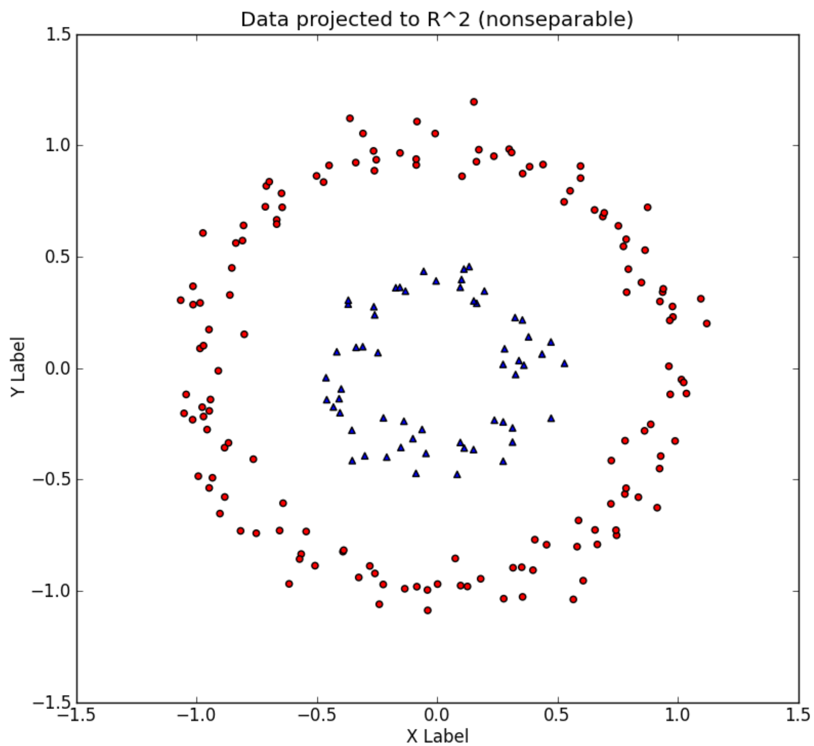
*From: ROBERTO BATTITI AND MAURO BRUNATO. The LION way. Machine Learning plus Intelligent Optimization. Lionsolver Inc., Los Angeles, June 2013.

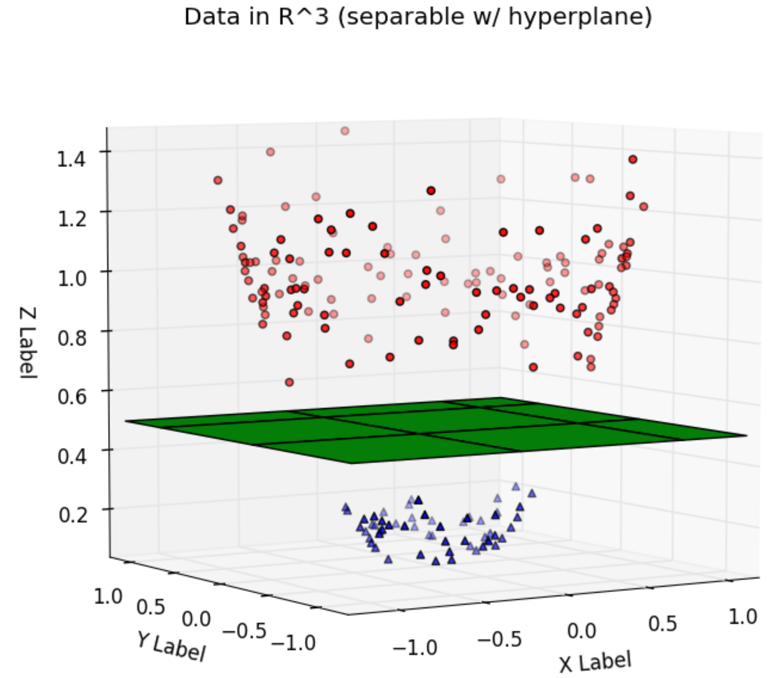
Nonlinear Decision Boundary

- What if the decision boundary is not linear?



Separable in higher-dimension





http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Kernels in Higher Dimensions Animated

1. https://www.youtube.com/watch?t=42&v=3liCbRZPrZA

2. https://www.youtube.com/watch?v=9NrALgHFwTo

Nonlinear Decision Boundary

- The trick is to transform data form its original coordinates space x into a new space $(\phi(x))$

$$\min\left(\left|\frac{||w||^2}{2}\right)\right.$$

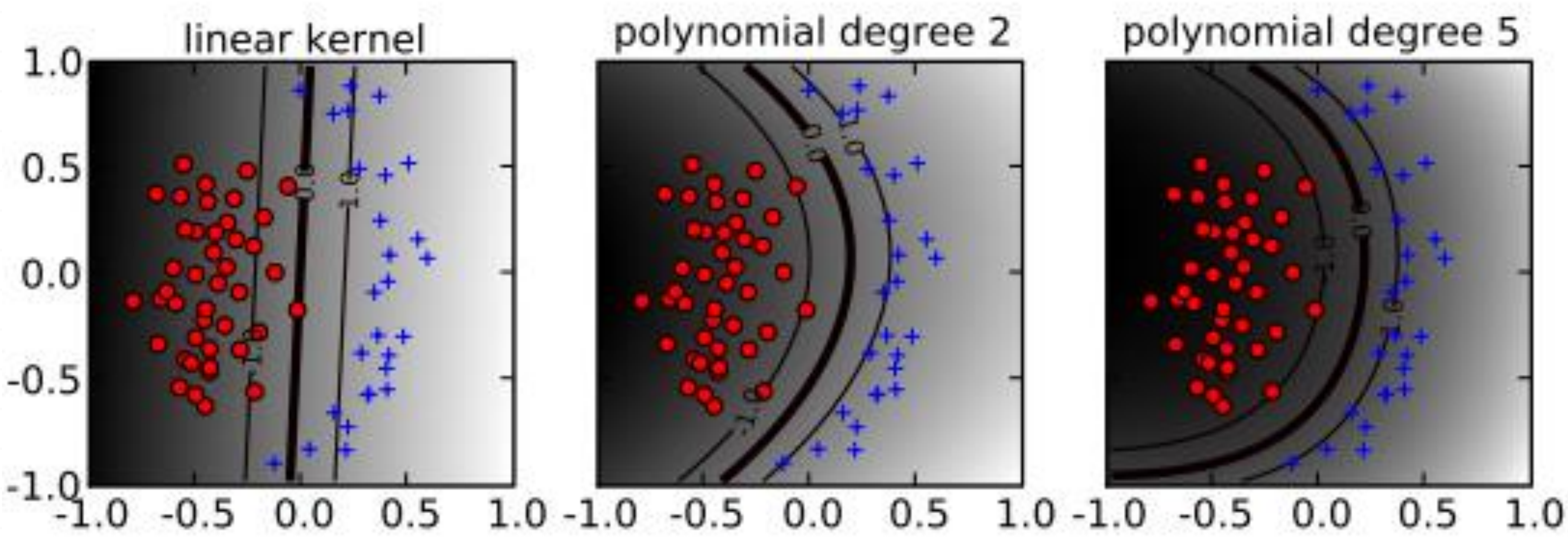
$$\text{Subject to } y_i(\boldsymbol{w}.\phi(\boldsymbol{x}_i)+b) \geq 1 \qquad i = 1,2,\ldots N$$

Solution

We use this equation in order to classify a new data point z.

$$f(z) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \phi(\boldsymbol{x}_i).\phi(z) + b\right)$$

Example



*A User's Guide to Support Vector Machines, Asa Ben-Hur, Jason Weston

Popular kernels

A kernel is a dot product in *some* feature space:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

<u>Examples:</u>

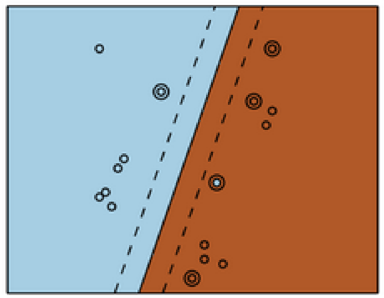
$K(x_i, x_j) = x_i \cdot x_j$	Linear kernel
$K(x_i, x_j) = \exp(-\gamma \lVert x_i - x_j \rVert^2)$	Gaussian kernel
$K(x_i, x_j) = \exp(-\gamma \lVert x_i - x_j \rVert)$	Exponential kernel
$K(x_i, x_j) = (p + x_i \cdot x_j)^q$	Polynomial kernel
$K(x_i, x_j) = (p + x_i \cdot x_j)^q \exp(-\gamma \lVert x_i - x_j \rVert^2)$	Hybrid kernel
$K(x_i, x_j) = \tanh(k x_i \cdot x_j - \delta)$	Sigmoidal

SVM Characteristics

- Convex optimization problem
  - Finds global minimum
  - Other methods such as neural network find local minimum
- Parameters should be tuned
  - Kernel type, C, ..
- High computational demands (both test and training)
  - Training stage
    - Naïve QP implementation
      - $O(n^3)$
    - More efficient techniques
      - Between $O(n)$ and $O(n^2)$

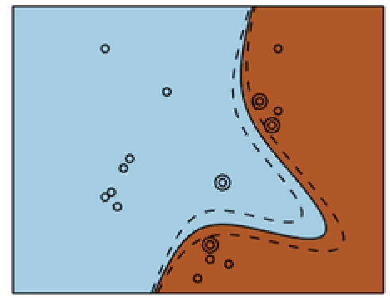
SVM in Scikit-learn (Python)

Linear kernel



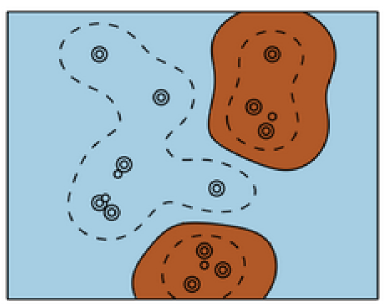
```
>>> svc = svm.SVC(kernel='linear')
```

Polynomial kernel



```
>>> svc = svm.SVC(kernel='poly',
...               degree=3)
>>> # degree: polynomial degree
```

RBF kernel (Radial Basis Function)

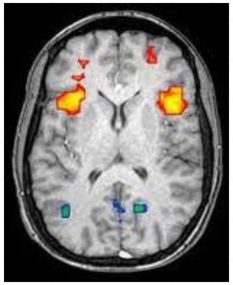


```
>>> svc = svm.SVC(kernel='rbf')
>>> # gamma: inverse of size of
>>> # radial kernel
```

see [link]

Neuroimaging Data (fMRI)

Neuroimaging Data Analysis

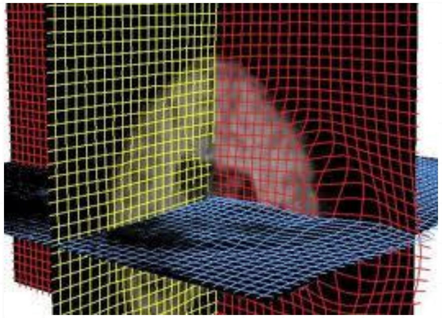


- Classify structural MR images
  - identify disease precursors as early as possible
    - e.g. MCI; Alzheimer's disorder (AD)
  - identify structural trajectories in neurodevelopment disorders
- Classify functional MR images
  - identify brain state associated with a stimulus

*Elizabeth Walter, Pattern Classification in Neuroimaging: Support Vector Machines

Neuroimaging Data

- Each image has lots and lots of voxels.
  - Of those, ~30,000 are actually brain voxels
    - grey matter ≈ 23,000 voxels
    - white matter ≈ 8,000 voxels
- To reduce the feature-space, we can:
  - resample data
  - do some latent variable extraction
    - e.g. using Principal Components Analysis (PCA) or some other automated feature extraction
- Then run SVM using a smaller number of "features"
  - e.g. 3-8 PCs rather than thousands of voxels



*Elizabeth Walter, Pattern Classification in Neuroimaging: Support Vector Machines

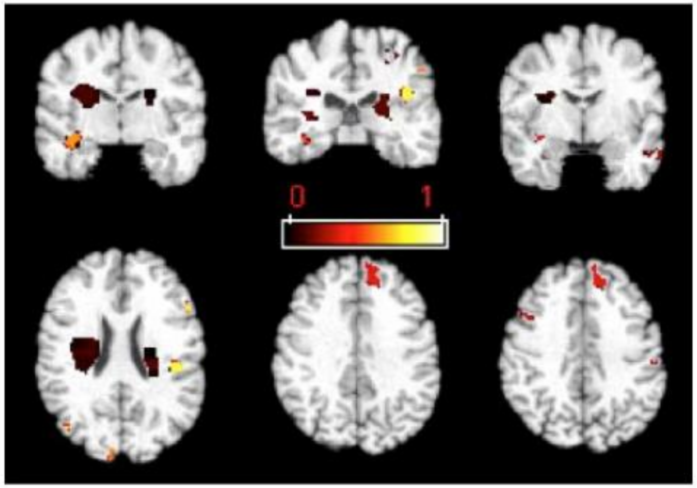
Example Features

- **Average**. For each ROI, calculate the mean activity over all voxels in the ROI. Use these ROI means as the input features.
- **ActiveAvg(n)**. For each ROI, select the most active voxels, then calculate the mean of their values. Again, use these ROI means as the input features.
- **"most active"** voxels are those whose activity while performing the task varies the most from their activity when the subject is at rest.
- **Active(n)**. Select the most active voxels over the entire brain. Use only these voxels as input features.

"Training fMRI Classifiers to Detect Cognitive States across Multiple Human Subjects," X. Wang, R. Hutchinson, and T. M. Mitchell, Neural Information Processing Systems 2003. December 2003.

Example: Schizophrenia

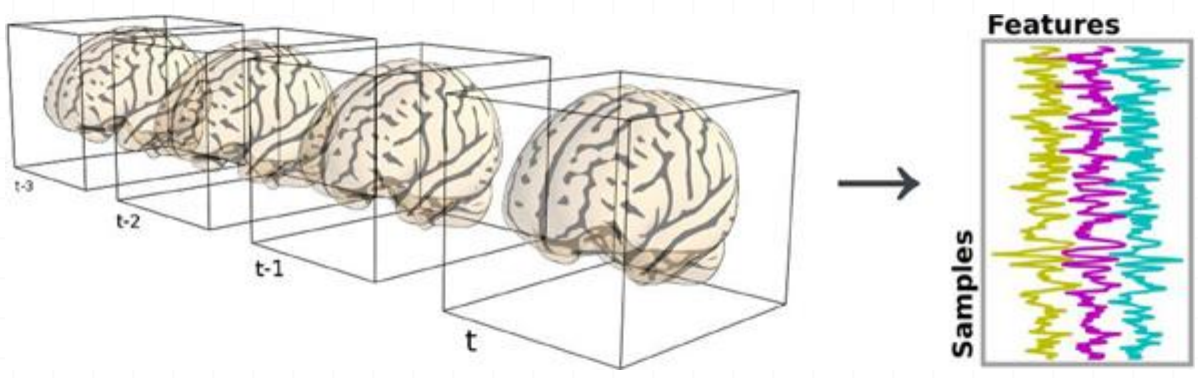
- Motivation: Which brain areas best distinguish brains of female schizophrenic patients from typical?



(Fan et al, Lect. Notes in CompSci, 2005)

Analysis with scikit-learn

```
#First we need to read data (and its mask)
#nibabel is a reader of common neuroimaging formats
>>import nibabel as ni
>>> X = ni.load("bold.nii").get_data()
```



Preprocessing

- De-trending: remove linear trend (scipy.signal.detrend)
- Normalization: all features will have the same range (sklearn.preprocessing.normalize)
- Frequency filtering: e.g. low frequencies due to physiological mechanisms, use Fourier Transform (scipy.fftpack.fft)

Prediction

```
from sklearn.linear_model import
   LogisticRegression as LR
from sklearn.cross_validation import
   cross_val_score

pipeline_LR = Pipeline([('selection',
   SelectKBest(f_classif, 500)),
   ('clf', LR(penalty='l1', C=0.05)])

scores_lr = []


for pixel in y_train.T:
    score = cross_val_score(pipeline_LR,
       X_train, pixel, cv=5)
    scores_lr.append(score)
```