Introduction to support vector machines

Tristan A. Hearn

November 17, 2010

Table of Contents

Introduction

Support vector machines are types of machine learning methods used in classification and regression problems.

- Similair to artificial neural networks
- Mathematically tractable

Introduction

Support vector machines are types of machine learning methods used in classification and regression problems.

- Similair to artificial neural networks
- Mathematically tractable

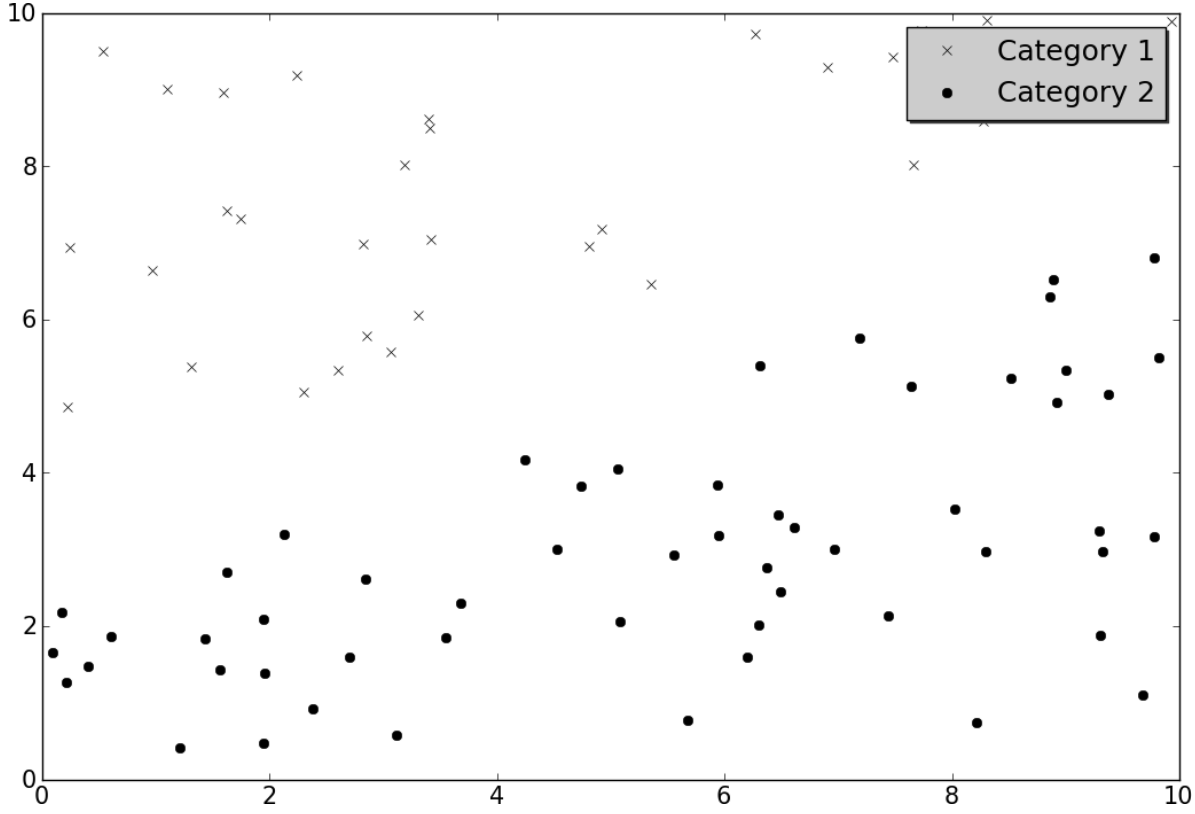
Introduction

Support vector machines are types of machine learning methods used in classification and regression problems.

- Similair to artificial neural networks
- Mathematically tractable

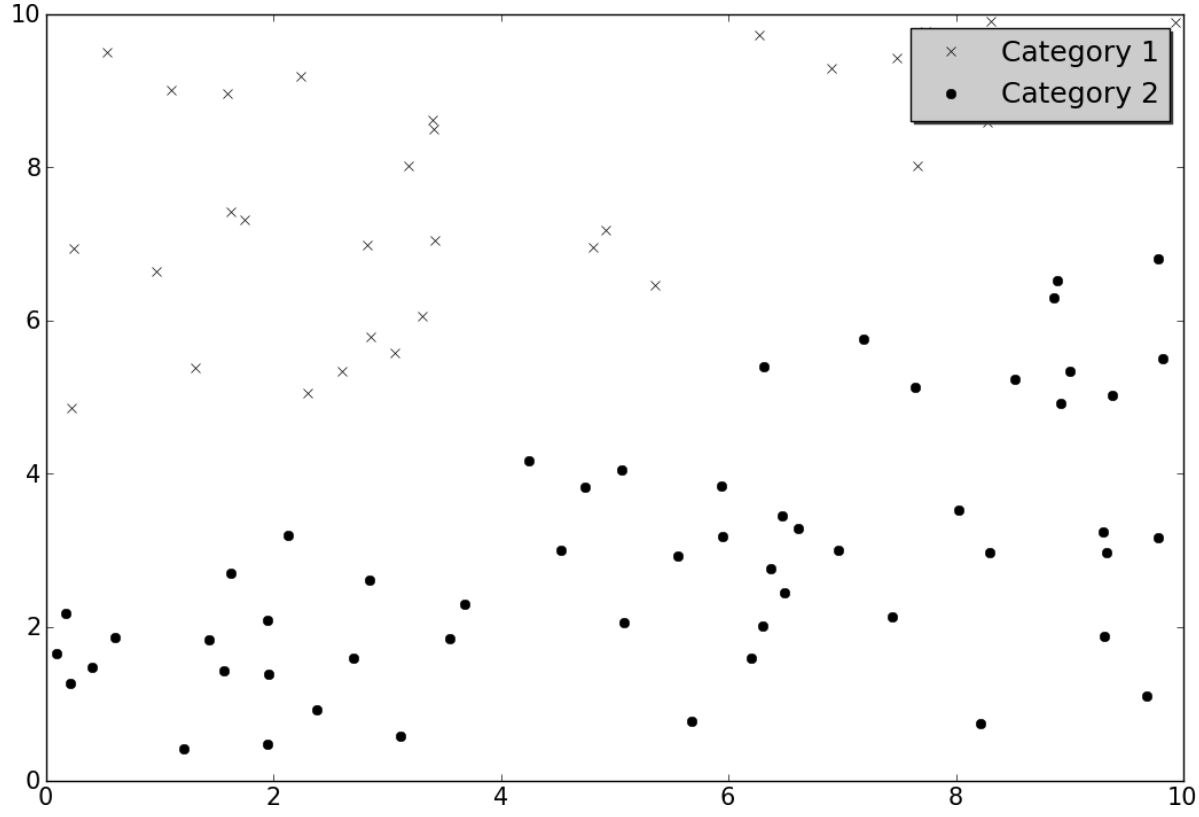
Linear classification problem

SVM methods are motivated by the linear classification problem: Given two categories and a series of data points in $\mathbb{R}^2$ each with a known classification, find a line which delimits the two categories:



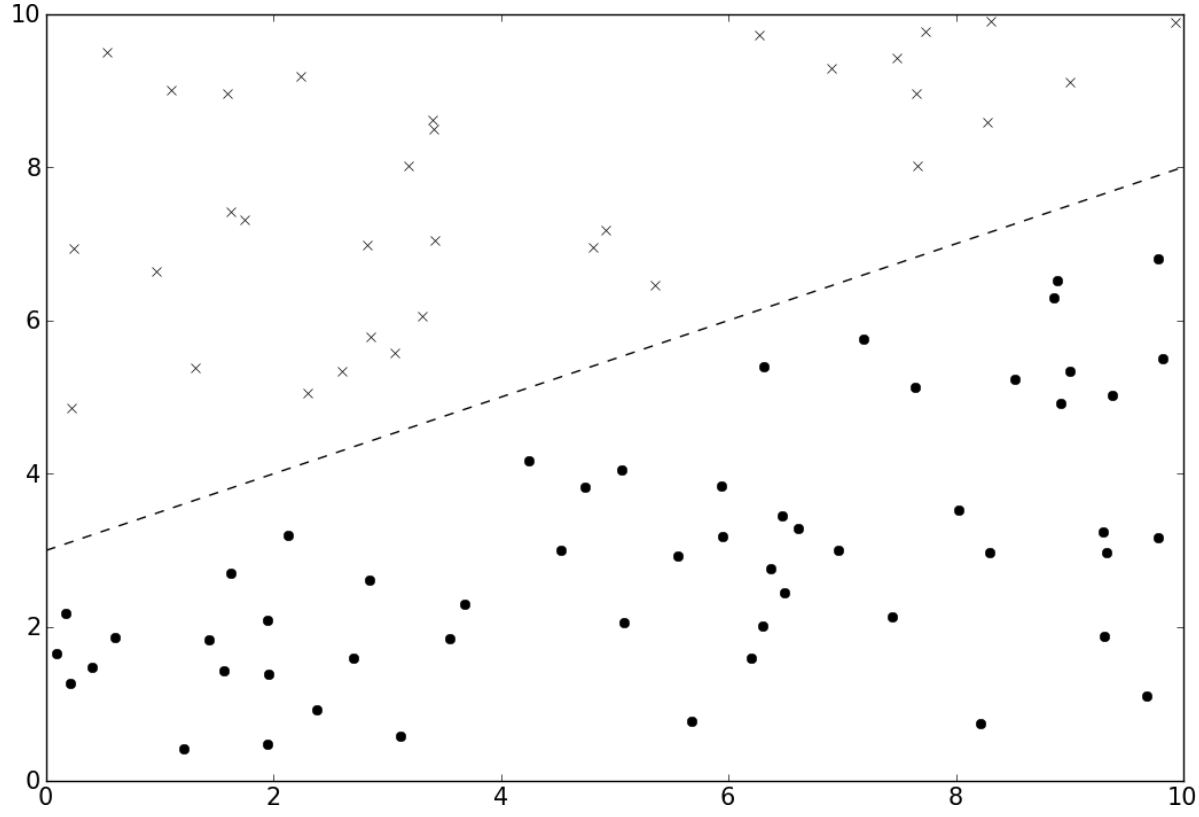
Linear classification problem

Without additional constraints, the problem is ill-posed:



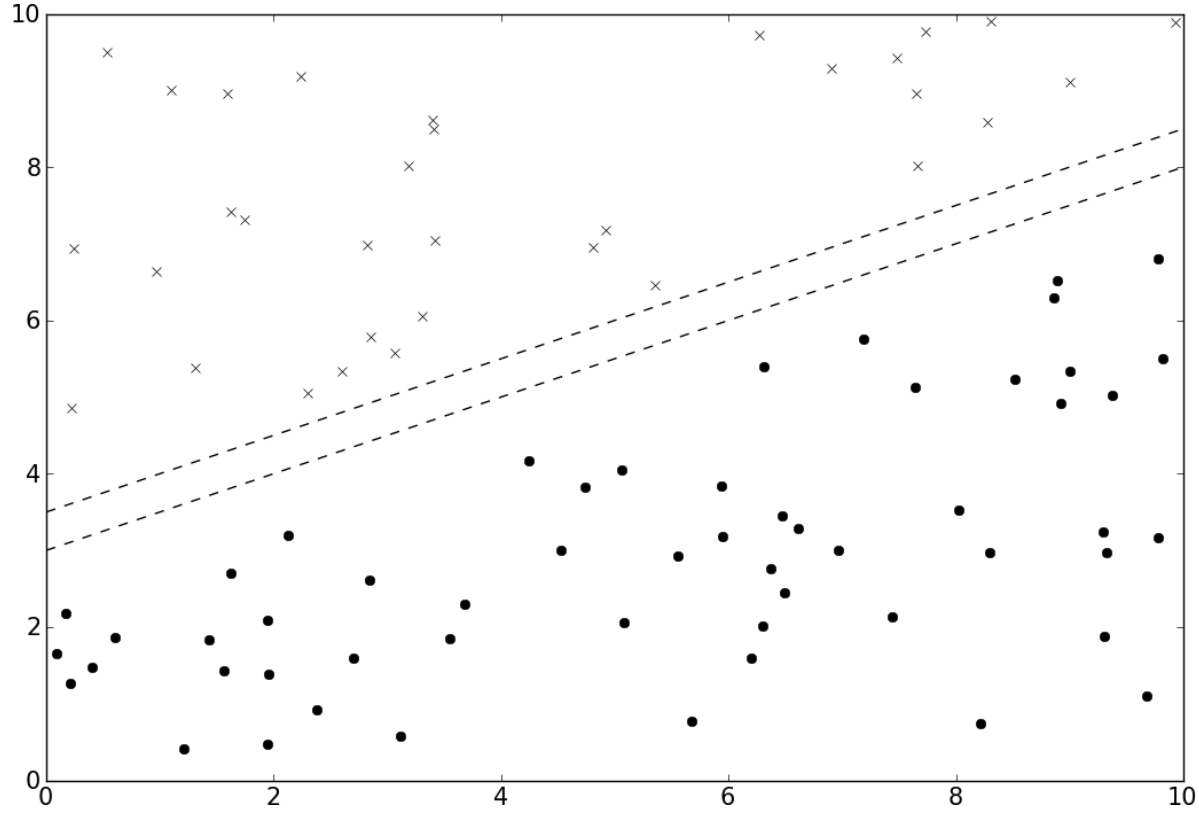
Linear classification problem

Without additional constraints, the problem is ill-posed:



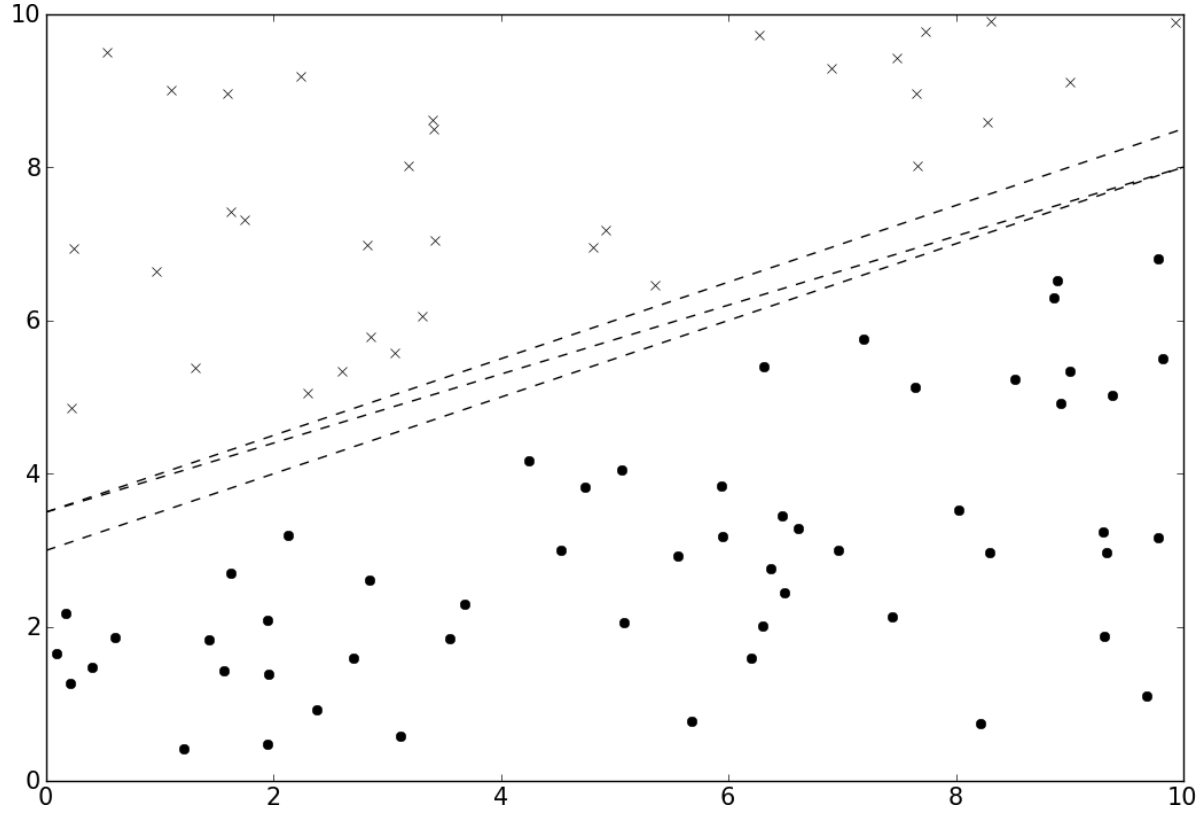
Linear classification problem

Without additional constraints, the problem is ill-posed:



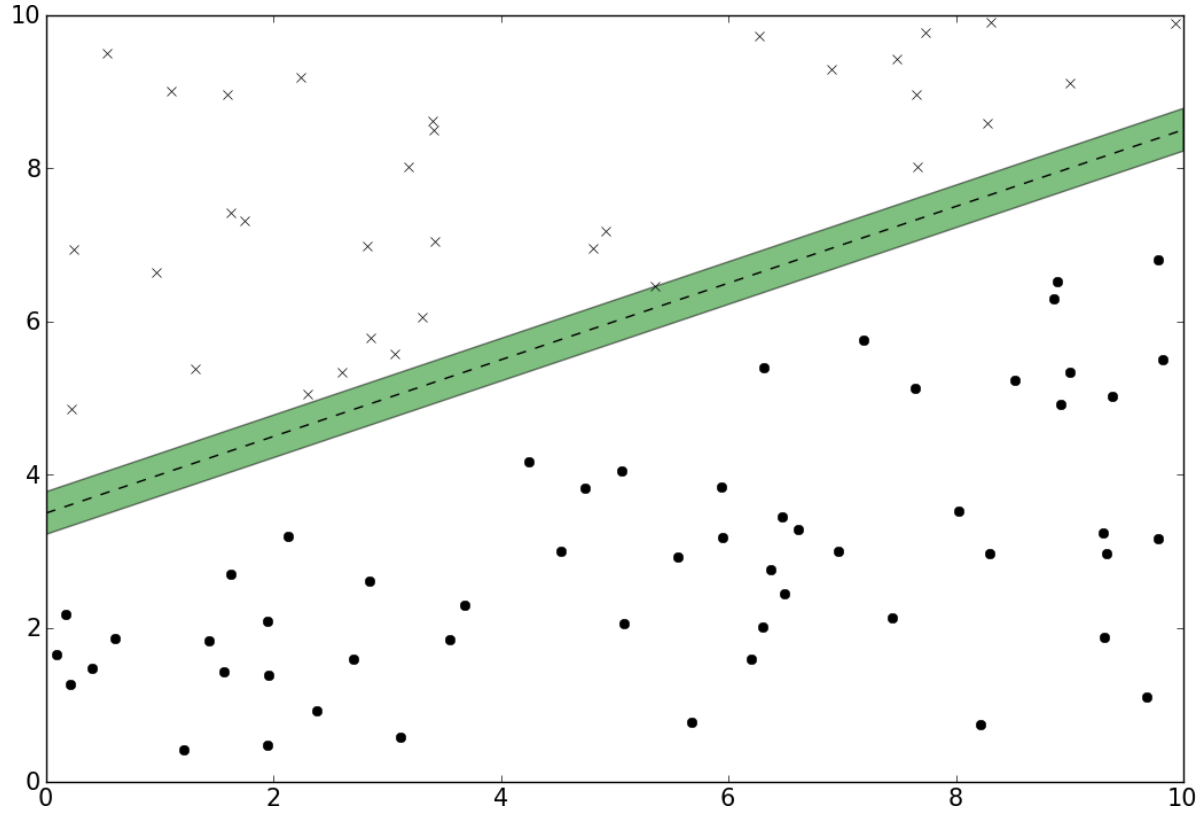
Linear classification problem

Without additional constraints, the problem is ill-posed:



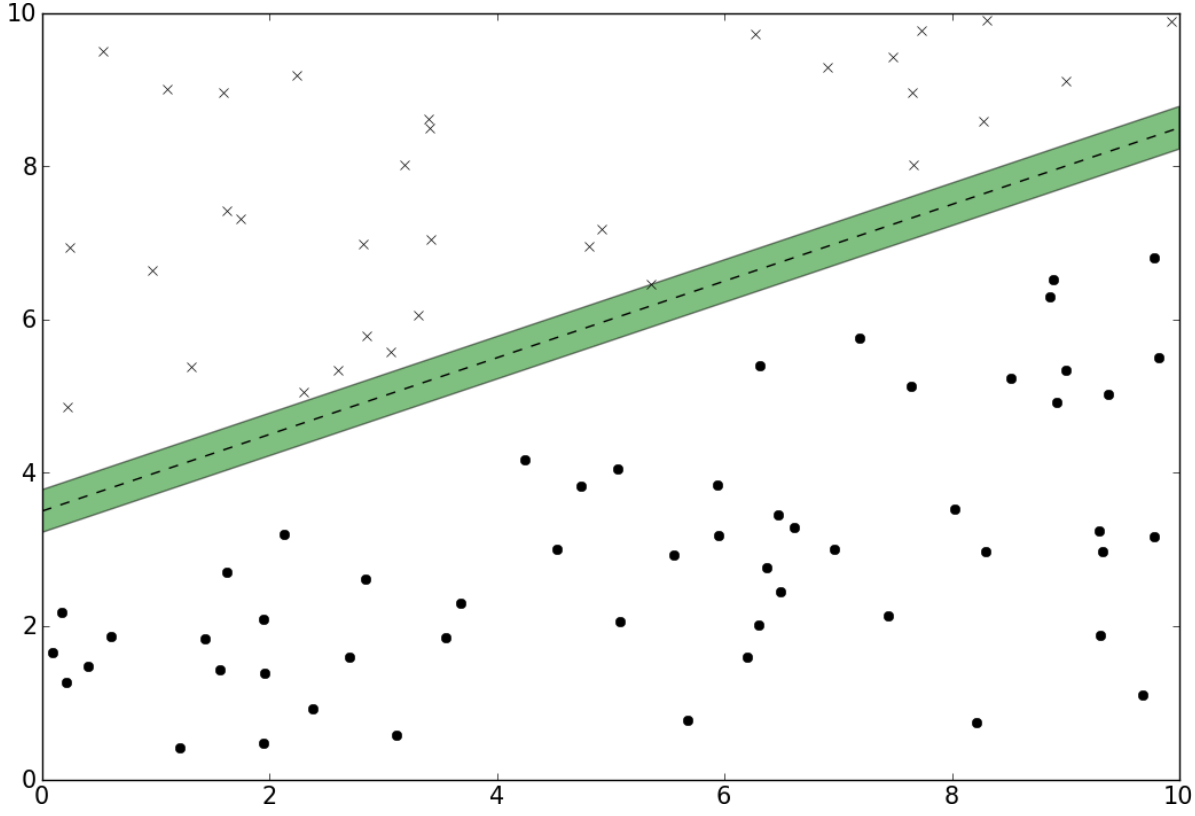
Linear classification problem

Define the margin to be the maximum distance (normal to the classification line) between the convex hulls of the two classes



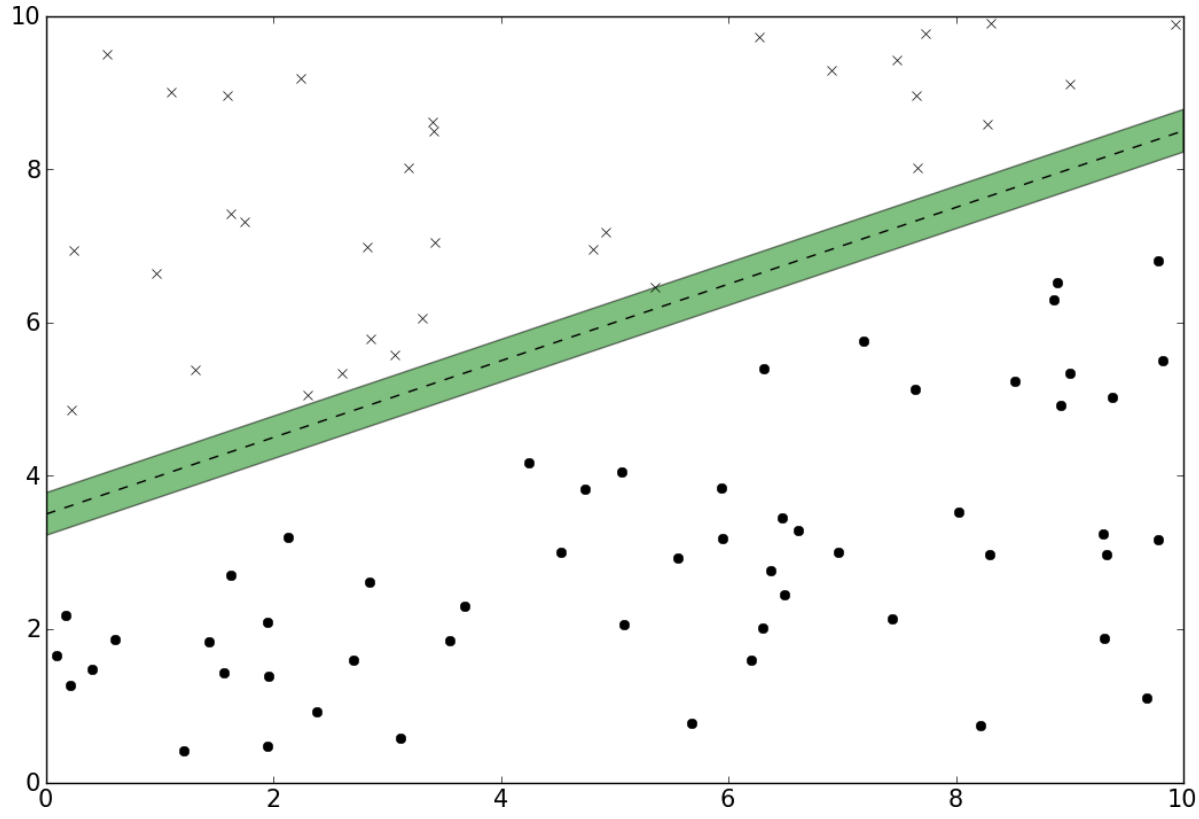
Linear classification problem

Colloquially, the margin is width that the boundary between the categories could be increased by before hitting a data point:



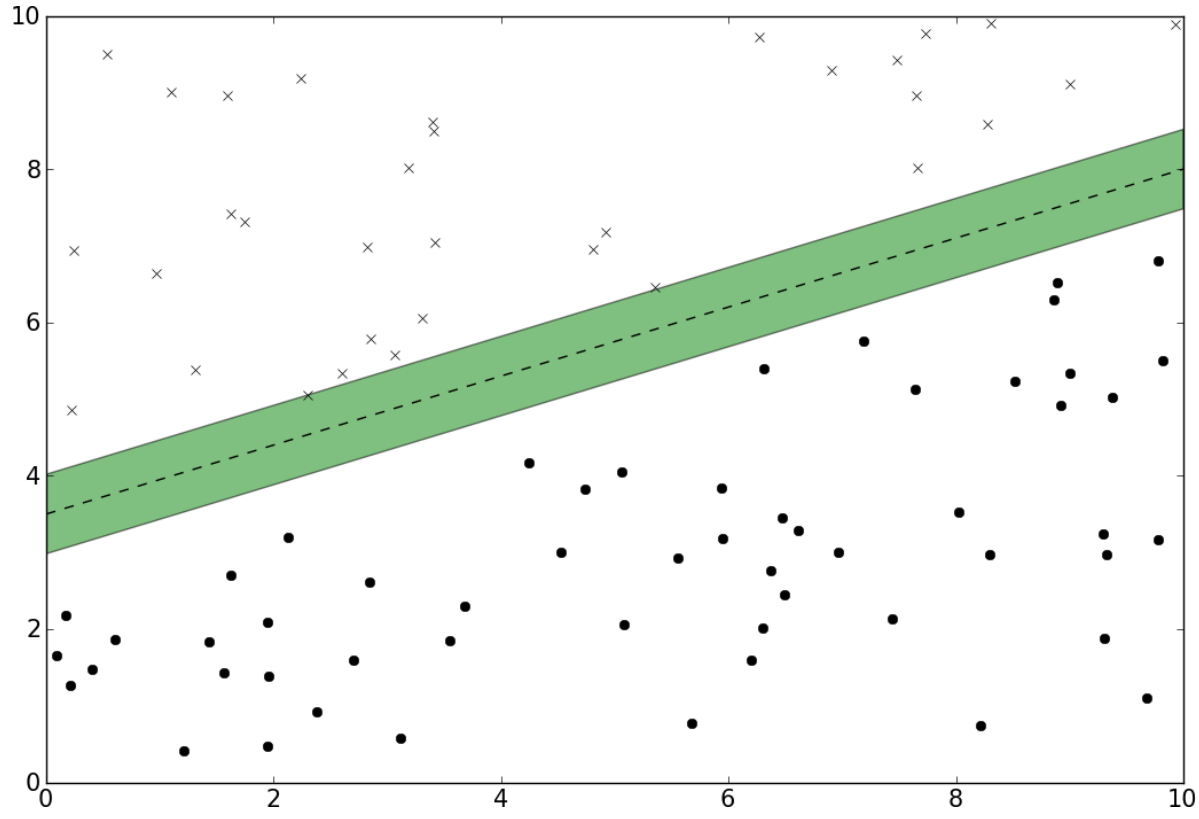
Linear classification problem

Objective: find the line which delimits the two categories with the largest margin:



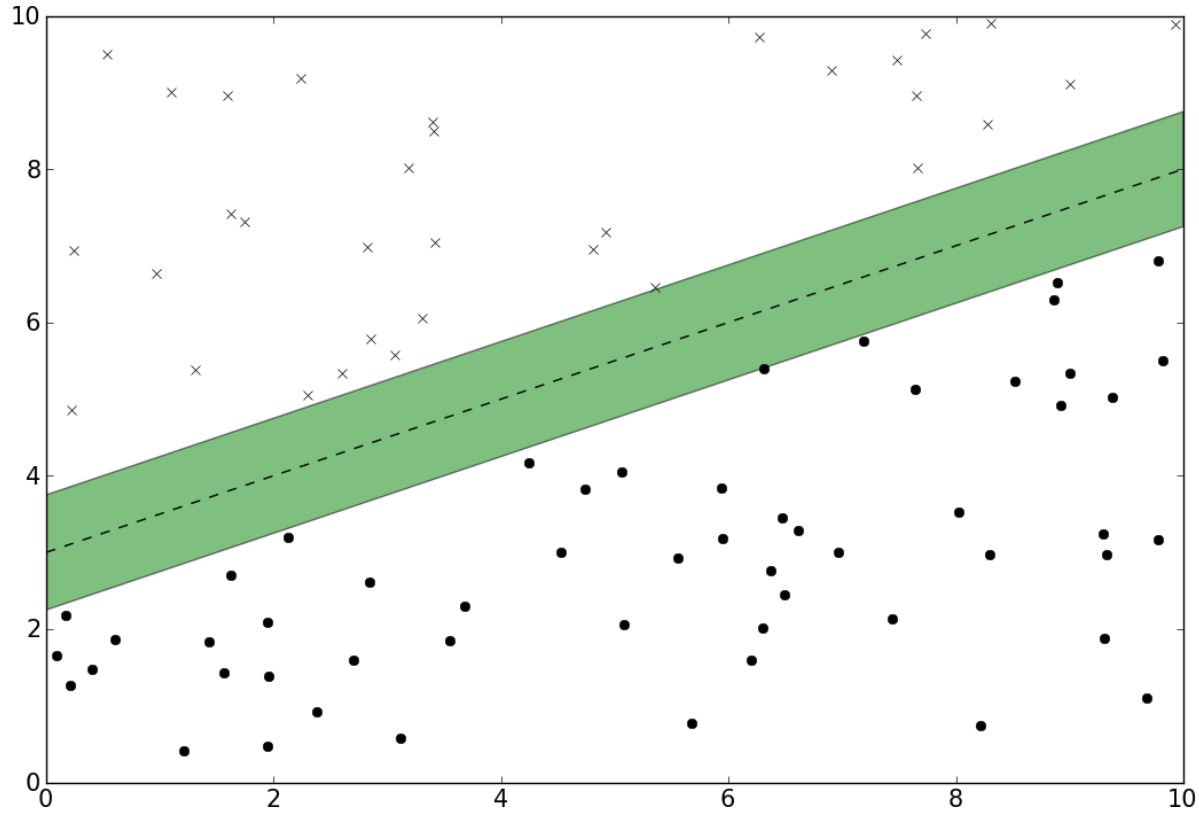
Linear classification problem

Objective: find the line which delimits the two categories with the largest margin:



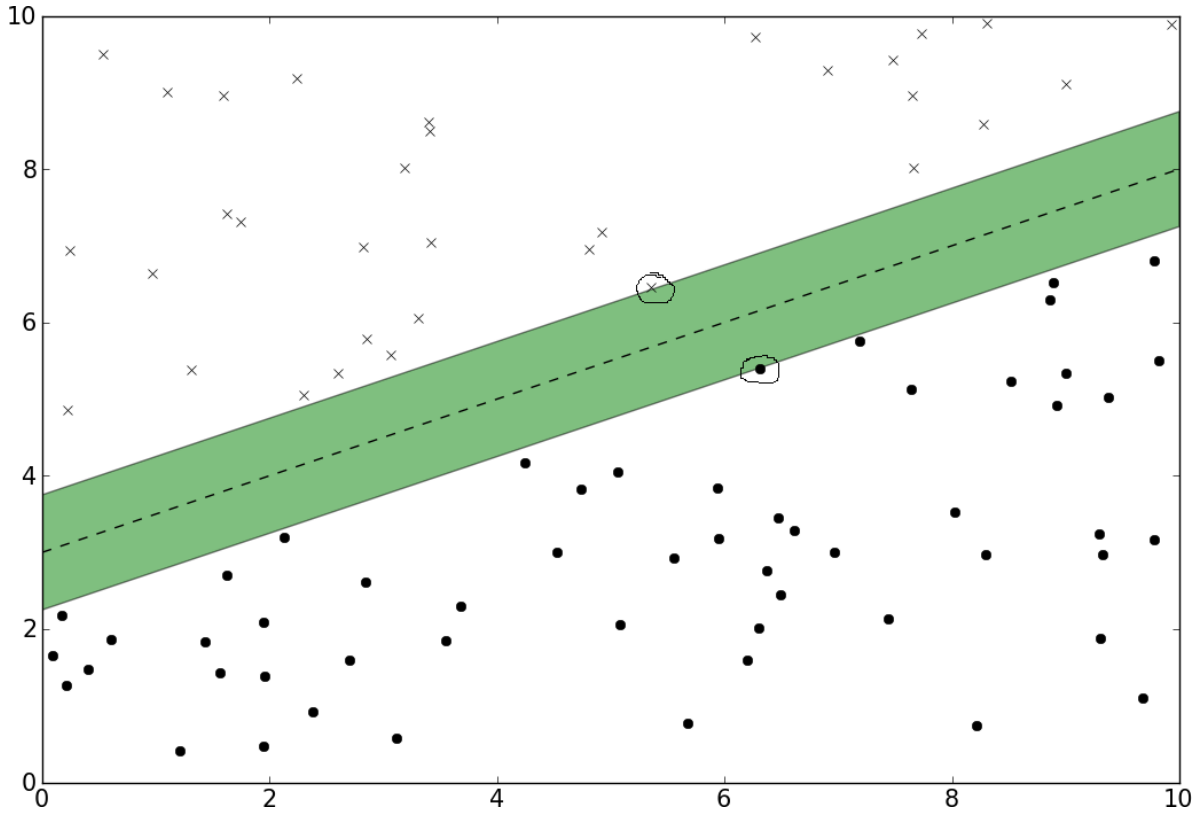
Linear classification problem

Objective: find the line which delimits the two categories with the largest margin:



Linear classification problem

The **support vectors** are the points which, if removed, would change the solution to the linear classification problem:



Linear classification problem

Why maximize the margin?

- Minimizes the risk of misclassification (based on observed data)
- The problem becomes more computationally efficient
- For the simplest cases, a unique solution exists

Linear classification problem

Why maximize the margin?

- Minimizes the risk of misclassification (based on observed data)
- The problem becomes more computationally efficient
- For the simplest cases, a unique solution exists

Linear classification problem

Why maximize the margin?

- Minimizes the risk of misclassification (based on observed data)
- The problem becomes more computationally efficient
- For the simplest cases, a unique solution exists

Linear classification problem

Let $x_i, \ldots, x_n \overset{iid}{\sim} P_1$ with support $\mathbb{R}^m$, and $y_i, \ldots, y_n \overset{iid}{\sim} P_2$ with support $\{-1, 1\}$

- If the observed data S is linearly seperable, then there exists some pair (W, b) such that:

$$W^T x_i + b \geqslant 1 \forall i \ni y_i = 1$$

$$W^T x_i + b \leqslant -1 \forall i \ni y_i = -1$$

which provides the decision rule:

$$f_{W,b}(x) = \text{sgn}\left(W^T x + b\right)$$

Linear classification problem

- Combining inequalities gives $y_i \left(W^T x_i + b\right) \geqslant 1 \forall x_i, y_i$
- WLOG, (W, b) may be scaled such that $\min_i \left|W^T x_i + b\right| = 1$
- So, the problem becomes:

$$\min : \|W\|^2 = W^T W$$
$$\text{s.t.} : y_i \left(W^T x_i + b\right) \geqslant 1 \forall x_i, y_i$$

- This is equivelant to maximizing the margins

Linear classification problem

- So, we want to solve the quadratic program:

$$\min : \frac{1}{2}\|W\|^2$$
$$\text{s.t.} : y_i \left(W^T x_i + b \right) \geqslant 1 \forall x_i, y_i$$

- The global optimum exists (unlike ann)
- Parameters of the QP solver only affect training time, not the quality of the solution

Linear classification problem

- Lagrangian formulation gives:

$$L(W, b, \Lambda) = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{n} \lambda_i \left[y_i \left(W^T x_i + b \right) - 1 \right]$$

$$\Rightarrow \frac{\partial L}{\partial W} = W - \sum_{i=1}^{n} \lambda_i y_i x_i = 0$$

$$\Rightarrow \frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \lambda_i y_i = 0$$

Linear classification problem

- The optimal solution:

$$W^{\dagger} = \sum_{i=1}^{n} \lambda_i^{\dagger} y_i x_i$$

- By substitution:

$$F(\Lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\|W\|^2 = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

- So, we have derived the dual problem:

$$\begin{aligned} Max: \;\; & F(\Lambda) = \Lambda^T I - \tfrac{1}{2}\Lambda^T D \Lambda \\ s.t.: \;\; & \Lambda \geqslant 0, \Lambda^T y = 0 \end{aligned}$$

where $y = (y_1, \ldots, y_n)^T$ and D is symmetric $n \times n$ with $D_{ij} = y_i y_j x_i^T x_j$.

Linear classification problem

- The Lagrange multipliers λ_i are only non-zero when $y_i \left(W^T x_i + b\right) = 1$
- The x_i such that this hold are called **Support Vectors**, and lie closest to the seperating hyperplane
- So we have the optimal weights:

$$W^\dagger = \sum_{i=1}^{n} \lambda_i^\dagger y_i x_i$$

- with bias:

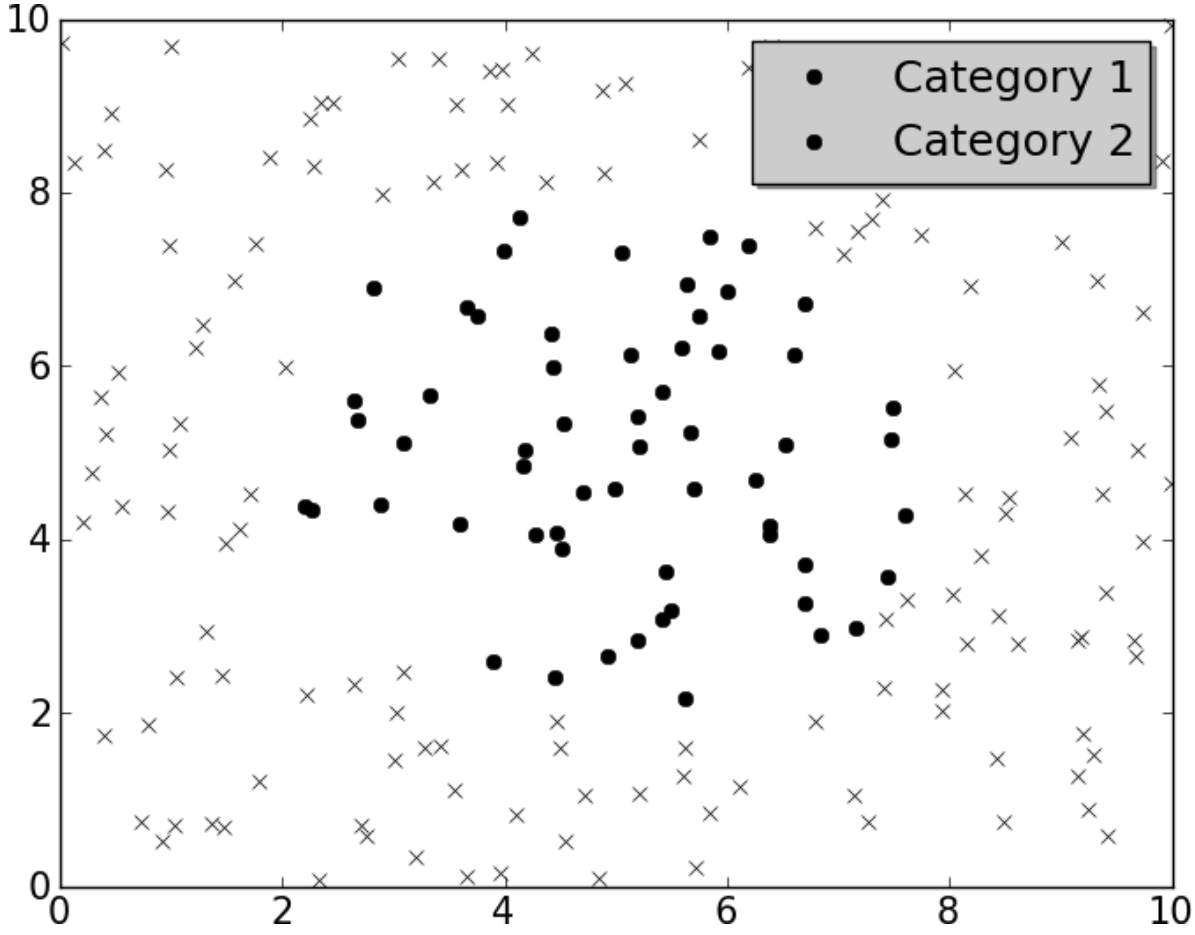
$$b^\dagger = y_i - \left(W^\dagger\right)^T x_i$$

- And decison (classification) rule:

$$f\left(x\right) = \text{sgn}\left(\sum_{i=1}^{n} y_i \lambda_i^\dagger x^T x_i + b^\dagger\right)$$

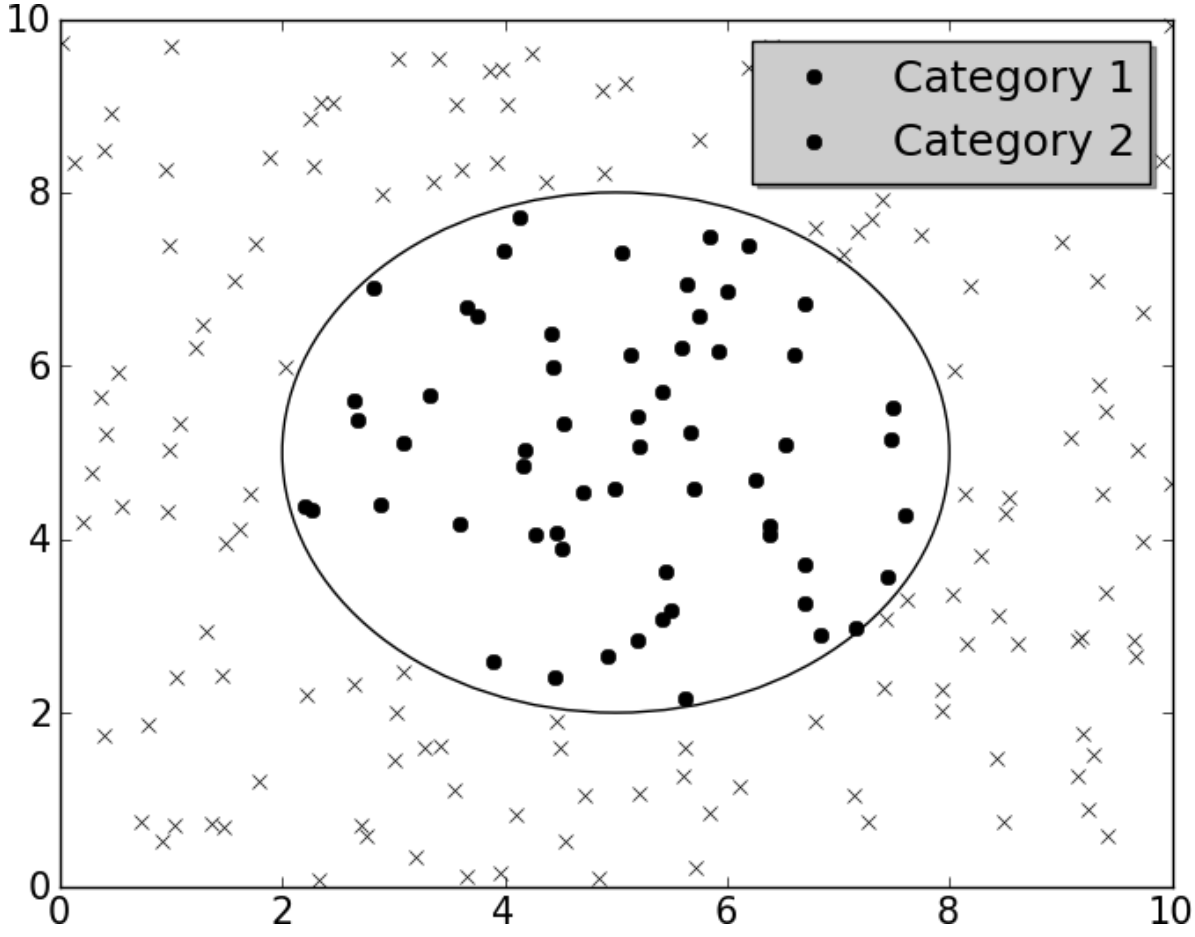
Non-Linear classification problem

- Suppose now that the data does not lend itself to linear classification:



Non-Linear classification problem

- Suppose now that the data does not lend itself to linear classification:



Non-Linear classification problem

- Strategy: map the data from the original observation space X to some **feature space** H, where the data will become linearly seperable:

$$X \to H$$

$$x \mapsto \phi(x)$$

- A linear SVM is performed; giving the decision rule:

$$f(x) = \text{sgn}\left(\phi(x)^T W^\dagger + b^\dagger\right)$$

$$= \text{sgn}\left(\sum_{i=1}^{n} y_i \lambda_i^\dagger \phi(x)^T \phi(x_i) + b^\dagger\right)$$

- Note that this only depends input data through inner products in H

Non-Linear classification problem

- Define a **Kernel function**: $K(x, z) = \phi(x)^T \phi(z)$
- 'SVM Kernel Trick': Allows for construction of optimal hyperplane in H without explicitly transforming data into H
- Non-linear decision rule:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \lambda_i^\dagger K(x, x_i) + b^\dagger\right)$$

- What constitutes an admissible kernel?

Non-Linear classification problem

Mercer's Theorem

Suppose $K \in L_\infty\left(X^2\right)$ *is a symmetric real-valued function such that the integral operator:*

$$T_k : L_2(X) \to L_2(X)$$

$$\left(T_k f\right)(x) = \int_{X^2} K(x,z)\, f(z)\, d\mu(z)$$

is positive-definite. Then, for non-increasing eigenvalues λ_i *and orthonormal eigenfunctions* ψ_i*:*

$$K(x,z) = \sum_i \lambda_i \psi_i(x) \psi_i(z)$$

with convergence absolute and uniform.

Non-Linear classification problem

Proposition

If K is a kernel satisfying the preconditions of Mercer's theorem, we can construct a mapping ϕ to some space where K acts as an inner product:

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

for almost all $x, z \in X$.

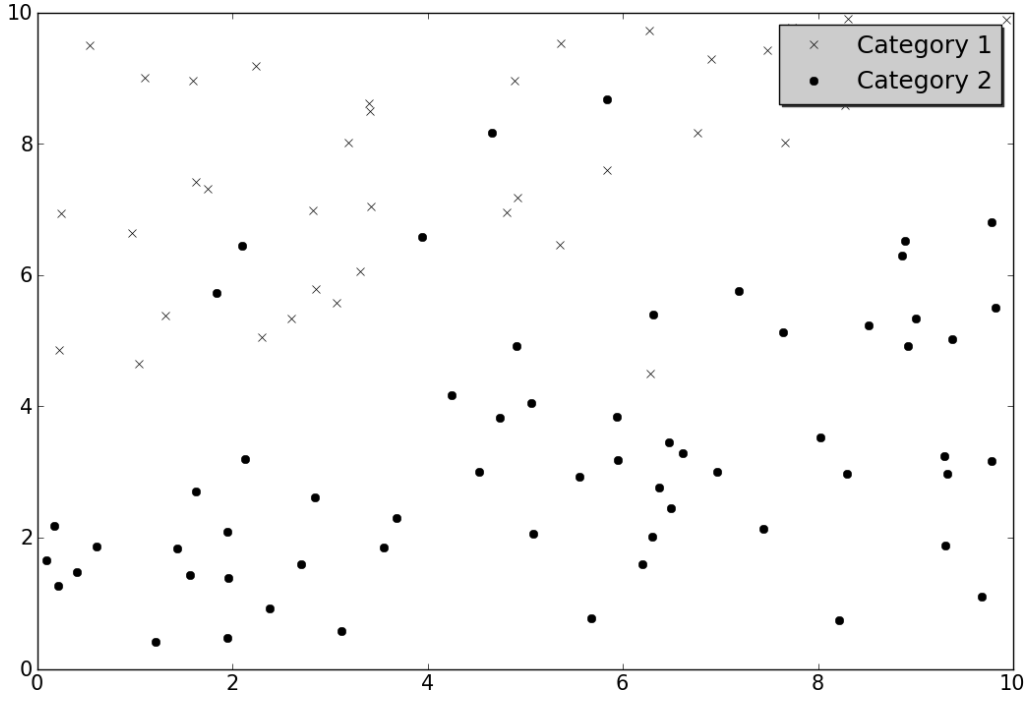
Non-Linear classification problem

Examples:

- **Polynomial**: $K(x, z) = \langle x, z \rangle^p, p > 0$
- **Gaussian/RBF**: $K(x, z) = \exp\left\{-\gamma \|x - z\|^2\right\}, \gamma > 0$
- **Sigmoid**: $K(x, z) = \tanh\left(\kappa x^T x + \nu\right), \kappa > 0, \nu < 0$
- With a suitable choice of kernel parameters, an SVM can seperate any consistent data set
- Regularization (i.e. margin maximization) prevents overfitting common in data mining, neural networks, etc

Non-seperable Data

- So far we have assumed complete seperability via kernels
- Consider the case of noisy or inconsistent data (even with kernels):



Non-seperable Data

- Trying to force zero training error will lead to a model with poor generalization
- Regularize further: Introduce tradeoff between fidelity to the data and tractability of the solution
- Introduce slack variables to the problem: $\Xi = (\xi_1, \ldots, \xi_n)^T$
- Problem becomes:

$$\min : \frac{1}{2}\|W\|^2 + c\sum_{i=1}^{n}\xi_i$$

$$s.t. : y_i\left(W^T\phi(x_i) + b\right) \geqslant 1 - \xi_i, \xi_i \geqslant 0, i = 1, \ldots, n$$

- where c is the regularization parameter

Non-seperable Data

- Lagragian has the form:

$$L = \frac{1}{2}\|W\|^2 + \sum_{i=1}^{n} \lambda_i \left(y_i \left(W^T \phi(x_i) + b \right) - 1 + \xi_i \right)$$

$$- \sum_{i=1}^{n} \gamma_i \xi_i + c \sum_{i=1}^{n} \xi_i$$

- so we must have:

$$\frac{\partial L}{\partial W} = W - \sum_{i=1}^{n} \lambda_i y_i \phi(x_i) = 0$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \lambda_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = c - \lambda_i - \gamma_i = 0$$

Non-seperable Data

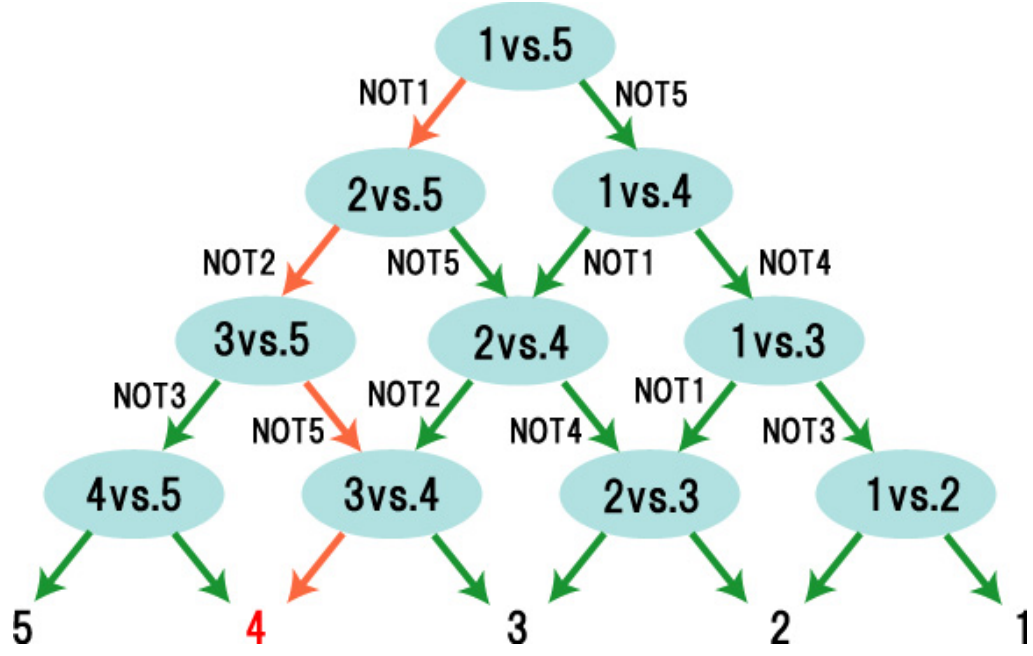
- This gives the dual formulation:

$$Max: \ F(\Lambda) = \Lambda^T I - \frac{1}{2}\Lambda^T D \Lambda$$
$$s.t.: \ 0 \leqslant \Lambda \leqslant c, \Lambda^T y = 0$$

- with $D_{ij} = y_i y_j K(x_i, x_j)$
- Here x_i is a support vector if $0 < \lambda_i < c$
- The decision function is just as before

Multi-class SVM

- In the case of N classes to be delimited via SVM, a series of $\frac{N(N-2)}{2}$ sub-SVMs must be trained:



Practical Considerations

To summarize the features of SVM:

- The more seperable the data set, the more sparse the representation of the decision function
- Once trained, even non-linear SVM can classify new data points relatively quickly via kernels
- Regularization works to prevent overfitting, provide a generalizable solution given noisy data

Practical Considerations

To implement an SVM using a given data set, one must choose:

- A method for encoding data from the observation space in a reasonably seperable manner for the SVM
- An appropriate seperating kernel K
- Values for any parameters of the kernel
- A value for the regularization parameter c

Methods for determining optimal choices of the above are active areas of current research

Practical Considerations

- Research has shown the Gaussian/RBF kernel to generalize particularly well
- Choice of γ, c are typically conducted via grid search
- Objective: maximize **cross-validation** on the training data
  - Randomly seperate the training data into K partitions, one of which used for validation and the remaining for training
  - The SVM is constructed, and accuracy is determined
  - Process is repeated, using each subset once as the validation set
  - 'Nearly-unbiased' estimator of fitness

Practical Considerations

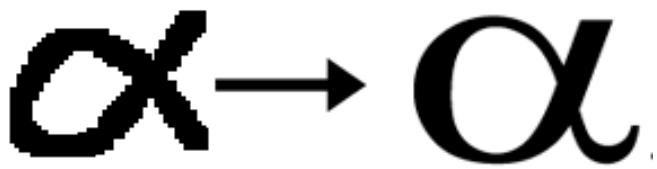
Existing implementations:

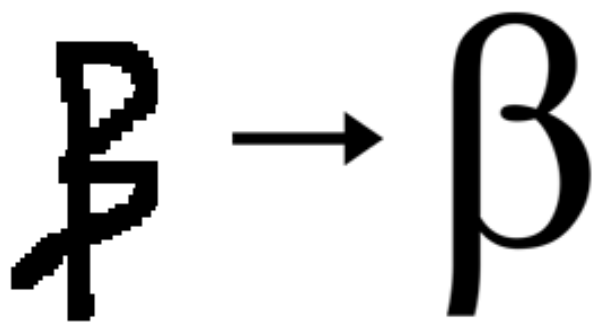
- libSVM (C++,Python, MATLAB,Java)
- SVMlite (C, Python, MATLAB, Java)
- Shogun (C++, Python, R, Octave)
- flssvm (Fortran)
- Shark (C++)

Handwriting Recognition

Consider a (simplified) problem of handwriting recognition:

- Want to find a method for accurately recognizing letters of the greek alphabet:





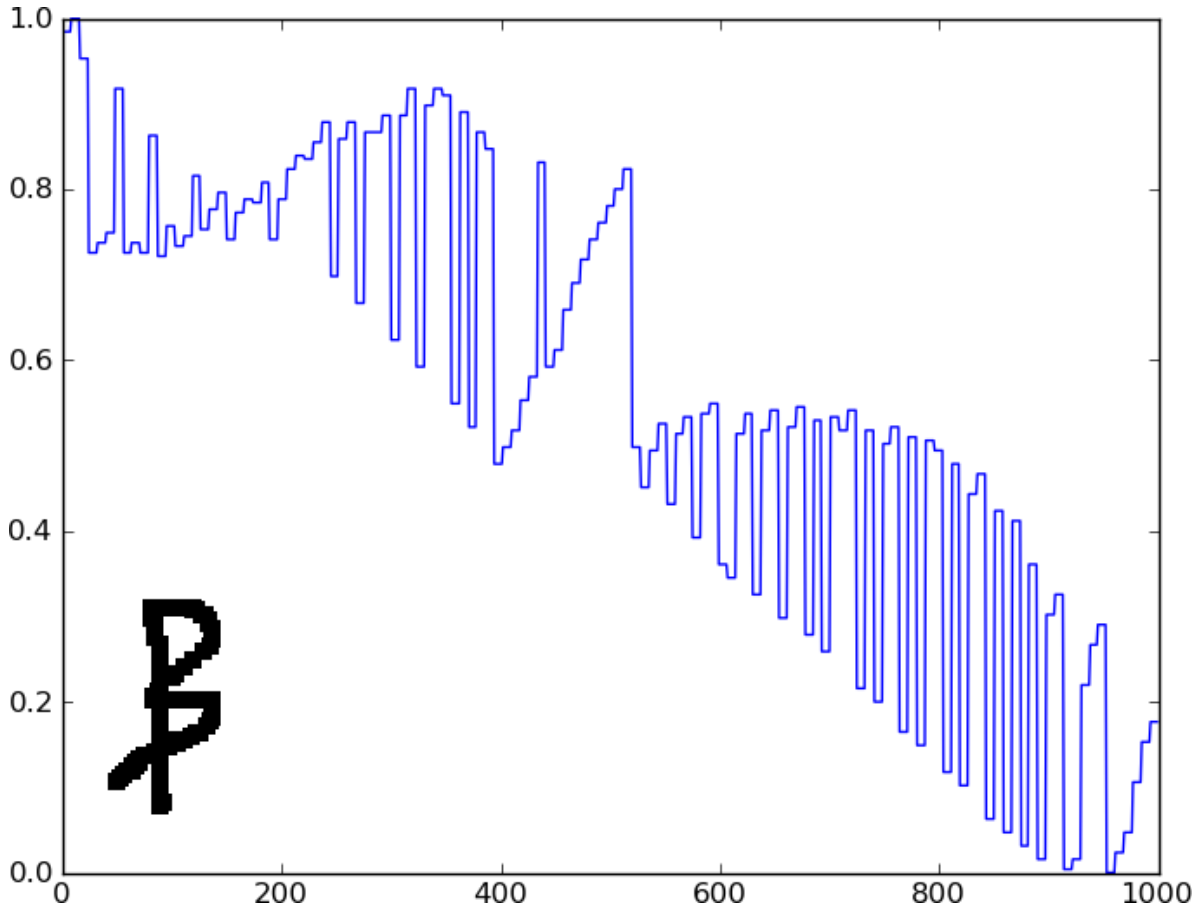
Handwriting Recognition

Consider a (simplified) problem of handwriting recognition:

- A tablet PC application records a series of (x, y) coordinates corresponding to a written charector
- For SVM encoding, each charector is encoded a vector with elements $z_i = x_i^2 + y_i^2$, where i is ordered with respect to the x_is (left to right)
- Each charector vector is downsampled/extended to have exactly 1000 components

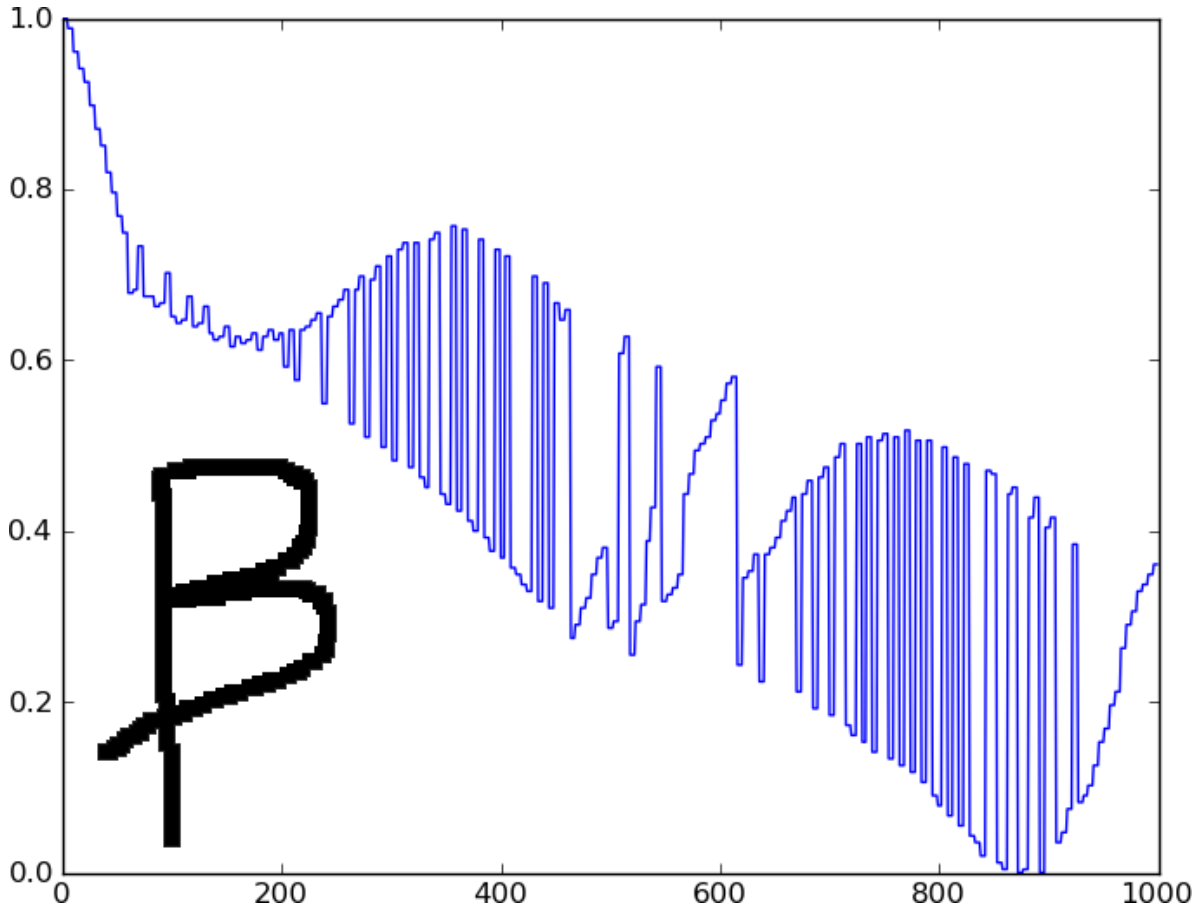
Handwriting Recognition

Example:



Handwriting Recognition

Example:



Handwriting Recognition

- Training samples were then collected for each greek letter, and used to construct an SVM (composed of $\frac{24 \cdot 22}{2} = 264$ sub-SVMs)
- Kernel function: Gaussian/RBF
- $C = 7.10, \gamma = 0.01$

(Run program)

Cognitive State Monitoring

NASA Glenn Research Center Cognitive State Monitoring Laboratory uses EEG and fNIRS to perform research in brain-computer interface technology
Potential applications:

- Monitor human Performance and Cognition in extreme environments
- Biofeedback for self-training, craving control
- Brain-computer interface for prosthetic activation or locked-in communication
- Cognitive assessment for Alzheimers disease
- Brain-controlled gaming applications

Cognitive State Monitoring

Key technologies:

EEG

fNIRS

Cognitive State Monitoring

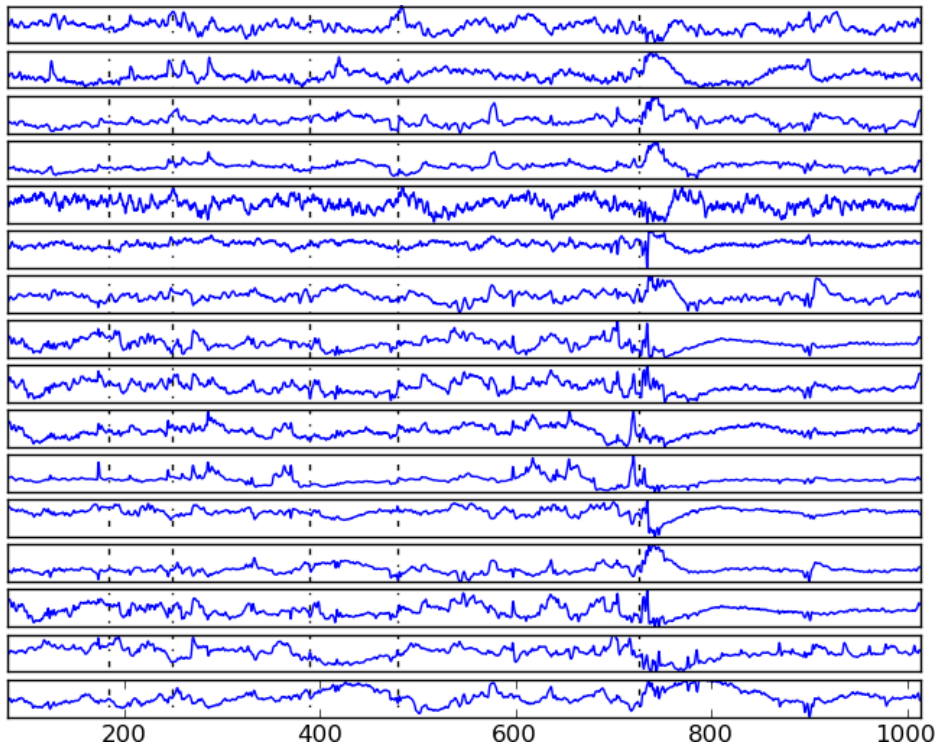
fNIRS stands for **functional Near InfraRed Spectroscopy**:

- Uses near-infared (650nm, 730nm) lasers to penetrate into the skull
- The two wavelengths are absorbed by different species of blood hemoglobin
- The light scatters and is detected at the surface near the emitters
- emitted vs. detected light allow for computation of blood hemoglobin species concentration

Cognitive State Monitoring

Task acquisition trial:

- Seeks to determine if the effectiveness of a training regimen detectable using fNIRS
- Test subjects each played a level of GoldenEye 007 for Nintendo 64 (followed by a rest period) for 5 successive trials



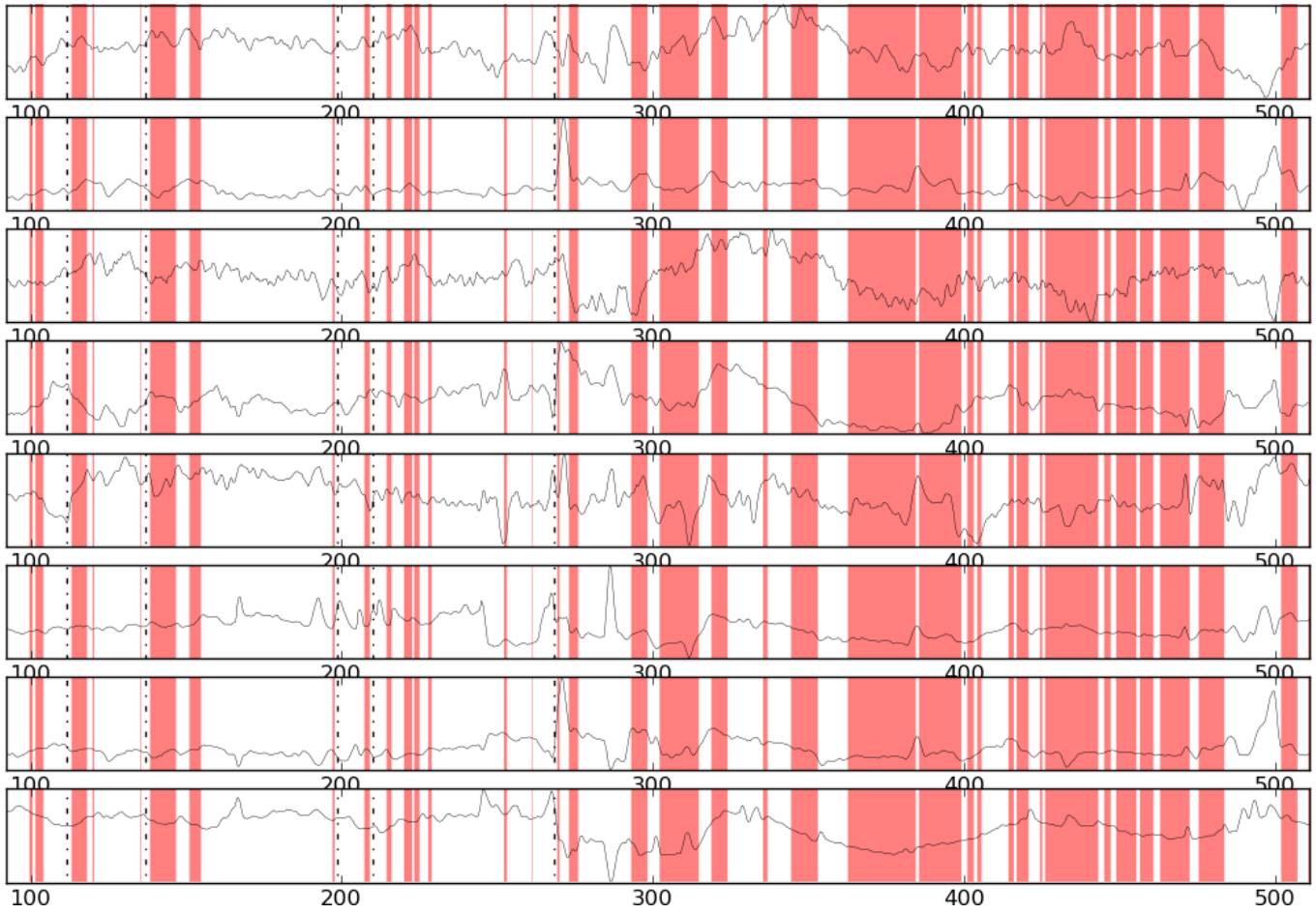
Cognitive State Monitoring

- Exploratory analysis with SVMs: using the first trial run, train an SVM to delimit between the 'Active' and 'Resting' phases of the trial
- Use the SVM to analyze the subsequent trial runs of the same subject to determine if skill acquisition is detectable via fNIRS/SVM

Cognitive State Monitoring

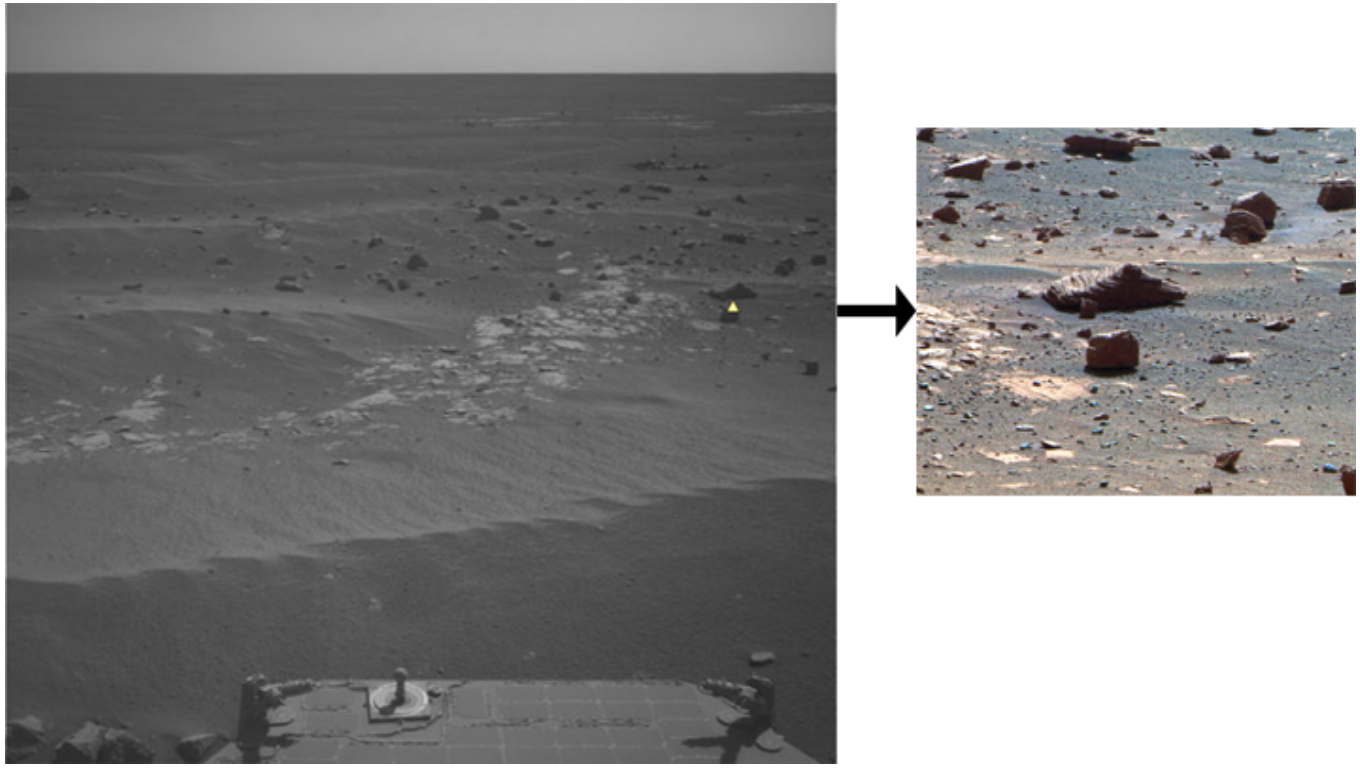
Results:

- Using RBF kernel, an SVM was trained on the first trial. The temporal classification of the final trial:



Autonomous Systems

- Provide limited situational awareness and autonomous decision making capabilities to robotic systems, etc
- Working SVM example: NASA JPL's AEGIS Project



Autonomous Systems

Other possibilities for robotics:

- Human face detection/recognition
- Automatic navigation
- Sound analysis

Speed of SVM classification allows for near real-time application: (run example)

Colin Campbell.

Kernel methods: a survey of current techniques.

Neurocomputing, 48(1-4):63–84, October 2002.

Kristin P. Bennett and Colin Campbell.

Support vector machines: hype or hallelujah?

SIGKDD Explor. Newsl., 2(2):1–13, 2000.

Alex J. Smola and Bernhard Schlkopf.

A tutorial on support vector regression.

Statistics and Computing, 14(3):199–222, 2004.

Christopher J.C. Burges.

A tutorial on support vector machines for pattern recognition.

Data Mining and Knowledge Discovery, 2(2):121–167, June 1998.

Questions?