



# Support Vector Machines

Theory & Applications

Philipp Schmidt  
July 2018



# Outline for today

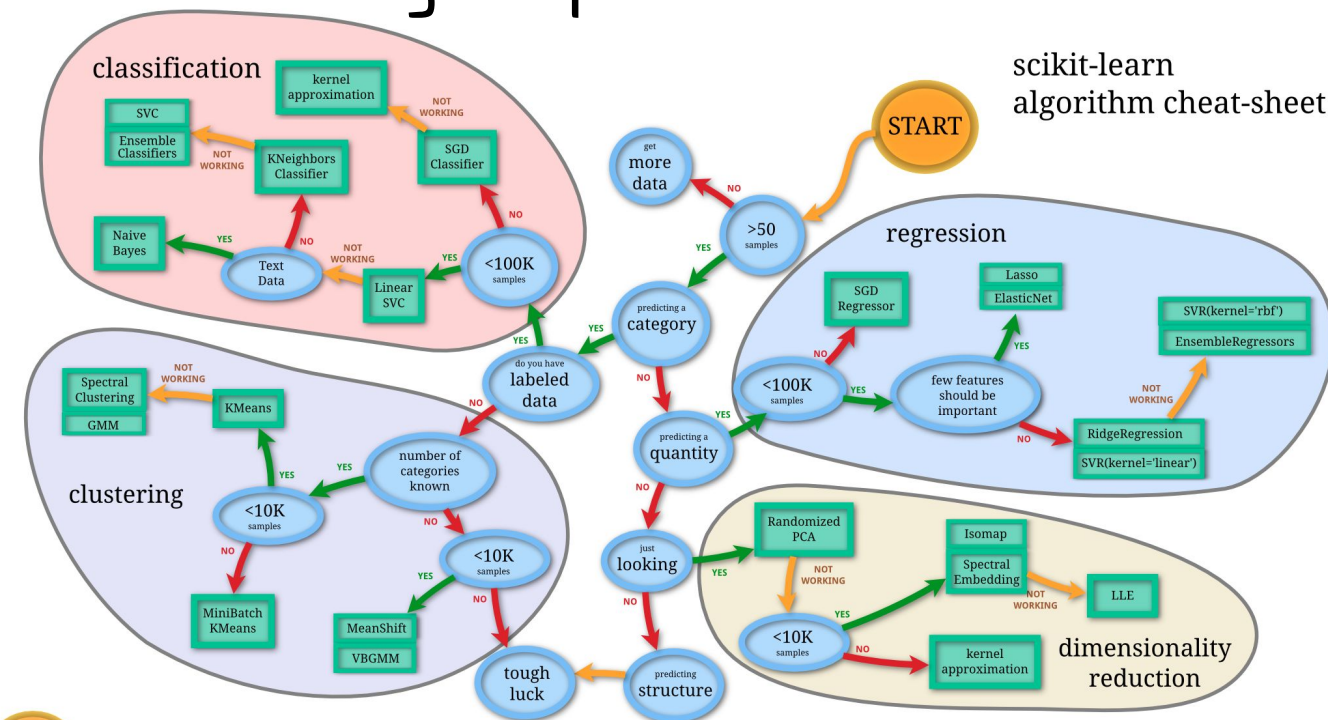
- Organization & Setup
- How to separate data?
- Discriminative functions
- Perceptron Algorithm + Exercise
- Support Vector Machine + Exercise
- **Lunch Break**
- Kernel Support Vector Machine
- Data Project
- Wrap-up

# Organization & Setup

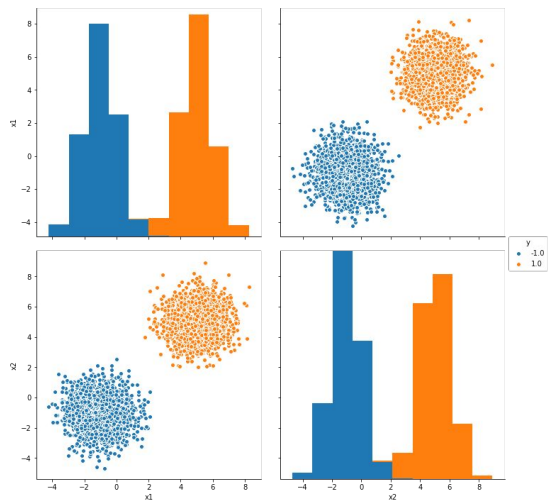
- Slides and code for exercises can be found on GitHub:
  - <https://github.com/tdhd/data-science-retreat-svm>
- Recommended IDE for working with Python:
  - <https://www.jetbrains.com/pycharm/download>
- Create conda environment from *env.yml*.

**Need help setting things up? Please ask me.**

# Machine Learning Map - Where are we?

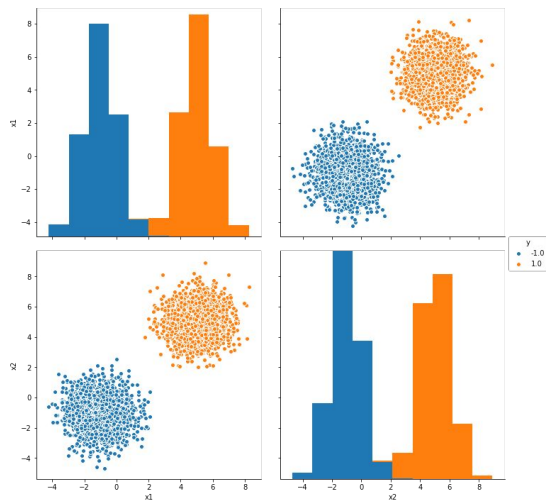


# How to separate data?



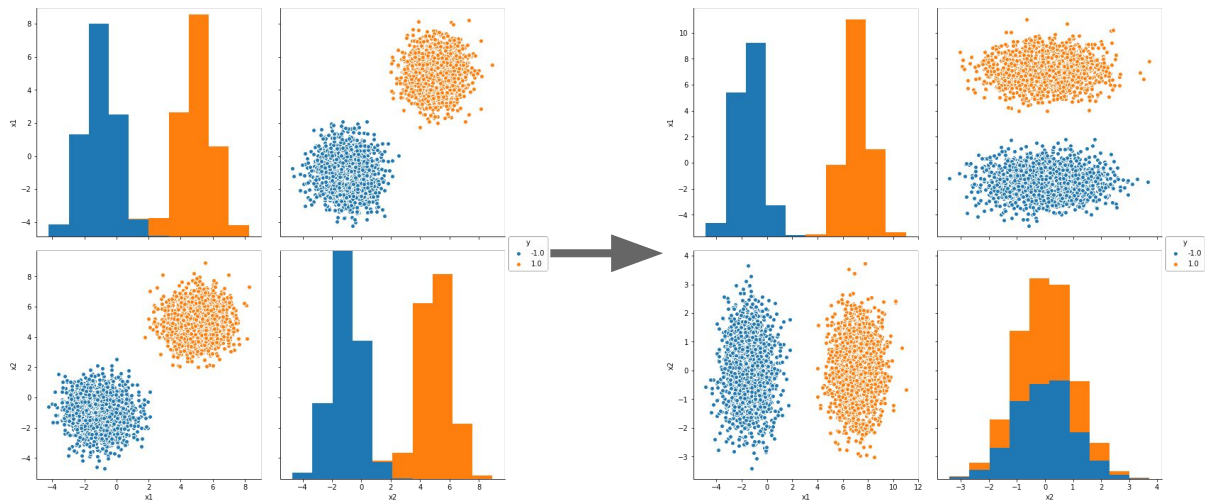
# How to separate data?

- Projections of data onto X and Y axis, have overlap.
- Linear coordinate transformation:
  - 45 degree rotation.



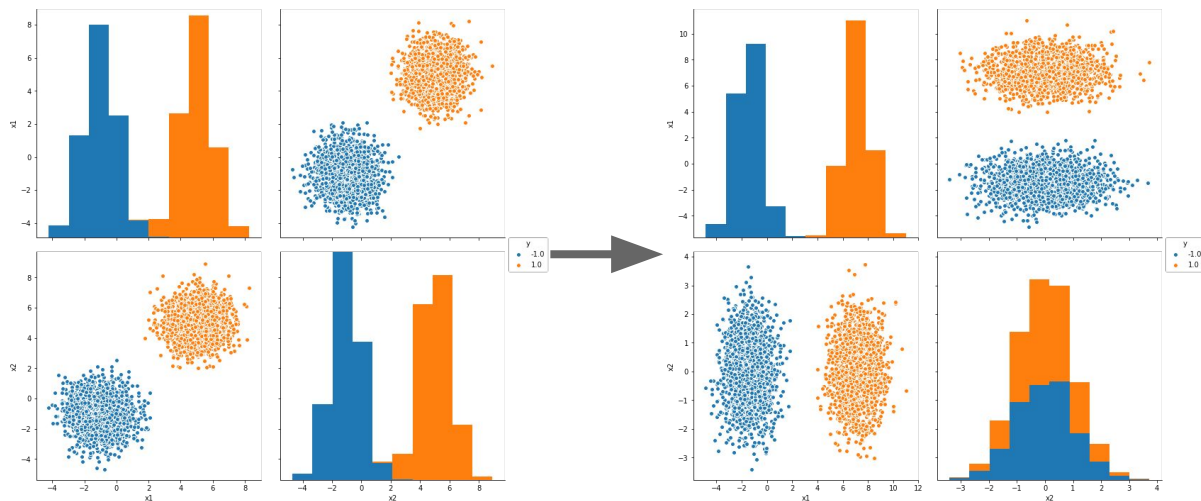
# How to separate data?

- Projections of data onto X and Y axis, have overlap.
- Linear coordinate transformation:
  - 45 degree rotation.

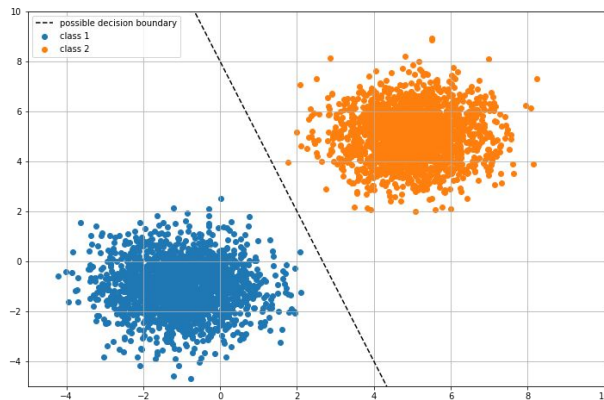


# How to separate data?

- Projections of data onto X and Y axis, have overlap.
- Linear coordinate transformation:
  - 45 degree rotation.



## Possible decision boundary

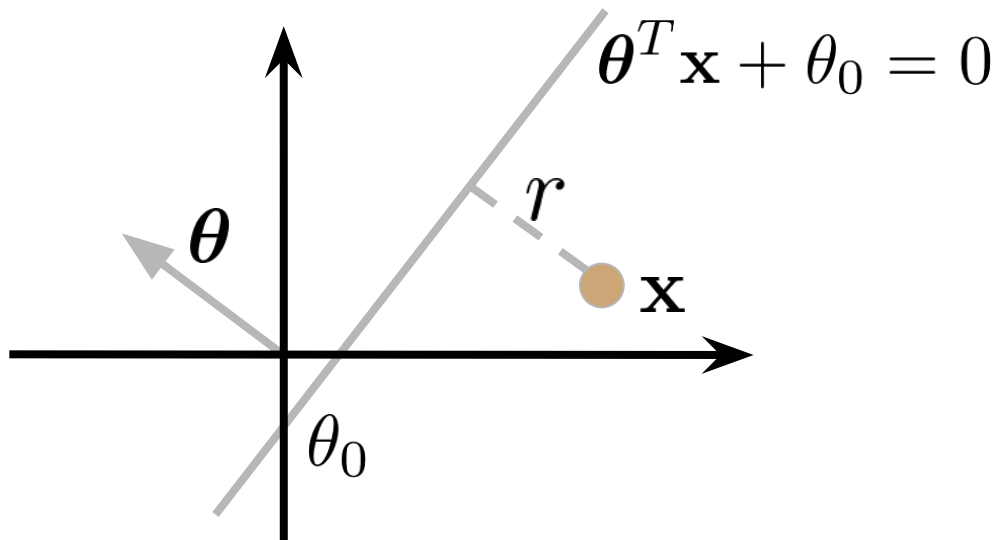




# Geometry recap

- Hyperplane defined by  $\theta^T \mathbf{x} + \theta_0 = 0$
- Signed (perpendicular) distance from hyperplane with bias to a point  $\mathbf{x}$  :

$$r = y \frac{(\theta^T \mathbf{x} + \theta_0)}{\|\theta\|}$$



# Learning discriminative functions

$$f_{\theta}(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{Y}$$

- Learn a mapping from input space to output space.
- Domain  $\mathcal{X}$  is typically  $\mathbb{R}^d$ .
- Target  $\mathcal{Y}$  for binary classification is typically  $\{-1, 1\}$ .
- Given Data  $\mathbf{X} \in \mathbb{R}^{n,d}$  and labels  $\mathbf{y} \in \{-1, 1\}^n$ 
  - Learn parameters of function  $f$ .

# How to learn discriminant functions from data?

Given labelled data instances:

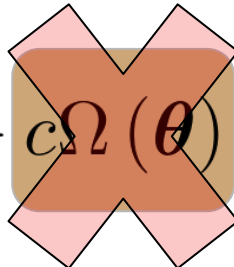
- Learn functions that:
  - **minimize the empirical risk** (perform well on given data).
    - Loss function defines goodness-of-fit.
  - **generalize** to unseen data (control model complexity).
    - Regularized objective functions can account for model complexity.
- Generic framework to do that:
  - Regularized Empirical Risk Minimization.

# Regularized Empirical Risk Minimization

$$L(\boldsymbol{\theta}) = \sum_{i=1}^n \underbrace{l(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)}_{\text{Empirical Risk}} + \underbrace{c\Omega(\boldsymbol{\theta})}_{\text{scaled Regularizer}}$$

- Solve  $\arg \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ 
  - Find theta, that minimizes both the data-loss and model complexity.
- Can be solved by gradient descent  $\nabla L(\boldsymbol{\theta})$ .
  - Component wise partial derivative of parameter vector.

# Perceptron algorithm

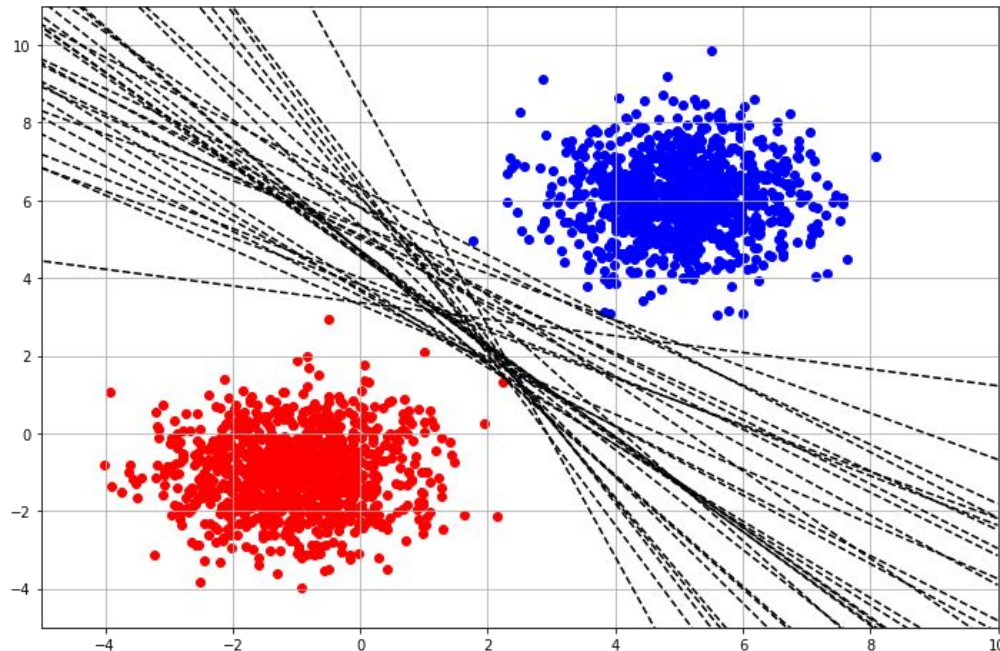
$$L(\boldsymbol{\theta}) = \sum_{i=1}^n l(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + c \Omega(\boldsymbol{\theta})$$


- Invented 1957 by Frank Rosenblatt.
- **No regularizer:**
  - Finds any separating hyperplane, if possible.
- Perceptron loss function incurs loss for every misclassified data point:

$$l(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \max(0, -y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0))$$

# Perceptron algorithm

- Finds any separating hyperplane.
- Solution depends on initialization of model parameters.



# Perceptron algorithm - Exercise

```
argument:
X := {x1, . . . , xm}  $\subset$  X (data)
Y := {y1, . . . , ym}  $\subset$  { $\pm 1$ } (labels)
function (theta) = Perceptron(X,Y)
  initialize theta = 0 # includes intercept
  repeat
    Pick (xi, yi) from data
    if yi(theta  $\cdot$  xi)  $\leq$  0 then
      theta = theta + yi  $\cdot$  xi
    end
  until yi(theta  $\cdot$  xi)  $>$  0 for all i
end
```

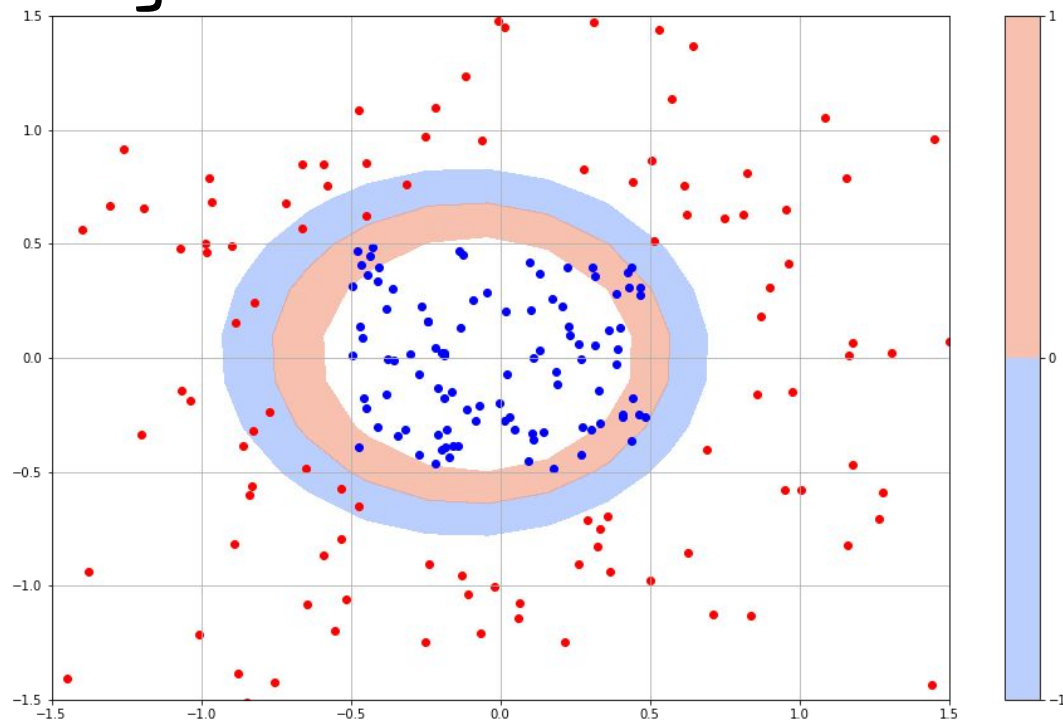
# Perceptron algorithm

- Possible shortcomings of this method?
  - Will converge for any separating hyperplane.
- What happens if data not linearly separable?
  - Will not terminate if data not linearly separable.



# Non-linear Perceptron algorithm

- Non-linear decision boundaries possible?
- Solve classification problems even when data not linearly separable.
- Define feature mapping function explicitly.



$$l(f_{\theta}(\mathbf{x}_i), y_i) = \max(0, -y_i(\theta^T \Phi(\mathbf{x}_i) + \theta_0))$$

# Non-linear Perceptron algorithm - Exercise

```
argument:
X := {x1, . . . , xm}  $\subset$  X (data)
Y := {y1, . . . , ym}  $\subset$  { $\pm 1$ } (labels)
function (theta) = Perceptron(X,Y, $\Phi$ )
    initialize theta = 0 # includes intercept
    repeat
        Pick (xi, yi) from data
        if yi(theta  $\cdot$   $\Phi$ (xi))  $\leq$  0 then
            theta = theta + yi  $\cdot$   $\Phi$ (xi)
        end
    until yi(theta  $\cdot$   $\Phi$ (xi))  $>$  0 for all i
end
```

# Non-linear Perceptron Algorithm

- Possible shortcomings?
  - Explicit feature construction expensive.
  - Feature mapping  $\Phi(\mathbf{x})$  might even become infeasible.
  - Non unique solution.
- Solution?
  - Kernels to the rescue!
  - More on that later.

# Linear Support Vector Machine

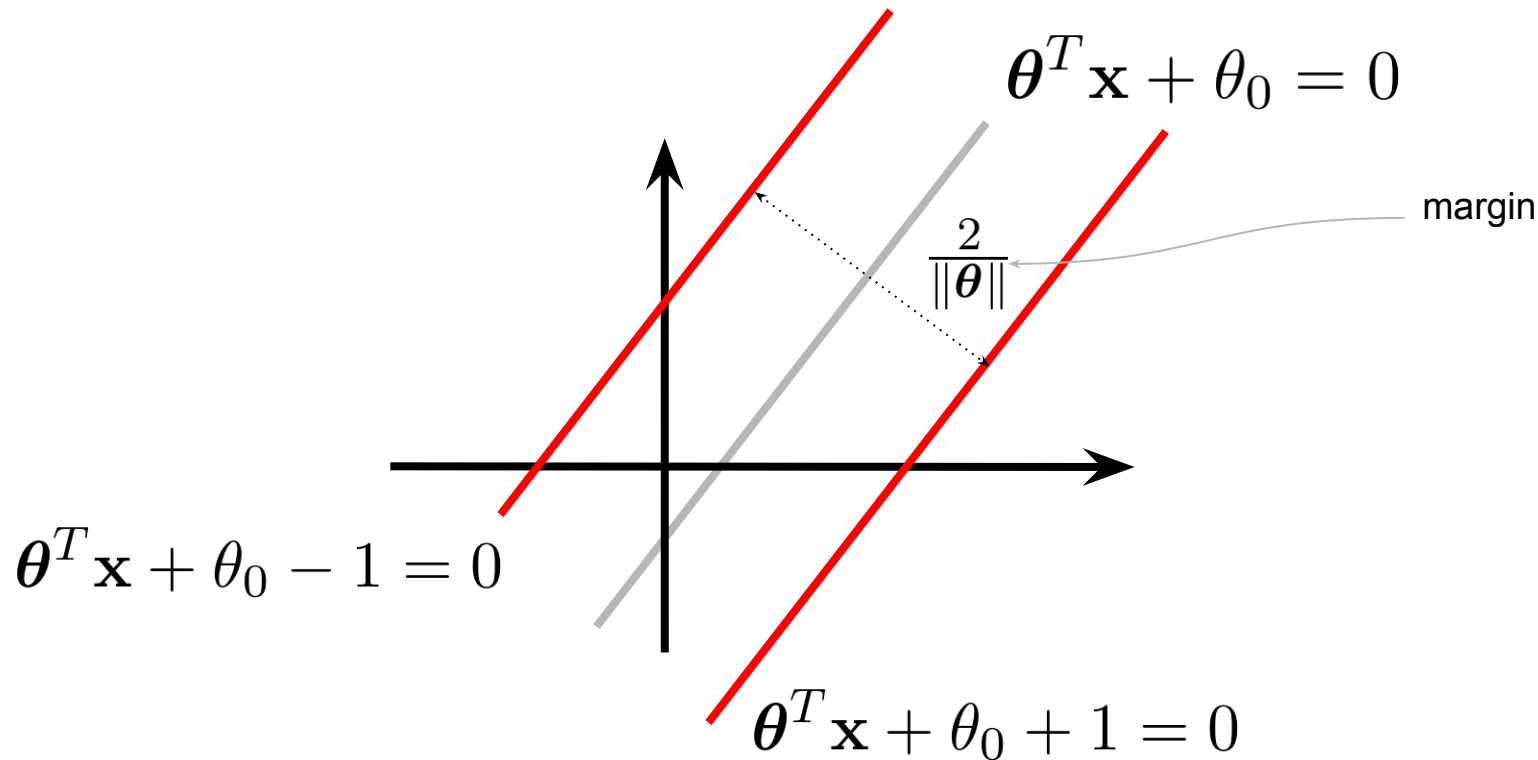
- Adds regularizer.
- SVMs are also called max-margin models, why?
  - Constant scaling  $k$  of hyperplane will result in the same discriminative function.
    - Freedom to set margin hyperplanes at signed distances  $+1, -1$  from decision boundary.

$$\boldsymbol{\theta}^T \mathbf{x} + \theta_0 - 1 = 0$$

$$\boldsymbol{\theta}^T \mathbf{x} + \theta_0 + 1 = 0$$

- The width of the geometric margin is  $\frac{2}{\|\boldsymbol{\theta}\|}$ , why?
  - Compute distance for any point on one of the margin hyperplanes as  $\frac{(\boldsymbol{\theta}^T \mathbf{x} + \theta_0)}{\|\boldsymbol{\theta}\|} = \frac{1}{\|\boldsymbol{\theta}\|}$
  - The margin is twice as wide.

# Support Vector Machine - Geometric Intuition



# Linear Support Vector Machine - hard margin

- Maximizing the margin is equivalent to minimizing the norm:

$$\arg \max_{\boldsymbol{\theta}} \frac{2}{\|\boldsymbol{\theta}\|} = \arg \min_{\boldsymbol{\theta}} \frac{\|\boldsymbol{\theta}\|}{2}$$

- Formulated as optimization problem:

$$\begin{aligned} & \underset{\boldsymbol{\theta}}{\text{minimize}} && \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} \\ & \text{subject to} && y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1, \quad i = 1, \dots, n. \end{aligned}$$

# Detour - Function optimization

- Support Vector Machine objective function has a convex shape.
  - Quadratic objective with linear constraints.
  - Quadratic solver applicable.
- Quadratic Solver implemented in python package [cvxopt](#)
  - MOSEK.
- Canonical QP formulation:

```
minimize      (1/2)*theta'*P*theta + q'*theta
subject to    G*theta <= h
              A*theta = b.
```

# Support Vector Machine - hard-margin QP

- Primal SVM objective can be formulated as a canonical QP problem.
- Will only converge, if data is linearly separable.
- Exercise.

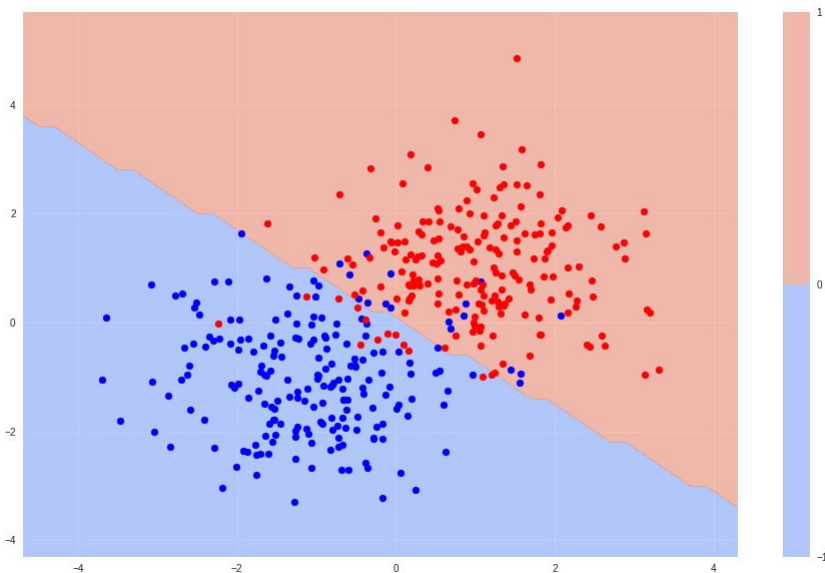
```
minimize      (1/2)*theta'*P*theta + q'*theta
subject to    G*theta <= h
              A*theta = b.
```



# Non linearly separable data

- Should be able to learn a decision boundary
  - even if data not linearly separable.
- Possible solution, separating red from blue, could be:

- How to achieve this?



# Support Vector Machine - soft-margin

$$L(\boldsymbol{\theta}) = \sum_{i=1}^n l(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + c\Omega(\boldsymbol{\theta})$$

- Introduces slack variables
  - Slack variables relax the hard margin constraints.
  - Every non-zero slack variable will be penalized by a regularization parameter.

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + C \sum_{i=1}^n \xi_i$$

$$\begin{aligned} \text{subject to} \quad & y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1 - \xi_i, \quad i = 1, \dots, n \\ & \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned}$$

# Lunch break

# Recap

- Regularized Empirical Risk Minimization as a general framework for function estimation.
- Perceptron algorithm:
  - Finds **any** separating hyperplane.
- Linear **hard-margin** Support Vector Machine:
  - Finds unique max-margin separating hyperplane.
- Linear **soft-margin** Support Vector Machine:
  - Same as hard-margin but with slack variables.
  - Allows for non-linearly separable data.
  - Adds hyperparameter to steer relative importance in objective.

# Duality of constrained optimization

- SVM optimization problem can be solved in primal or dual form.
  - Previous SVM optimization problems were posed in primal form.
- To work with kernels, we need the dual formulation.

# Duality of constrained optimization

Constrained optimization problem

$$\begin{array}{ll} \underset{\boldsymbol{\theta}}{\text{minimize}} & \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} \\ \text{subject to} & y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1, \quad i = 1, \dots, n. \end{array}$$

# Duality of constrained optimization

Constrained optimization problem

$$\begin{array}{ll}\underset{\boldsymbol{\theta}}{\text{minimize}} & \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} \\ \text{subject to} & y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1, \quad i = 1, \dots, n.\end{array}$$

Corresponding Lagrange function

$$L(\boldsymbol{\theta}, \theta_0, \boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} + \sum_{i=1}^n \alpha_i (1 - y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0))$$

# Duality of constrained optimization

Constrained optimization problem

$$\begin{array}{ll}\underset{\boldsymbol{\theta}}{\text{minimize}} & \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} \\ \text{subject to} & y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1, \quad i = 1, \dots, n.\end{array}$$

Corresponding Lagrange function

$$L(\boldsymbol{\theta}, \theta_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + \sum_{i=1}^n \alpha_i (1 - y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0))$$

Take partial derivatives w.r.t. primal parameters, set to 0 and substitute.

$$\begin{array}{ll}\underset{\boldsymbol{\alpha}}{\text{minimize}} & \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^n \alpha_i \\ \text{subject to} & \sum_{i=1}^n \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \quad i = 1, \dots, n\end{array}$$



# Duality of constrained optimization

Constrained optimization problem

$$\begin{aligned} & \underset{\boldsymbol{\theta}}{\text{minimize}} && \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} \\ & \text{subject to} && y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1, \quad i = 1, \dots, n. \end{aligned}$$

Corresponding Lagrange function

$$L(\boldsymbol{\theta}, \theta_0, \boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} + \sum_{i=1}^n \alpha_i (1 - y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0))$$

Take partial derivatives w.r.t. primal parameters, set to 0 and substitute.

$$\begin{aligned} & \underset{\boldsymbol{\alpha}}{\text{minimize}} && \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^n \alpha_i \\ & \text{subject to} && \sum_{i=1}^n \alpha_i y_i = 0 \\ & && \alpha_i \geq 0, \quad i = 1, \dots, n \end{aligned}$$

Primal parameter & decision function

$$\begin{aligned} \boldsymbol{\theta} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \\ f(\mathbf{x}) &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + \theta_0 \end{aligned}$$

# Kernelized Learning

- In dual optimization problem, training data only enter as inner products.
  - Explicit feature mapping possible, but might be expensive to compute.
- Kernel functions implicitly define, possibly infinite dimensional, feature mappings.

- Some kernel functions:  $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + c)^d$$

$$k(\mathbf{x}, \mathbf{y}) = \exp(\gamma \|\mathbf{x} - \mathbf{y}\|)$$

- Specialized kernel functions for Graphs, Trees and Strings also exist.

# Kernelized Learning

- Dual optimization problem can be reformulated, for suited function  $k$ , as:

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^n \alpha_i$$

$$\text{subject to} \quad \sum_{i=1}^n \alpha_i y_i = 0$$

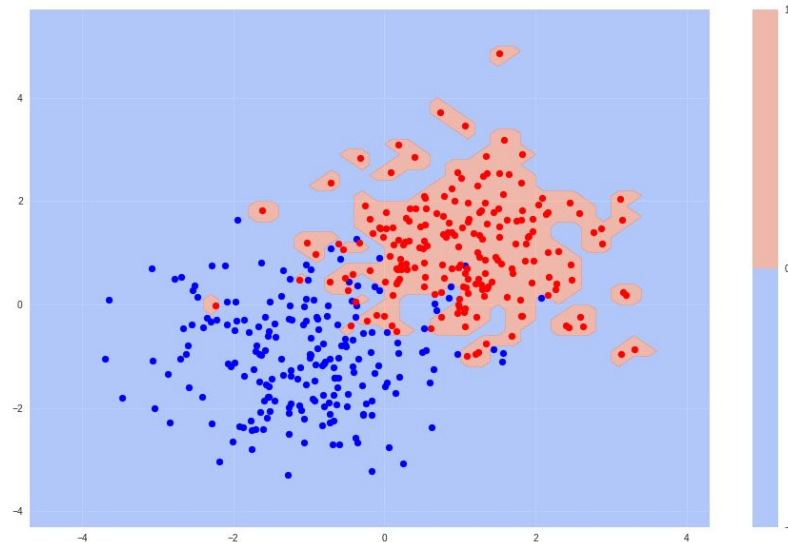
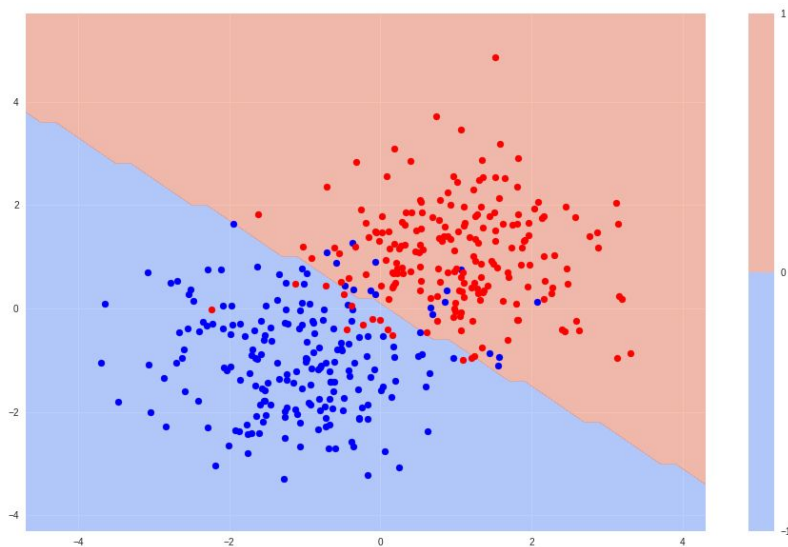
$$\alpha_i \geq 0, \quad i = 1, \dots, n$$

- Kernelized decision function
  - Evaluates function  $k$  for all training samples with the given example:

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + \theta_0$$

# Sparse Kernel Machines

- Highly non-linear decision boundaries possible.
- Dual SVM formulation supports sparse solutions:
  - only need training samples (*support-vectors*) for which the **alphas are non zero**.



# Data Project

- Choose between
  - 20 Newsgroups, around 20.000 samples
    - Learn to predict newsgroup association from email
  - Amazon review data, between 10.000 and 9.000.000 reviews ([ucsd.edu](https://www.ucsd.edu))
    - Possible tasks include
      - Learn to predict rating from text
      - Learn to predict product category from text
      - User/Product based models?

# Data Project - 20 Newsgroups

- Data

- [http://scikit-learn.org/stable/datasets/twenty\\_newsgroups.html](http://scikit-learn.org/stable/datasets/twenty_newsgroups.html)
- Posts from 20 different newsgroups.
  - *rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.misc ...*
- Fairly balanced: Each group has roughly same amount of posts.
- About 11.000 posts in total for training and testing respectively.

- Example post from *sci.electronics*:

*From: ritley@uimrl7.mrl.uiuc.edu ()  
Subject: SEEKING THERMOCOUPLE AMPLIFIER CIRCUIT  
Reply-To: ritley@uiucmrl.bitnet ()  
Organization: Materials Research Lab  
Lines: 17*

*I would like to be able to amplify a voltage signal which is output from a thermocouple, preferably by a factor of 100 or 1000 ---- so that the resulting voltage can be fed more easily into a personal-computer-based ADC data acquisition card. ...*

# Data Project - 20 Newsgroups

- Task
  - Infer most likely newsgroup, having seen the text of a post.
- Possible models
  - [`sklearn.svm.LinearSVC`](#), [`sklearn.svm.SVC`](#) (Calibrated probability estimates possible with Platt-Scaling).
  - SGD variant [`SGDClassifier\(loss='hinge'\)`](#) for larger datasets and/or incremental learning.
- Sample pipeline implemented [here](#).

# Production deployments

- Common Machine Learning pipeline requirements in industry projects:
  - Often need **fast inference**.
  - Often need **low memory footprint**.
  - Often need low technical overhead for deployments.
- Support Vector Machine is a good candidate for the requirements:
  - Inference as simple as a sparse dot product (fast to compute).
  - Primal representation consists only of one parameter vector and a bias term.
- Additional processing time and memory might be needed for potential pre- and post-processing steps.



# Programming Exercise: Production deployments

- Make model consumable with an easy interface.
- Wrap model in web service of choice, could be:
  - [Flask-Restful](#)
  - [Django](#)
  - ...
- Offer HTTP endpoint that takes a newsgroup post.
  - The response should contain the respective newsgroup association
- Sample implementation [here](#).

# Data Project wrap-up

- What dataset did you choose?
- Open Questions?
- Difficulty?
- Follow up ideas?

# Class takeaways

- Solid understanding of:
  - Perceptron Algorithm.
  - Support Vector Machine
    - Primal and Dual formulations.
- Practical experiences:
  - Develop and evaluate sample machine learning pipeline.
  - Expose pipeline via HTTP interface.
- Awareness of frequently encountered industry requirements for machine learning pipelines.

Thank you - Questions?

# References

- Project Repository
  - <https://github.com/tdhd/data-science-retreat-svm>
- [Stephen Boyd - Convex Optimization](#)
- Christopher Bishop
  - Pattern Recognition and Machine Learning
- [scikit-learn](#)
- [cvxopt](#)
- [Alex Smola - Introduction to Machine Learning](#)
- [MOSEK](#) - Optimization Software



# Duality of constrained optimization

Constrained optimization problem

$$\begin{aligned} & \underset{\boldsymbol{\theta}}{\text{minimize}} && \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} \\ & \text{subject to} && y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1, \quad i = 1, \dots, n. \end{aligned}$$

Corresponding Lagrange function

$$L(\boldsymbol{\theta}, \theta_0, \boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} + \sum_{i=1}^n \alpha_i (1 - y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0))$$

Take partial derivatives w.r.t. primal parameters, set to 0 and substitute.

$$\begin{aligned} & \underset{\boldsymbol{\alpha}}{\text{minimize}} && \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^n \alpha_i \\ & \text{subject to} && \sum_{i=1}^n \alpha_i y_i = 0 \\ & && \alpha_i \geq 0, \quad i = 1, \dots, n \end{aligned}$$

Primal parameter & decision function

$$\begin{aligned} \boldsymbol{\theta} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \\ f(\mathbf{x}) &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + \theta_0 \end{aligned}$$

# Kernelized Learning - Background

- Valid kernel functions can be expressed as inner products in space  $V$ .

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- Explicit representation of feature map not required as long as  $V$  is an inner product space.

$$\phi : \mathcal{X} \rightarrow \mathcal{V}$$



# Kernelized Learning

- Not all functions are kernel functions:
  - Compute Gram matrix, on all pairs of training data.

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathbf{K} \in \mathbb{R}^{n \times n}$$

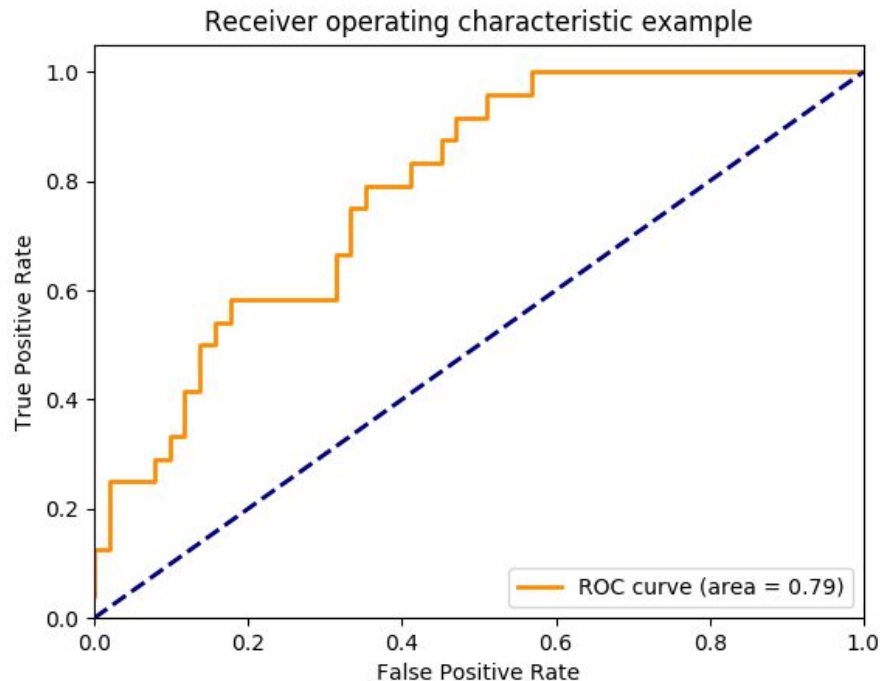
- PSD  $\rightarrow$  all eigenvalues positive.
- Gram matrix needs to be PSD.
- Mercer Kernel, if and only if  $\mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0$ , for all  $\mathbf{x}$  in  $\mathbb{R}^d$  (Gram matrix PSD)

$$\mathbf{K}_{i,j} = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$\mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0$$

# Typical machine learning metrics

- ROC
  - Area under curve
- <http://www.navan.name/roc/>
- Assumes equal cost for Type I and Type II errors
  - Cost sensitive optimization



# Typical machine learning metrics

- ROC
  - Area under curve
- Precision-Recall Curve
  - Area under curve

