Support Vector Machines

Theory & Applications

Philipp Schmidt & Gerrit Gruben
September 2017

Outline for today

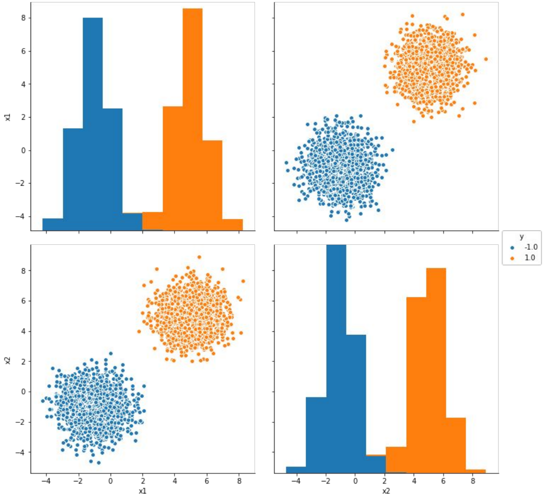
- Organization & Setup.
- How to separate data?
- Discriminative functions.
- Perceptron Algorithm + Exercise.
- Support Vector Machine + Exercise.
- **Lunch Break.**
- Kernel Support Vector Machine.
- Data Project.
- Wrap-up.

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
- Install packages found in *requirements.txt*.

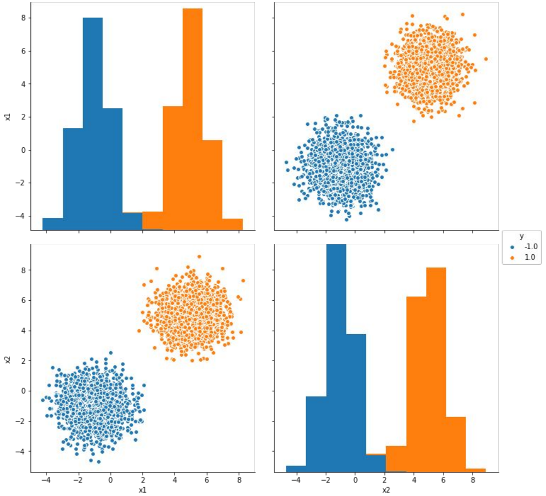
Need help setting things up? Please ask me.

How to separate data?



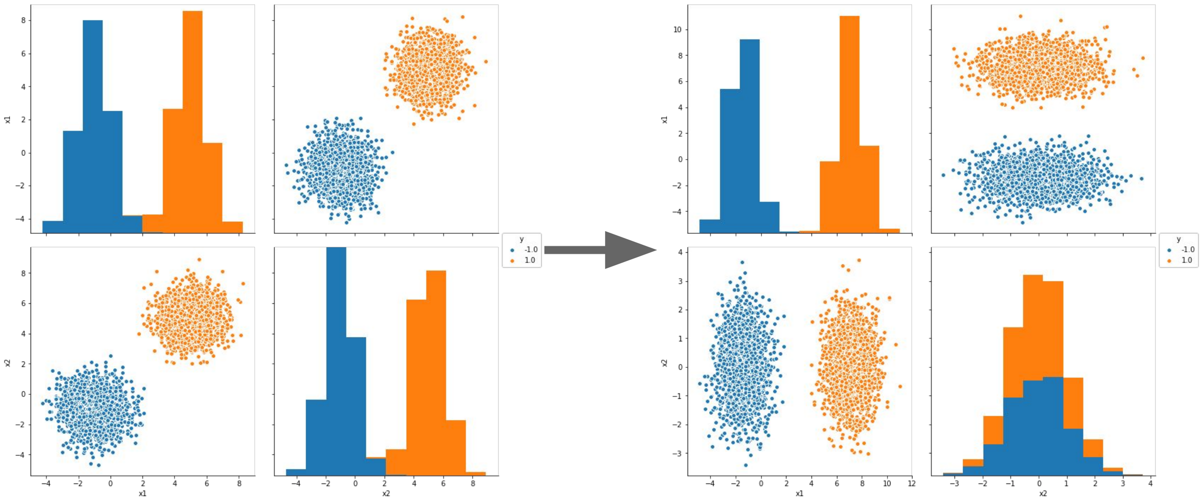
How to separate data?

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



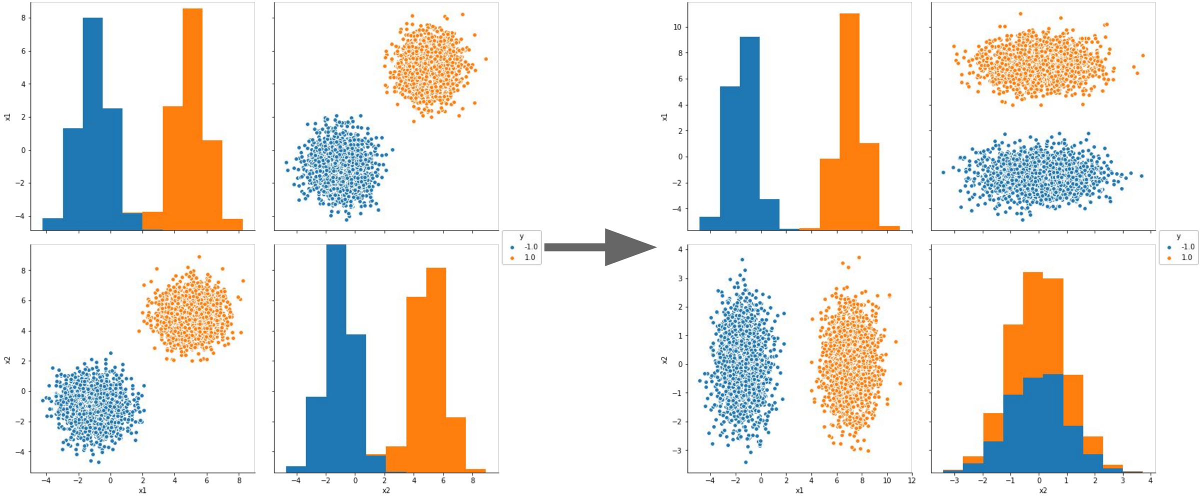
How to separate data?

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

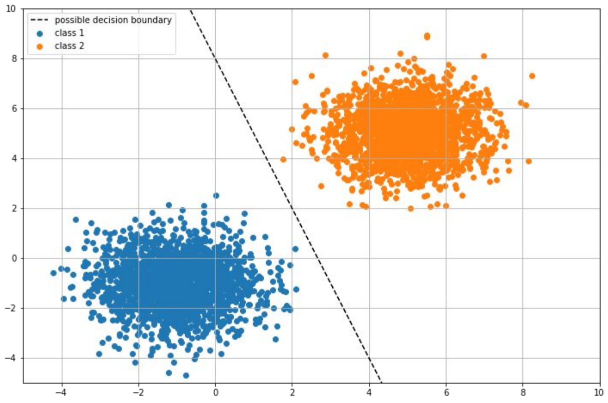


How to separate data?

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



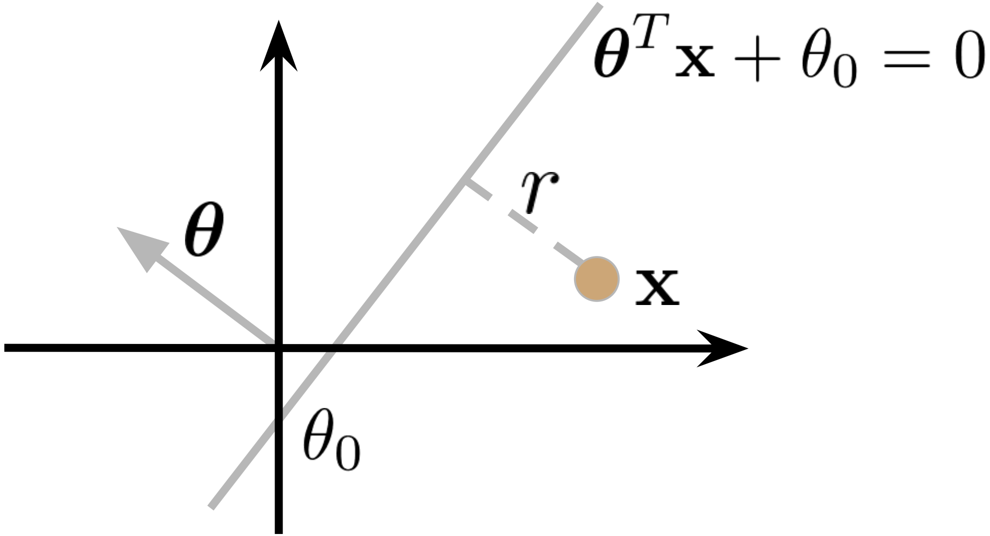
Possible decision boundary



Geometry recap

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

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



Learning discriminative functions

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

- Learn a mapping from input space to output space.
- Domain $\mathcal{X}$ is typically $\mathbb{R}^d$.
- Target 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

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

- 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

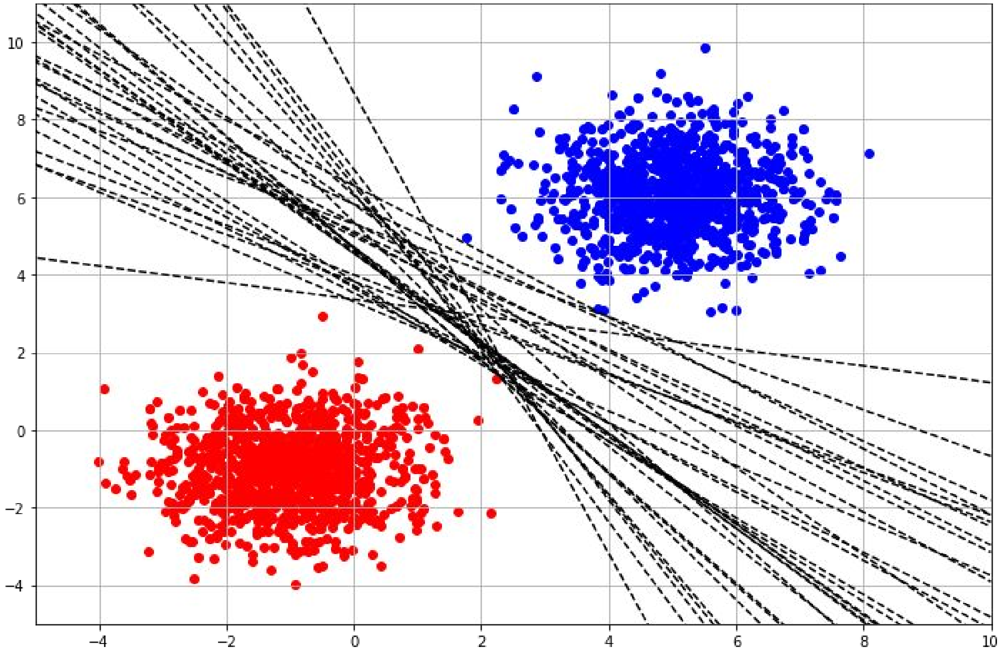
$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \boxed{l\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right), y_i\right)} + \cancel{c\Omega(\boldsymbol{\theta})}$$

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

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

Perceptron algorithm

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



Perceptron algorithm - Exercise

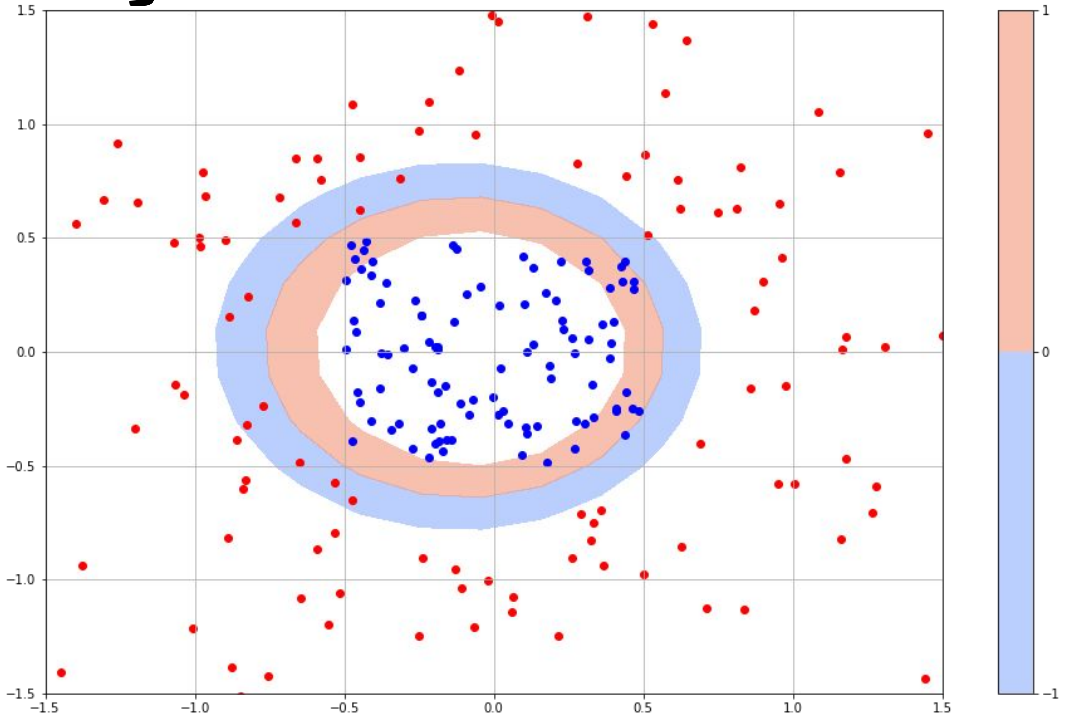
```
argument:
X := {x1, . . . , xm} ⊂ X (data)
Y := {y1, . . . , ym} ⊂ {±1} (labels)
function (theta) = Perceptron(X,Y)
    initialize theta = 0 # includes intercept
    repeat
        Pick (xi, yi) from data
        if yi(theta · xi) ≤ 0 then
            theta = theta + yi · xi
        end
    until yi(theta · 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.



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

Non-linear Perceptron algorithm - Exercise

```
argument:
X := {x1, . . . , xm} ⊂ X (data)
Y := {y1, . . . , ym} ⊂ {±1} (labels)
function (theta) = Perceptron(X,Y,Φ)
    initialize theta = 0 # includes intercept
    repeat
        Pick (xi, yi) from data
        if yi(theta · Φ(xi)) ≤ 0 then
            theta = theta + yi · Φ(xi)
        end
    until yi(theta · Φ(xi)) > 0 for all i
end
```

Non-linear Perceptron Algorithm

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

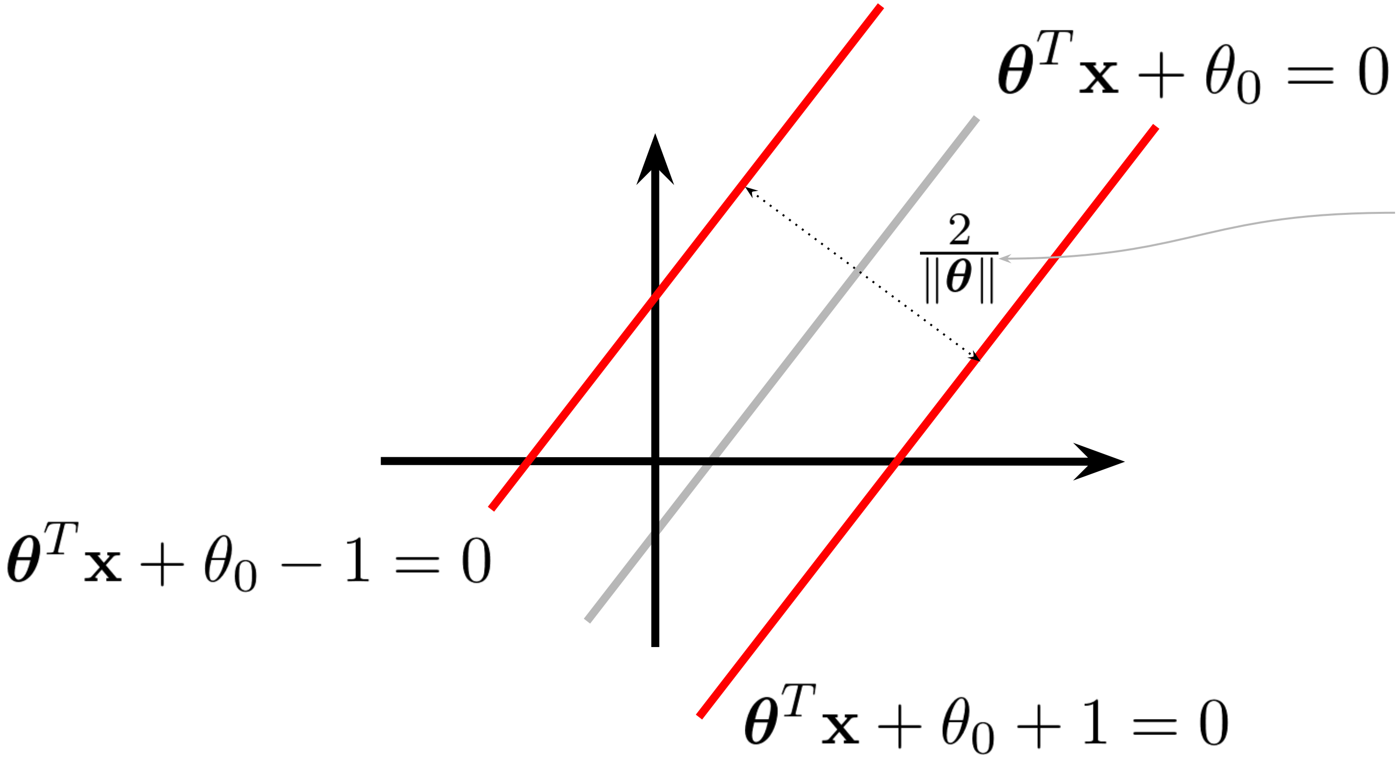
Linear Support Vector Machine

- 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:

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

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)*x'*P*x + q'*x
subject to   G*x <= h
             A*x = b.
```

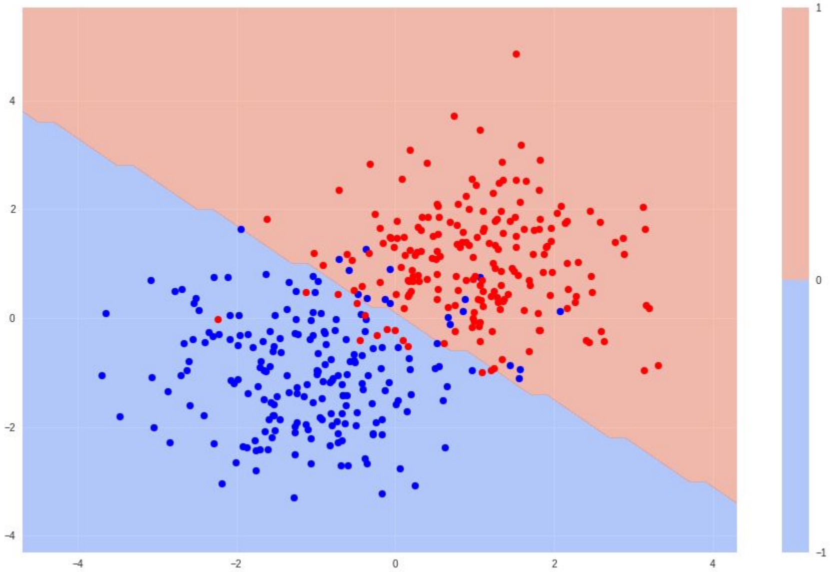
Support Vector Machine - hard-margin QP

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

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

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

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

$$\begin{aligned}
\underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i\left(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0\right) \geq 1 - \xi_i,\ i = 1, \ldots, n \\
& \xi_i \geq 0,\ i = 1, \ldots, n
\end{aligned}$$

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 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

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

Duality of constrained optimization

Constrained optimization problem

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

Corresponding Lagrange function

$$L\left(\boldsymbol{\theta},\, \theta_0,\, \boldsymbol{\alpha}\right) = \tfrac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + \textstyle\sum_{i=0}^{n} \alpha_i\left(1 - y_i\left(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0\right)\right)$$

Duality of constrained optimization

Constrained optimization problem

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

Corresponding Lagrange function

$$L\left(\boldsymbol{\theta},\ \theta_0,\ \boldsymbol{\alpha}\right) = \tfrac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + \textstyle\sum_{i=0}^{n} \alpha_i \left(1 - y_i\left(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0\right)\right)$$

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

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

Duality of constrained optimization

Constrained optimization problem

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

Corresponding Lagrange function

$$L\left(\boldsymbol{\theta},\ \theta_0,\ \boldsymbol{\alpha}\right) = \tfrac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + \textstyle\sum_{i=0}^{n} \alpha_i \left(1 - y_i\left(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0\right)\right)$$

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

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

Primal parameter & decision function

$$\boldsymbol{\theta} = \textstyle\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$f\left(\mathbf{x}\right) = \textstyle\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + \theta_0$$

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:

$$k\left(\mathbf{x},\mathbf{y}\right)=\mathbf{x}^T\mathbf{y}$$

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

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

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

Kernelized Learning

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

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

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

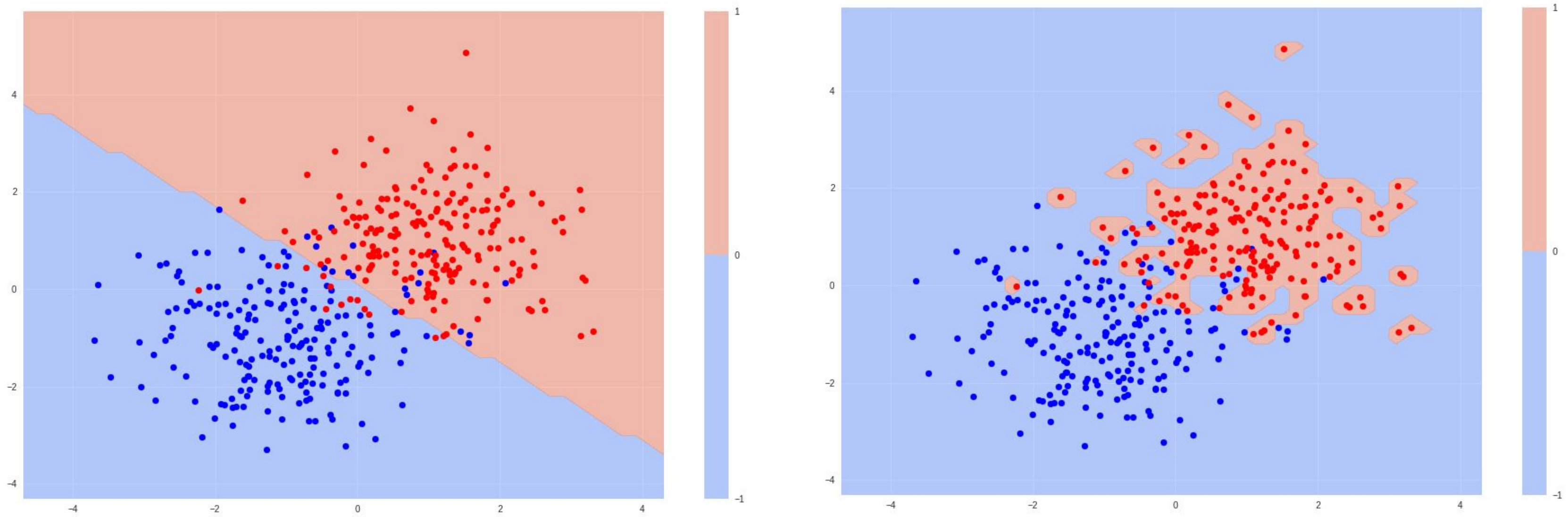
$$\alpha_i \geq 0,\ i = 1, \ldots, n$$

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

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

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**.



SVM Data Project - 20 Newsgroups

- Data
  - 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. ...*

SVM Data Project - 20 Newsgroups

- Task
  - Infer most likely newsgroup, having seen the text of a post.
- Possible models
  - sklearn.svm.LinearSVC
    - Solves optimization problem with LibLinear.
  - sklearn.svm.SVC
    - Solves optimization problem with LibSVM (non-linear kernels possible).
    - Calibrated probability estimates possible with Platt-Scaling.
- Sample pipeline implemented here.

Data Project wrap-up

- Open Questions?
- Difficulty:
  - Too hard?
  - Too easy?
- Follow up ideas?

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 postprocessing 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.

Take aways

- 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

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

Corresponding Lagrange function

$$L\left(\boldsymbol{\theta},\ \theta_0,\ \boldsymbol{\alpha}\right) = \tfrac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + \textstyle\sum_{i=0}^{n} \alpha_i\left(1 - y_i\left(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0\right)\right)$$

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

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

Primal parameter & decision function

$$\boldsymbol{\theta} = \textstyle\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$f\left(\mathbf{x}\right) = \textstyle\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + \theta_0$$

Kernelized Learning - Background

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

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right)$$

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

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

Kernelized Learning

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

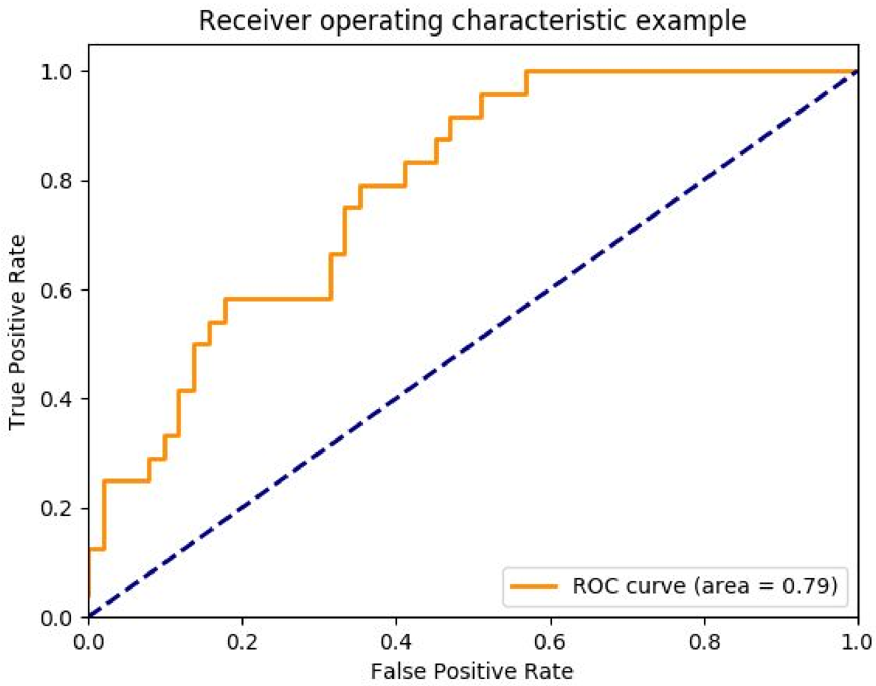
$$\mathbf{K}_{i,\,j} = k\left(\mathbf{x}_i,\,\mathbf{x}_j\right)$$
$$\mathbf{K} \in \mathbb{R}^{n \times n}$$

- PSD -> all eigenvalues positive.
- Gram matrix needs to be PSD.
- Mercer Kernel, if and only if xTKx >= 0, forall x in R^d (Gram matrix PSD)

$$\mathbf{K}_{i,\,j} = \Phi\left(\mathbf{x}_i\right)^T \Phi\left(\mathbf{x}_j\right)$$
$$\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$$

Typical machine learning metrics

- ROC
  - Area under curve



Typical machine learning metrics

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