Support Vector Machines (SVM)

Objectives

1. Gain an intuition about the *purpose* and *power* of SVMs.
2. Explore (some) of the mathematics behind SVMs.
3. Supercharge SVMs with kernels and soft margins.
4. Gain an intuition about the Bias-Variance tradeoff while using SVMs.

Maximum Margin Classifier: (morning lecture)
1963: Vapnik, Chervonenkis

Soft Margins and the "Kernel Trick": (afternoon lecture)
1992-1995: Vapnik, Boser, Guyon, Cortes
This is the modern Support Vector Machine (SVM).

Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

Supervised Learning

High level: What is supervised learning?

Learn an unknown function from a set of **labeled** training data.

- Our training data is limited and finite. A useful algorithm must generalize well to "unseen" data.
- Example: Children learning colors.
- Support Vector Machines (SVMs) are a *supervised* learning algorithm.

Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

Notation

Goal:

Learn a model of a function $F : X \to Y$ from a training set D.

$D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, where

- $x^{(j)} \in X$ is often called the "input".
- $y^{(j)} \in Y$ is often called the "label" or "target".

Notation (cont.)

$F : X \rightarrow Y$

- Often, $X = \mathbb{R}^n$
- Often, Y is a finite set (i.e. a classification task)

We want our learned model to *generalize* well.

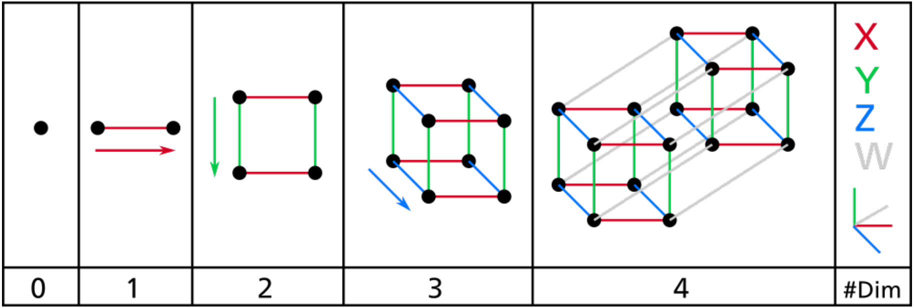
Generalization error is a measure of the model's performance on all possible "unseen" data.

Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

Dimensions

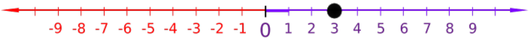
Basic stuff, I know...



[1] By NerdBoy1392 (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

1D

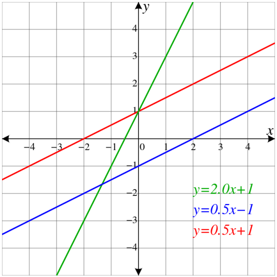
How do you split this space?



[1] By HakunamentaMathsIsFun at en.wikipedia [CC0], from Wikimedia Commons, Public Domain

2D

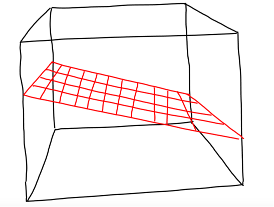
How do you split this space?



[1] By ElectroKid (talk • contribs). Original: HiTe. (Modification from the original work.) [CC BY-SA 1.0 (http://creativecommons.org/licenses/by-sa/1.0)], via Wikimedia Commons

3D

How do you split this space?



4D, 5D, etc...

Hard to visualize... :/

In general, an n-dimensional space can be separated by an $(n-1)$-dimensional *hyperplane*.

In an n-dimensional space any hyperplane can be defined by $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The hyperplane includes all $x \in \mathbb{R}^n$ where:

$$w_0 x_0 + w_1 x_1 + ... + w_{n-1} x_{n-1} - b = 0$$

usually written:

How to interpret w and b

So, w and b define a hyperplane. Is there an interpretation of w and b that can help us visualize this hyperplane?

How to interpret w and b

So, w and b define a hyperplane. Is there an interpretation of w and b that can help us visualize this hyperplane?

- $\frac{w}{||w||}$ is the hyperplane's normal vector.
- $\frac{b}{||w||}$ is the hyperplane's distance from the origin.

Example in 2D

$\frac{w}{||w||}$ is the hyperplane's normal vector

$\frac{b}{||w||}$ is the hyperplane's distance from the origin

Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

Binary Classification

A supervised learning problem

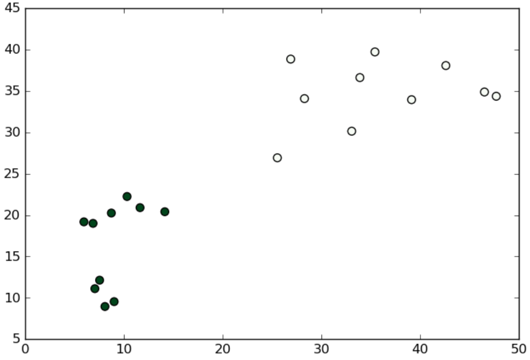
Recall, we're trying to learn $F : X \to Y$.

- Let, $X = \mathbb{R}^n$
- For binary classification, $Y = \{0, 1\}$

Big idea: Let's have our model find a hyperplane that splits our n-dimensional data X into the set where $y = -1$ and the set where $y = 1$.

Binary Classification: Example

How many ways can we use a hyperplane to classify this dataset correctly?

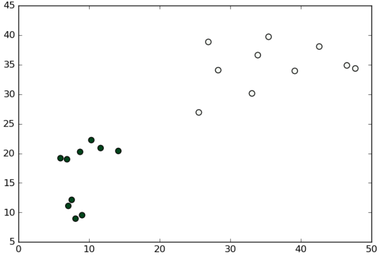


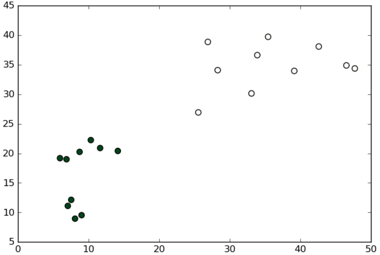
$$X = \mathbb{R}^2$$

$$Y = \{-1, 1\}$$

Binary Classification: Example

Two Example Solutions



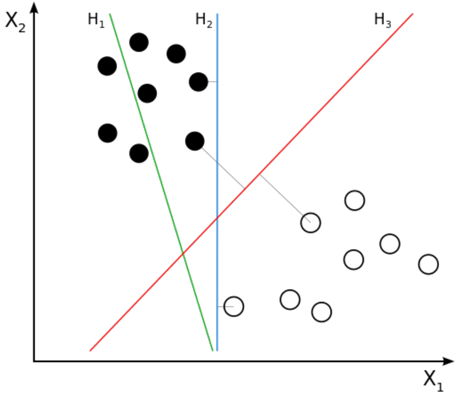


Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

Defining Margin

The distance from the hyperplane to the nearest training-data point.



[1] By User:ZackWeinberg, based on PNG version by User:Cyc [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Why Maximize the Margin?

Our goal is to train a model that generalizes to "unseen" data.

Large margin means better generalization.

Intuitively, this makes sense (see previous slide)

Outline

1 Review
- Supervised Learning
- Notation
- Hyperplanes

2 Motivation
- Binary Classification
- Margin
- Maximum Margin Classifier

3 SVMs
- Soft Margin
- Kernels
- Misc Topics

Maximum Margin Classifier

Goal

Goal: Calculate w and b of the hyperplane:

$$w \cdot x - b = 0$$

... such that the classes are split correctly and the margin is maximized.

First, some house cleaning: What happens to the hyperplane when we scale w and b by some factor c?

We need to define a "canonical" w and b. This will help later.

Let

$$|w \cdot x^{(i)} - b| = 1$$

where $x^{(i)}$ is the closest point to the hyperplane.

There will be a unique scaled w and b to achieve this.

Now, if $x^{(i)}$ is the closest point to the hyperplane, then the distance from $x^{(i)}$ to the hyperplane (at point x_0) is our margin. What is that distance?

Maximum Margin Classifier

Margin (cont.)

Maximum Margin Classifier

First Attempt

Maximize $\frac{1}{||w||}$

subject to:

$$|w \cdot x^{(i)} - b| \geq 1,$$

$$\text{for all } x^{(i)} \in D$$

... but we don't know how to solve this optimization problem. Let's reformulate.

Maximum Margin Classifier

Reformulated

Minimize $||w||^2$

subject to:

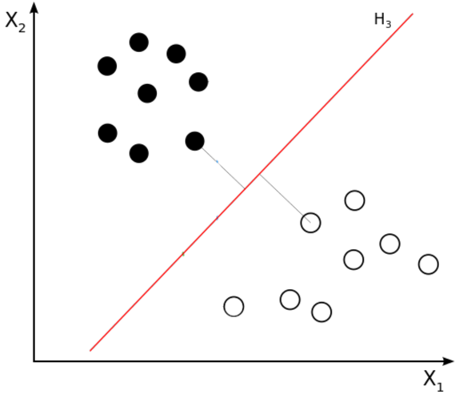
$$y^{(i)}(w \cdot x^{(i)} - b) \geq 1,$$

for all $(y^{(i)}, x^{(i)}) \in D$

... plus more steps... and we eventually get a quadratic programming formulation.

Support Vectors

The maximum margin hyperplane is defined only by the points that touch the margin. These are called the “support vectors”.



LogisticRegression:

Link

SVC:

Link

(end of morning lecture)

Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

Soft Margin Motivation

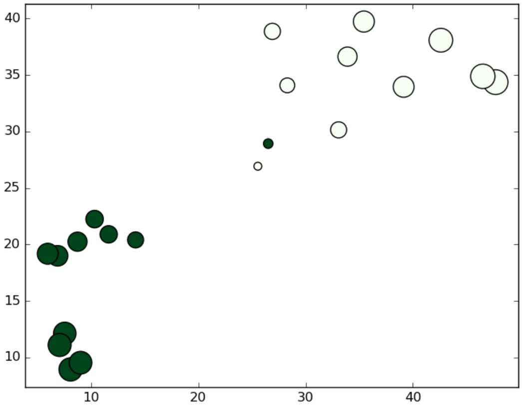
What if:

1. Your data isn't linearly separable?
2. Your data is noisy/has outliers?

Soft Margins address these problems.

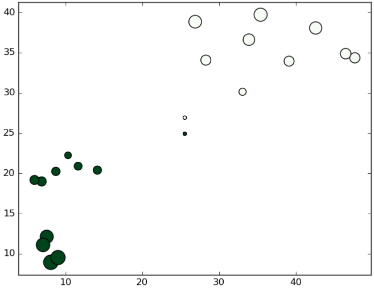
Soft Margins

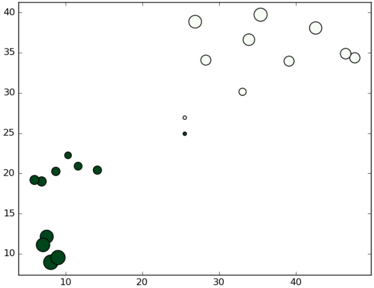
Inseparable Data



Soft Margins

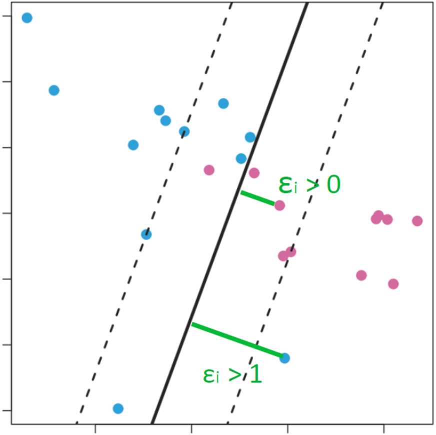
Outliers in Data





Soft Margins

We need some sort of slack/budget



$\varepsilon_i = 0$ for being on correct side of margin
$\varepsilon_i > 0$ for violating the margin
$\varepsilon_i > 1$ for being on wrong side of hyperplane

Soft Margin

The C hyperparameter

An extension to Maximum Margin Classifiers adds a hyperparameter C to control the misclassification error penalty/margin tradeoff.

Minimize $||w||^2$

subject to:

$$y^{(i)}(w \cdot x^{(i)} - b) \geq 1$$

for all $(y^{(i)}, x^{(i)}) \in D$

Support Vector Classifier

The C hyperparameter

Support Vector Classifier (SVC) extends Maximum Margin Classifiers by adding a hyperparameter C to control the misclassification error penalty/margin tradeoff.

Large C: Harder margins: value classification accuracy over a large margin

Small C: Softer margins: value a large margin over classification accuracy

Soft Margins

scikit-learn code

```
from sklearn.svm import SVC
...
svc = SVC(C=1.0, kernel='linear')
svc.fit(x, y)
```

SVC supports the C parameter as the soft-margin hyperparameter.

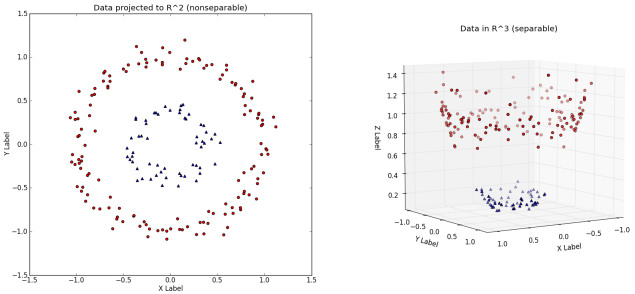
Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

The “Kernel Trick”

The idea...

Idea: If data is inseparable in its input space, maybe it will be separable in a higher-dimensional space.



[1] Unknown source

We have not discussed exactly how the SVC is computed but it turns out that the optimization problem to maximize the margin involves only the *dot products* of the observations (as opposed to the observations themselves):

$$\mathcal{L}(\alpha) = \sum_{i=1}^{m} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha^{(i)} \alpha^{(j)} (x^{(i)} \cdot x^{(j)})$$

The “Kernel Trick”

Creating a kernel...

What if we never took the dot product, but we instead replaced it with a generalization of the dot product of the form (a “kernel function”)?

$$x^{(i)} \cdot x^{(j)} \rightarrow K(x^{(i)}, x^{(j)})$$

The "Kernel Trick"

Why is this so cool?

$$\mathcal{L}(\alpha) = \sum_{i=1}^{m} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha^{(i)} \alpha^{(j)} (K(x^{(i)}, x^{(j)}))$$

- Saves some computation.
- Opens new possibilities. A kernel can operate in infinite dimensions!

The Linear Kernel

$$K(x^{(i)}, x^{(j)}) = x^{(i)} \cdot x^{(j)}$$

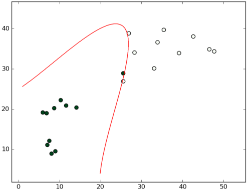
The Polynomial Kernel

$$K(x^{(i)}, x^{(j)}) = (1 + x^{(i)} \cdot x^{(j)})^d$$

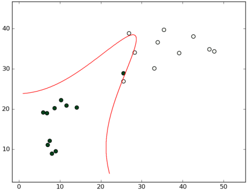
- Equivalent to the dot product in the d-order ϕ space
- Requires an extra hyper-parameter, d, for "degree"

The Polynomial Kernel

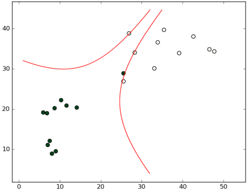
Example



svc = SVC(C=10000.0, kernel='poly', degree=3)



svc = SVC(C=10000.0, kernel='poly', degree=5)



svc = SVC(C=10000.0, kernel='poly', degree=10)

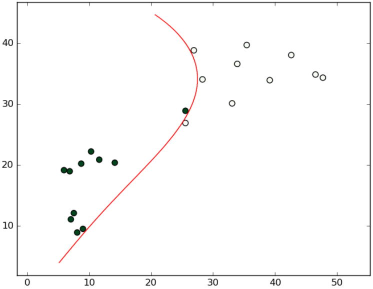
The RBF Kernel ("Gaussian")

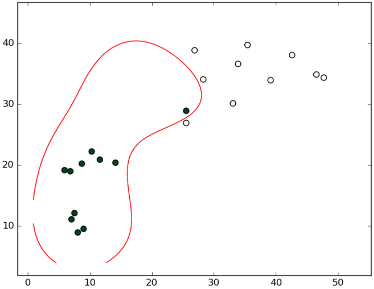
(Radial Basis Function)

$$K(x^{(i)}, x^{(j)}) = \exp\left(-\gamma ||x^{(i)} - x^{(j)}||^2\right)$$

- Equivalent to the dot product in the Hilbert space of infinite dimensions
- Requires an extra hyper-parameter, γ, "gamma"

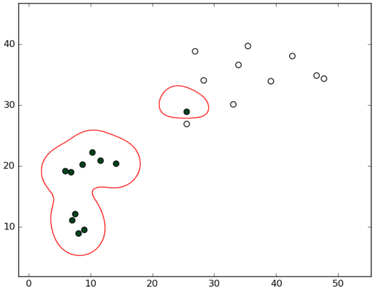
The RBF Kernel
Examples



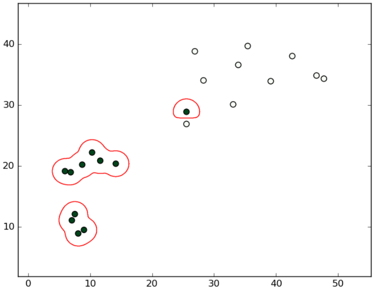


svc = SVC(C=10000.0, kernel='rbf', gamma=0.001)

svc = SVC(C=10000.0, kernel='rbf', gamma=0.01)



svc = SVC(C=10000.0, kernel='rbf', gamma=0.1)



svc = SVC(C=10000.0, kernel='rbf', gamma=1.0)

High-Bias Models

Makes many assumptions and prefers to solve problems a certain way.

E.g. A linear SVM looks for dividing hyperplanes in the input space *only*.

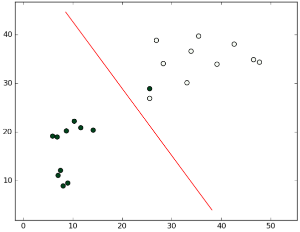
For complex data, high-bias models often *underfit* the data.

High-Variance Models

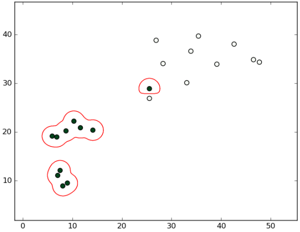
Makes fewer assumptions and has more representational power.

E.g. An RBF SVM looks for dividing hyperplanes in an infinite-dimensional space.

For simple data, high-variance models often *overfit* the data.



svc = SVC(C=0.01, kernel="linear")



svc = SVC(C=10000.0, kernel='rbf', gamma=1.0)

Outline

1. Review
   - Supervised Learning
   - Notation
   - Hyperplanes
2. Motivation
   - Binary Classification
   - Margin
   - Maximum Margin Classifier
3. SVMs
   - Soft Margin
   - Kernels
   - Misc Topics

SVMs vs Logistic Regression
(some rules of thumb)

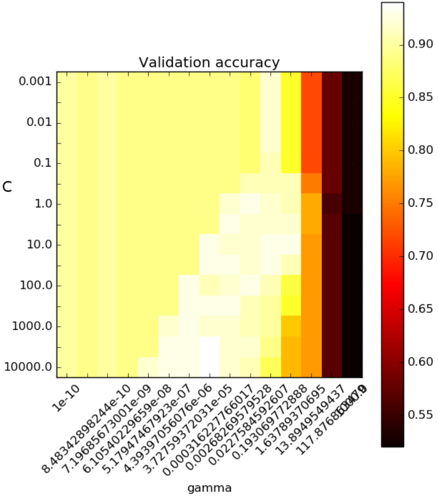
1. Logistic Regression maximizes the *Binomial Log Likelihood* function. SVMs maximize the *margin*.
2. When classes are nearly separable, SVMs tends to do better than Logistic Regression. Otherwise, Logistic Regression (with Ridge) and SVMs are similar.
3. If you want to estimate probabilities however, Logistic Regression is the better choice.
4. With kernels, SVMs work well. Logistic Regression works fine with kernels but can get computationally too expensive.

Grid Search

Hyperparameter Tuning

Let's find C and γ by searching through values we expect might work well.

Use cross-validation accuracy to determine which values are best.



Grid Search

code

```
svc_rbf = SVC(kernel='rbf')

param_space = {'C':     np.logspace(-3, 4, 15),
               'gamma': np.logspace(-10, 3, 15)}

grid_search = GridSearchCV(svc_rbf, param_space,
                           scoring='accuracy', cv=10)
grid_search.fit(x, y)

print grid_search.grid_scores_
print grid_search.best_params_
print grid_search.best_score_
print grid_search.best_estimator_
```