What's an SVM

Like Logistic Regression Support Vector Machines are a classification algorithm.

Remember Logistic Regression?

- Linear model whose parameters derived from maximizing log-likelihood of odds ratios.
- Outputs were probability estimates.

Why do we need SVM?

- Imagine a trivial case where we have to categories of points that are completely separable.
- Why should we choose the logistic regression line out of all the possible options? Maybe some other line would be best?

Linear Algebra

Let's let L be an *affine set*, a hyperplane that doesn't pass through the origin.

$$L = x : f(x) = \beta_0 + \beta^T x = 0$$

Let's take it to be true (or convince ourselves later) that $f(x)$ is proprtional to the signed distance from x to the hyperplane L.

This yields a natural way to classify!

Optimal Separating Hyperplanes

Still considering the case where our data is perfectly separable.

$$max_{\beta,\beta_0,||\beta||=1} M$$

subject to:

$$y_i(x_i^T\beta + \beta_0) \geq M, \forall i = 1, \ldots, N$$

Maximizing the Margin

The optimal separating hyperplane gives us an alternative separating line to the logistic regression.

Why might we want an alternative?

- Boundary points (support) are most important
- Consequences for prediction

Non-separable case

What if our data is not separable? This means we can't satisfy these constraints for all i.

$$y_i(x_i^T \beta + \beta_0) \geq M$$

Which we'll discuss this afternoon.

Support Vector Machines

- What's an SVM?
- Derivation in completely seperating case.
- Full specification
- Examples and Code

Imperfectly Separable

$$max_{\beta,\beta_0,||\beta||=1} M$$

subject to:

$$y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i)\forall i,$$

and

$$\xi_i \geq 0, \sum \xi_i \leq constant$$

We can transform this into a minimization problem that looks like:

$$min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i$$

subject to:

$$\xi_i \geq 0$$

and

$$y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\forall i$$

Which of the terms above is a cost function and which represents the margin?

Hinge Loss

$$y_i(x_i^T\beta + \beta_0 \geq 1 - \xi_i$$

$$1 - y_i(x_i^T\beta + \beta_0) \leq \xi_i$$

So what does this function look like if things are classified correctly (incorrectly)?

Things to consider when you use an SVM:

- Class imbalance (addressable with class_weight arg in sklearn)
- Scaling (as usual)

Another linear classifier

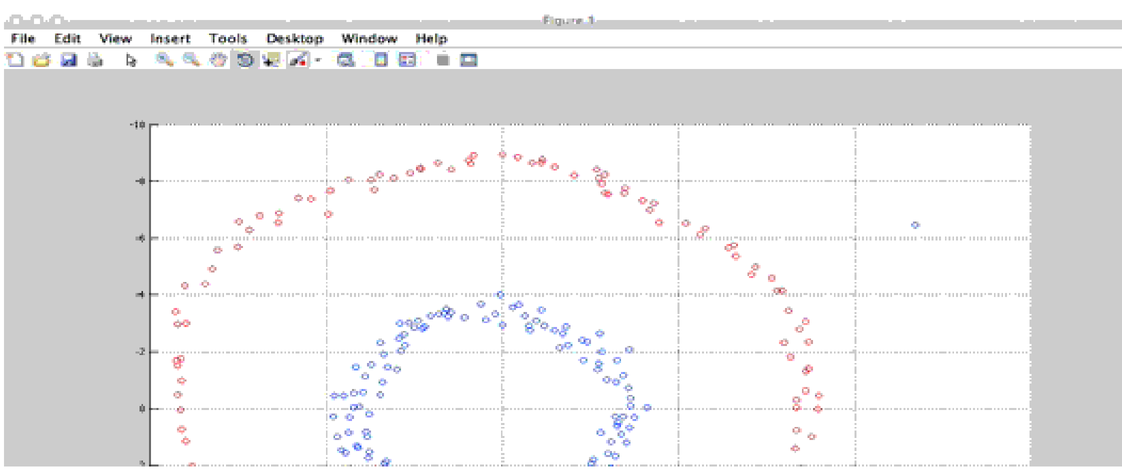
What if our data is non-linear? Of course we could use basis transformations like those we used with logistic regression for example. Something like:

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$$

Where

$$h : N \rightarrow V$$

What about lots of transformations? At some point that would become difficult to compute.



Or would it?

We can use the "kernel trick" to avoid explicitly evaluating h for some cases.

There are certain kernel functions $k : NxN \rightarrow \mathbb{R}$, which can be expressed as an inner product in some other (much) higher-dimensional space.

$$k(x, x') = \langle h(x), h(x') \rangle v$$

Since the inner product gives us a notion of similarity, which is related to the inverse of a distance function, we will have a solution for:

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i k(x, x_i)$$

Computing $k(. , .)$ can often be cheaper than directly computing the basis transforms $h(.)$

Can I still use my favorite basis transforms?

Popular kernels include:

Radial Basis Function or Gaussian

$$K(x, y) = exp(\gamma ||x - y||^2)$$

Polynomial

$$K(x, y) = (a + x^T y)^d$$

and Sigmoid (Hyperbolic Tangent)

$$\tanh(\gamma \langle x, x' \rangle + r$$

Other Kernels are possible, but in practice, RBF is commonly used on a wide range of problems.

Why use SVM or Kernels?

- Sparsity of solutions via l1 penalty.
- Rows and Columns: lots of columns may allow linearity, few rows may require it.
- Interpretability of coefficients, linear.
- Kernels lose interpretability, may gain predictive.

Multi-Class Classification

So far, we've talked about how to classify two-class data, but what if there are more than two classes?

Two approaches:

- One vs. Rest
- One vs. One

One vs. Rest

If we have K classes, train K models. Let f_k be the kth model. To choose a class for our problem we simply:

$$f(x) = argmax_k f_k(x)$$

One vs. One

If we have K classes, train $K * (K-1)/2$ models.

Then let

$$f(x) = argmax_k(\sum_j f_{kj}(x))$$

for example, choose the case with the maximum number of votes.

OvR vs. OvO

OvR

- Requires probabilities
- Trains K models, with N rows each.

OvO

- Votes may have ties
- Trains $K(K-1)/2$ models, but models have only $2\frac{N}{K}$ rows in them.

Real examples

Let's see a real example of logisitic regression vs. SVMs.

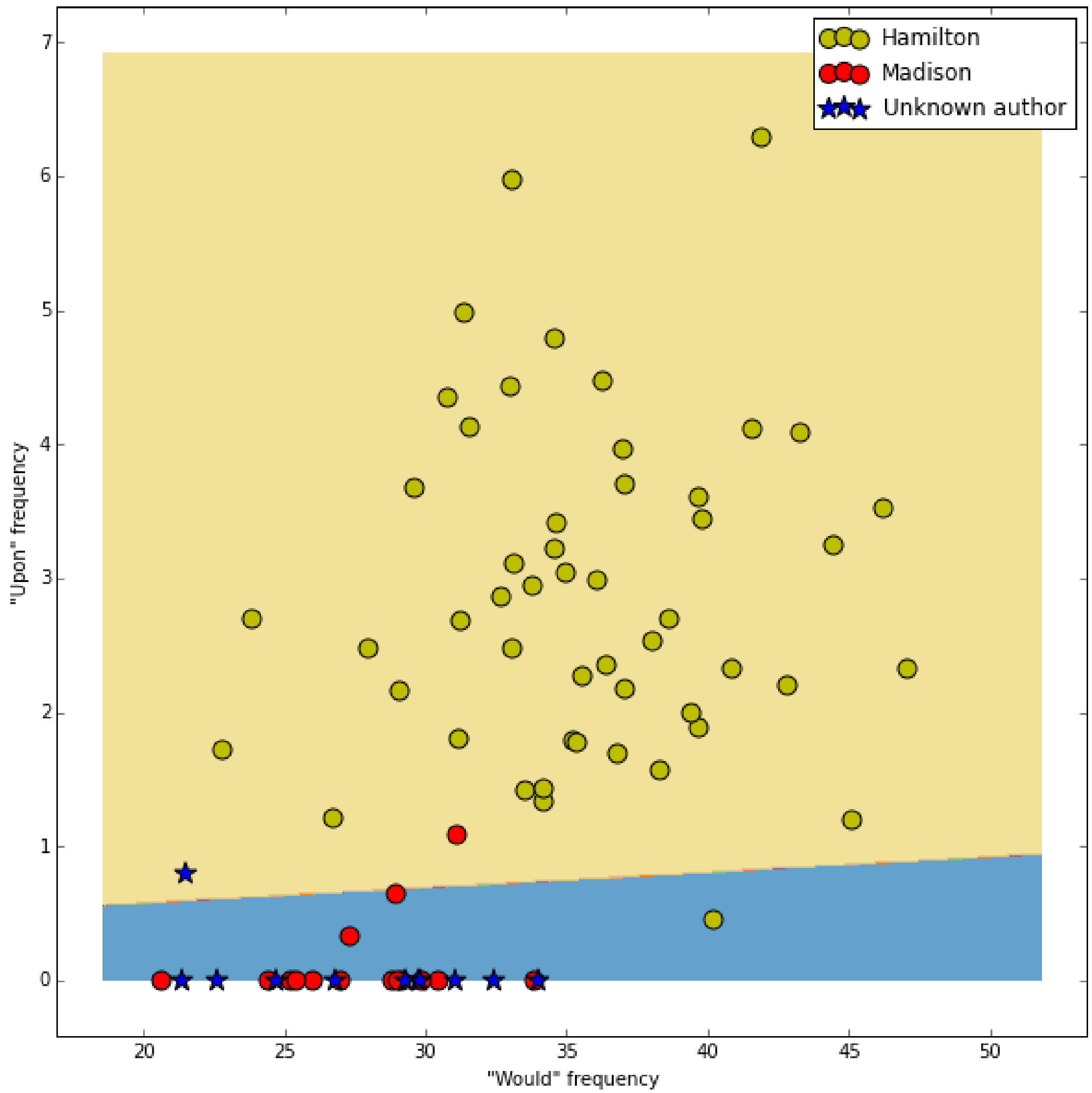
We'll use 2-word frequency variables to predict the authorship of the famous federalist papers.

In [31]:
```
X.head()
```

Out[31]:

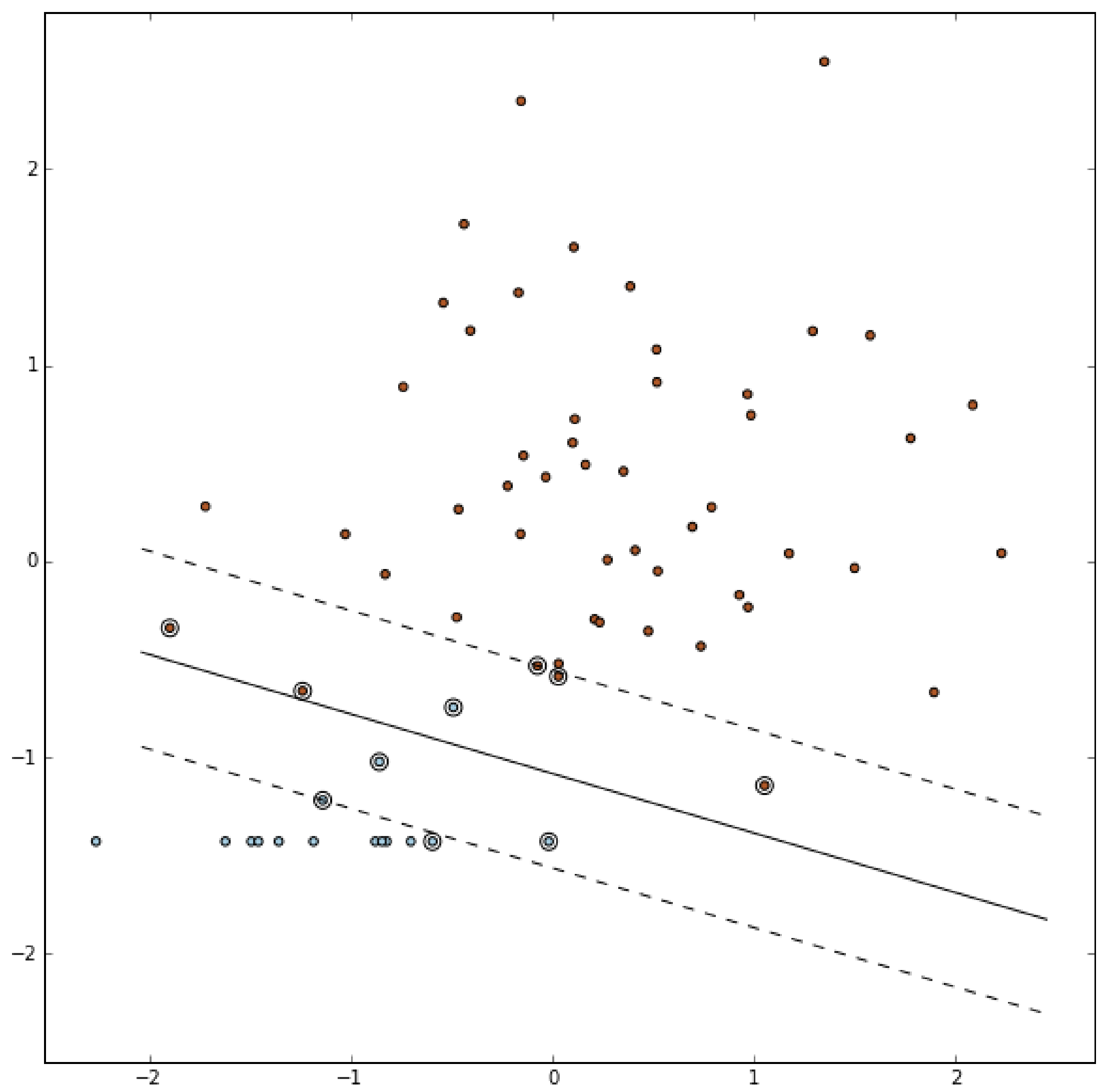
	to	upon
0	34.639409	3.407155
5	22.825151	1.722653
6	30.805687	4.344392
7	34.161491	1.330967
8	31.193490	1.808318

In [44]:
```
#Fit logistic regression with normal l2 error function.
logistic = linear_model.LogisticRegression()
logistic_w = logistic.fit(X,y)
plot_results(logistic_w)
```



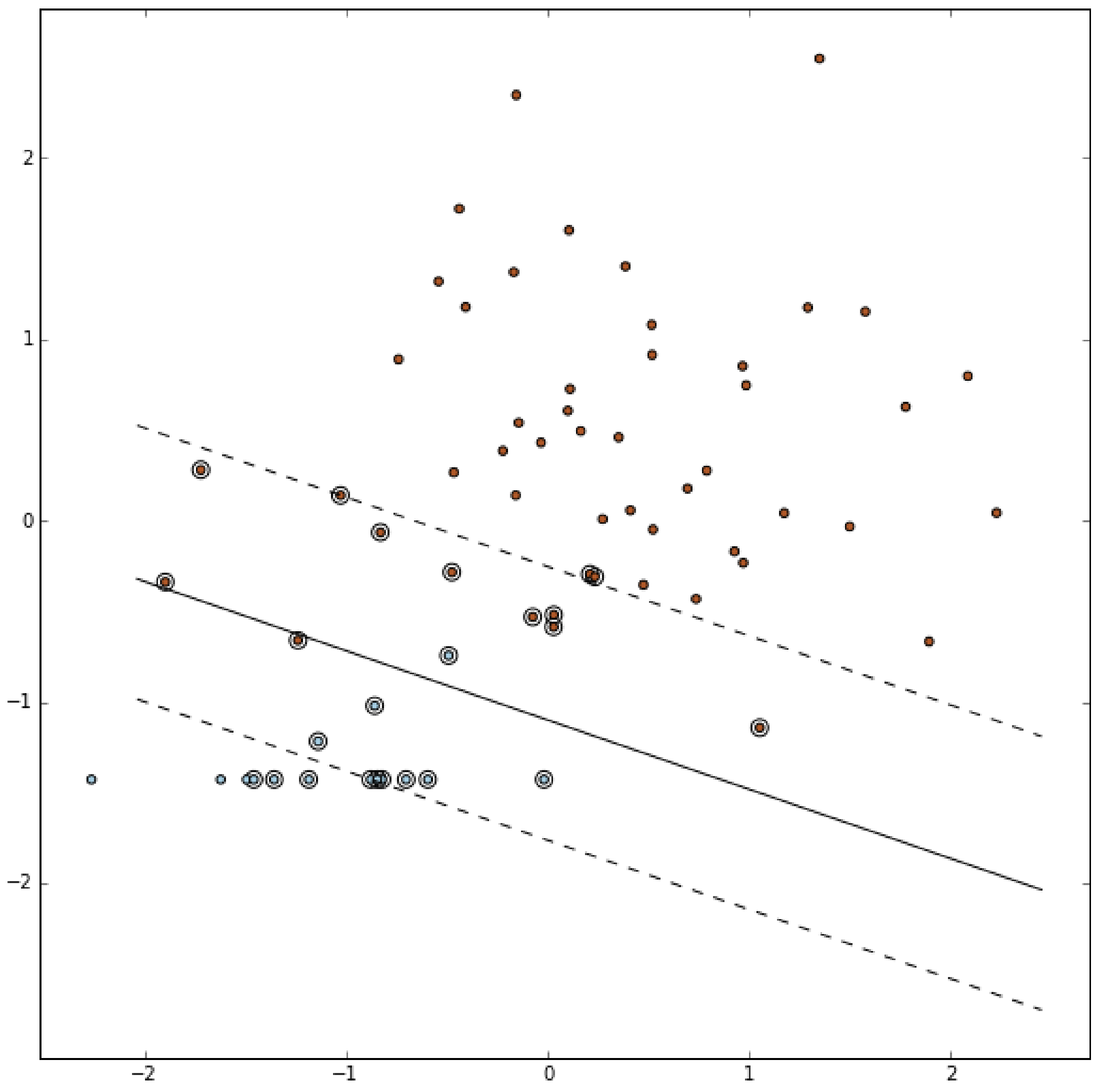
In [58]:

```
s = svm.SVC(kernel="linear")
w_svm = s.fit(scale(X),y)
plot_svm(w_svm, scale(X), y)
#Show effect of C params.
```



What will happen to margin if we reduce C?

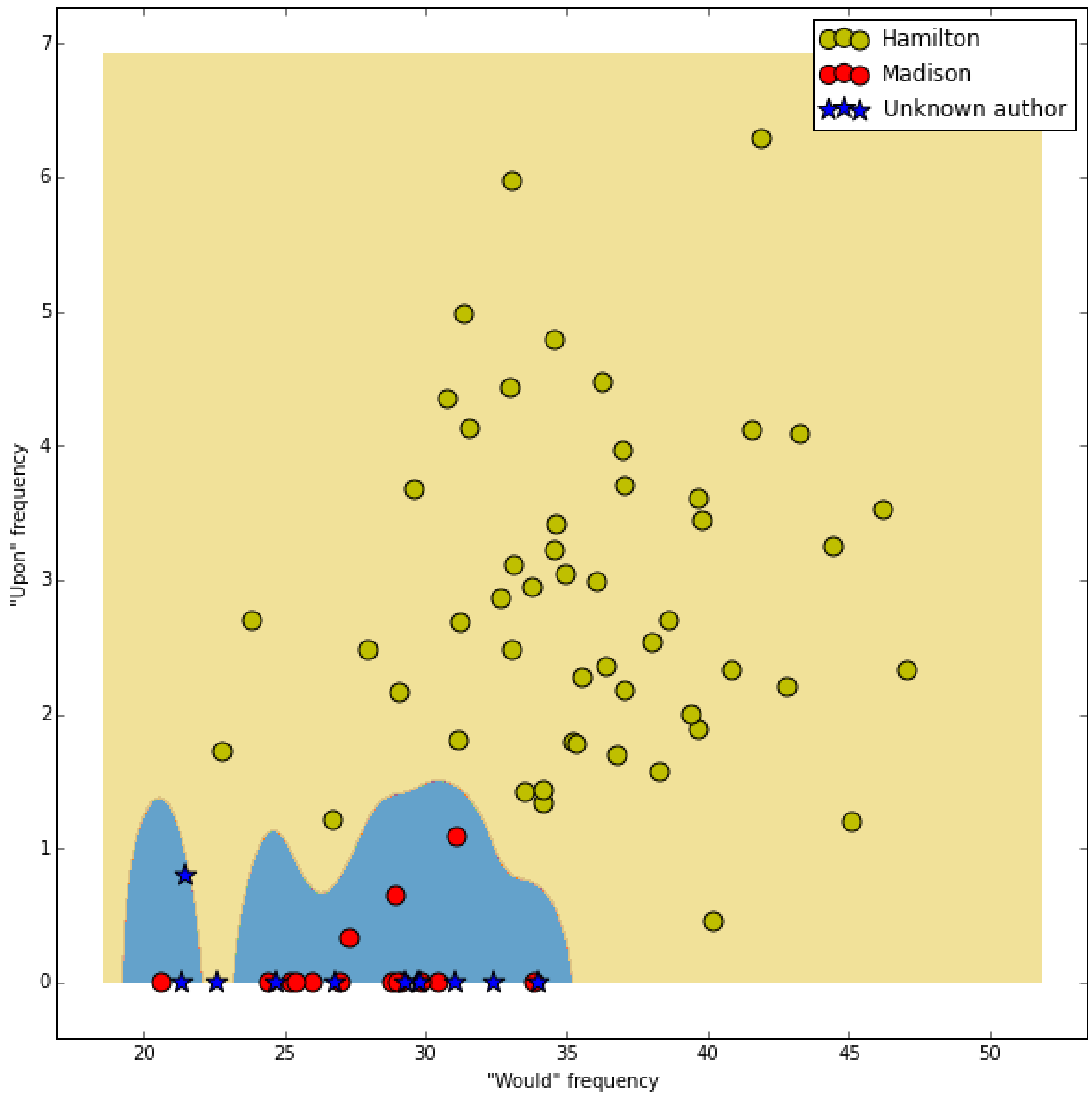
In [60]:
```
s = svm.SVC(kernel="linear", C=.1)
w_svm = s.fit(X,y)
plot_svm(w_svm, scale(X), y)
#Show effect of C params.
```



What Does a Kernel Do To Solutions?

In [25]:

```
s = svm.SVC(kernel="rbf", C=10)
w_svm = s.fit(X,y)
plot_results(w_svm)
```



In [28]:
```
s = svm.SVC(kernel="rbf", C=.5)
w_svm = s.fit(X,y)
plot_results(w_svm)
```