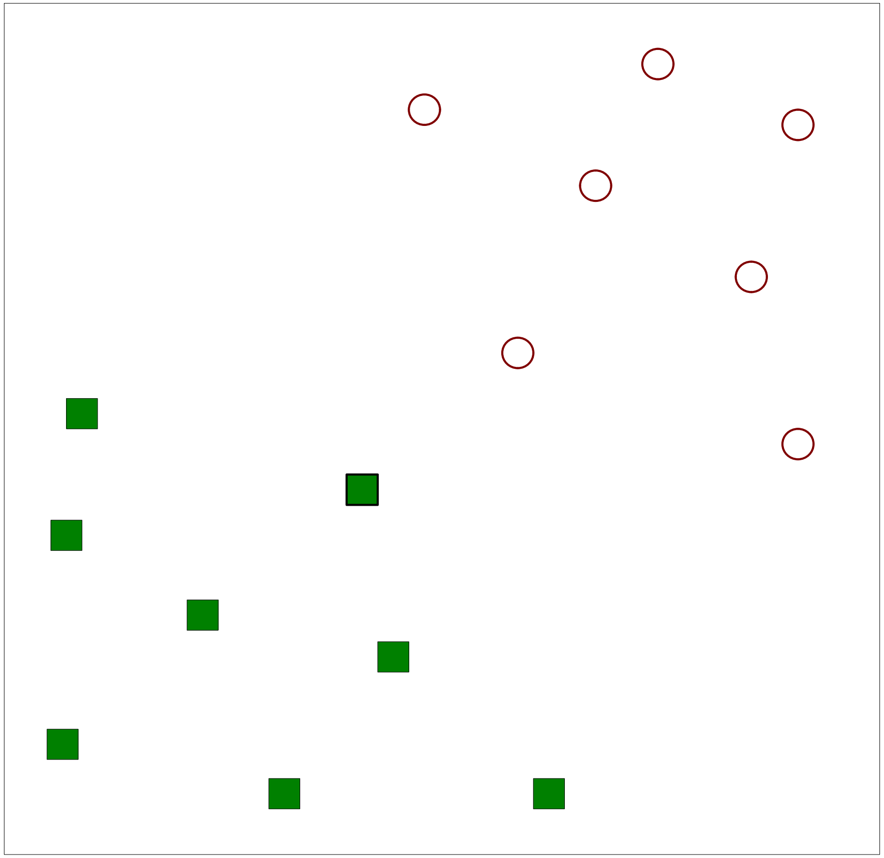
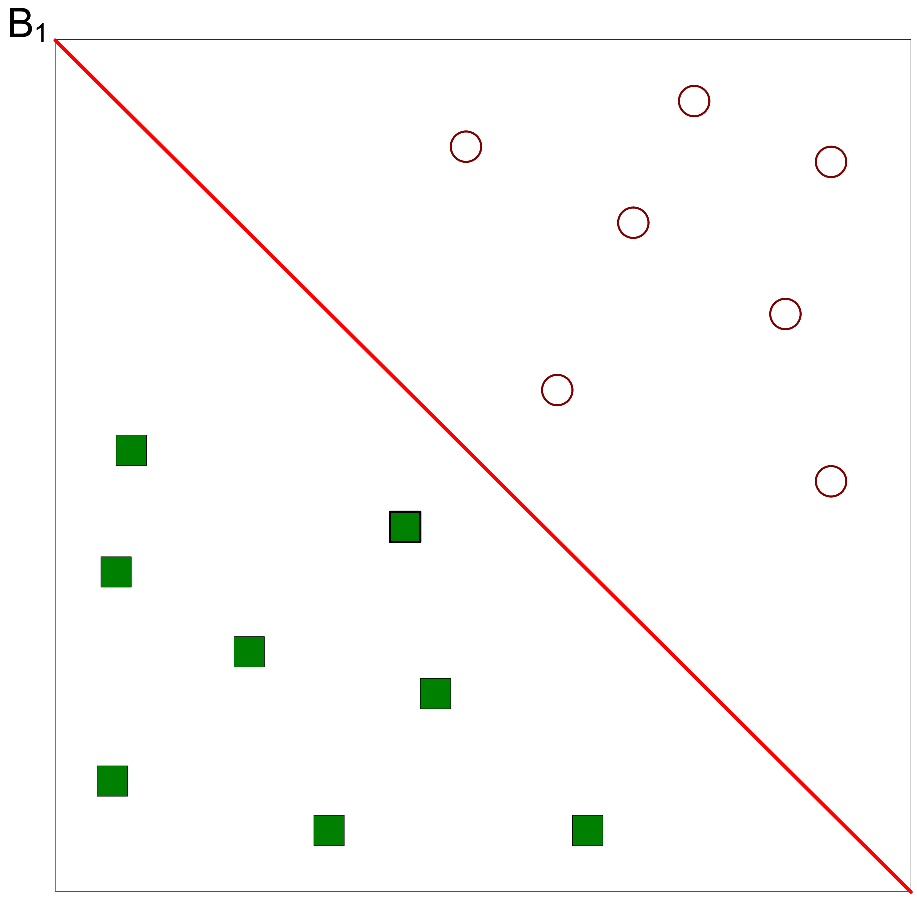

SUPPORT VECTOR MACHINES

SUPPORT VECTOR MACHINES (SVM)



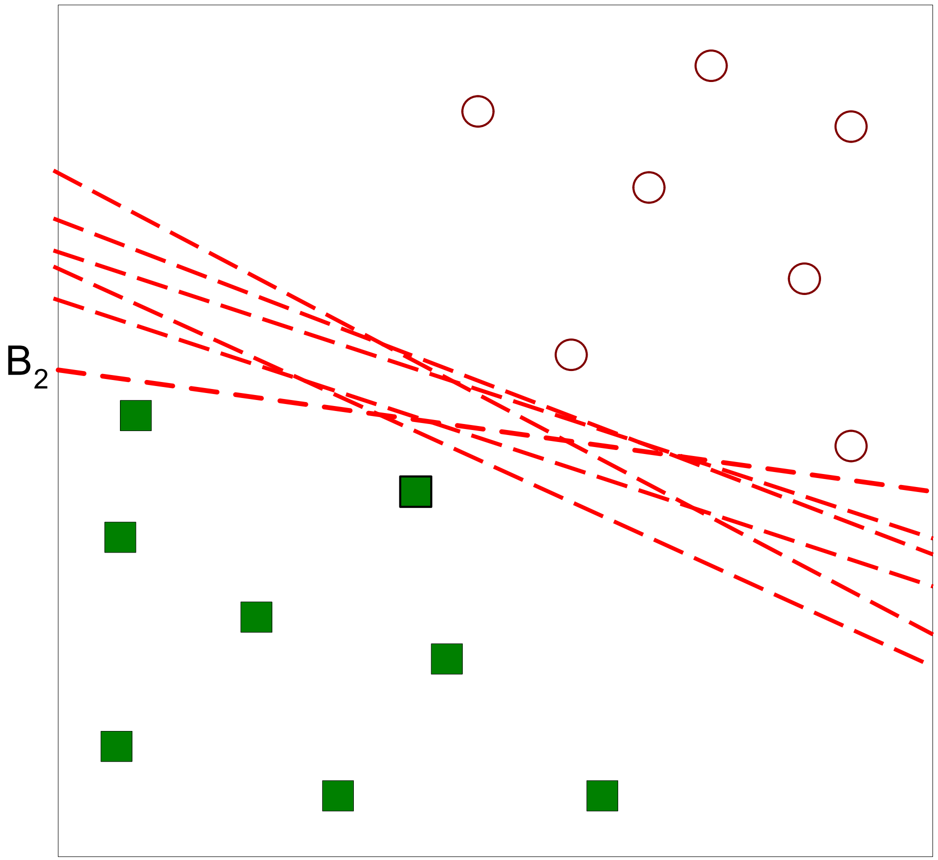
Find a linear hyperplane (decision boundary) that will separate the data

SUPPORT VECTOR MACHINES



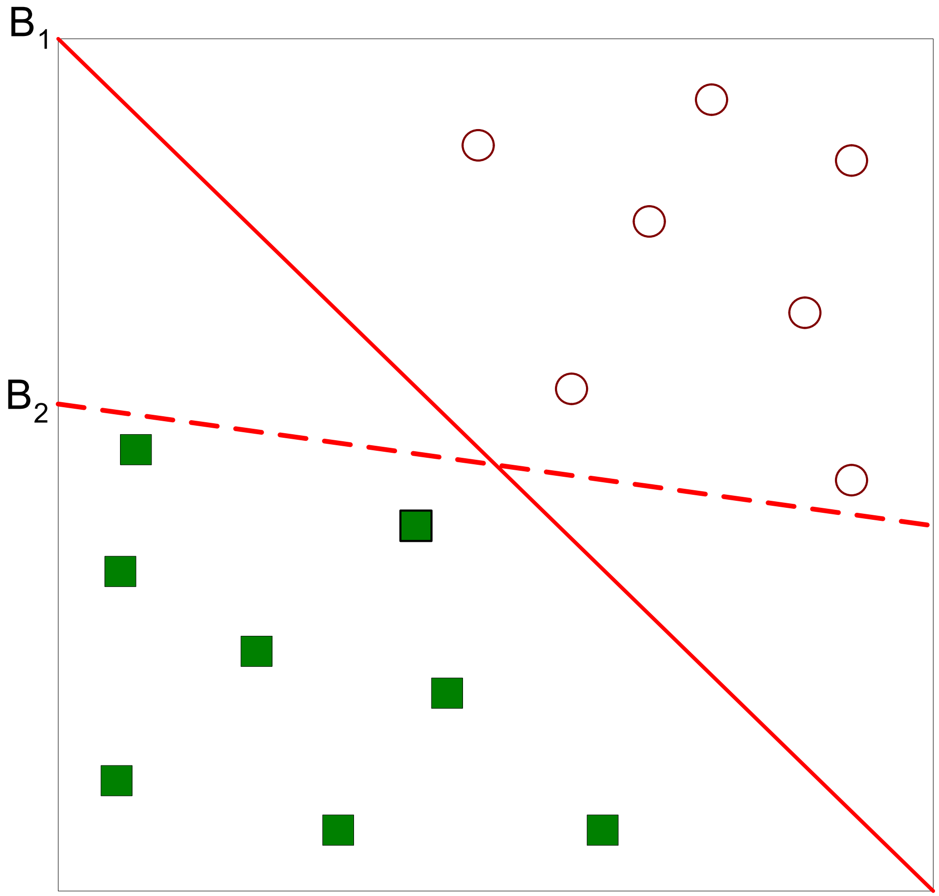
One possible solution

SUPPORT VECTOR MACHINES



Other possible solutions

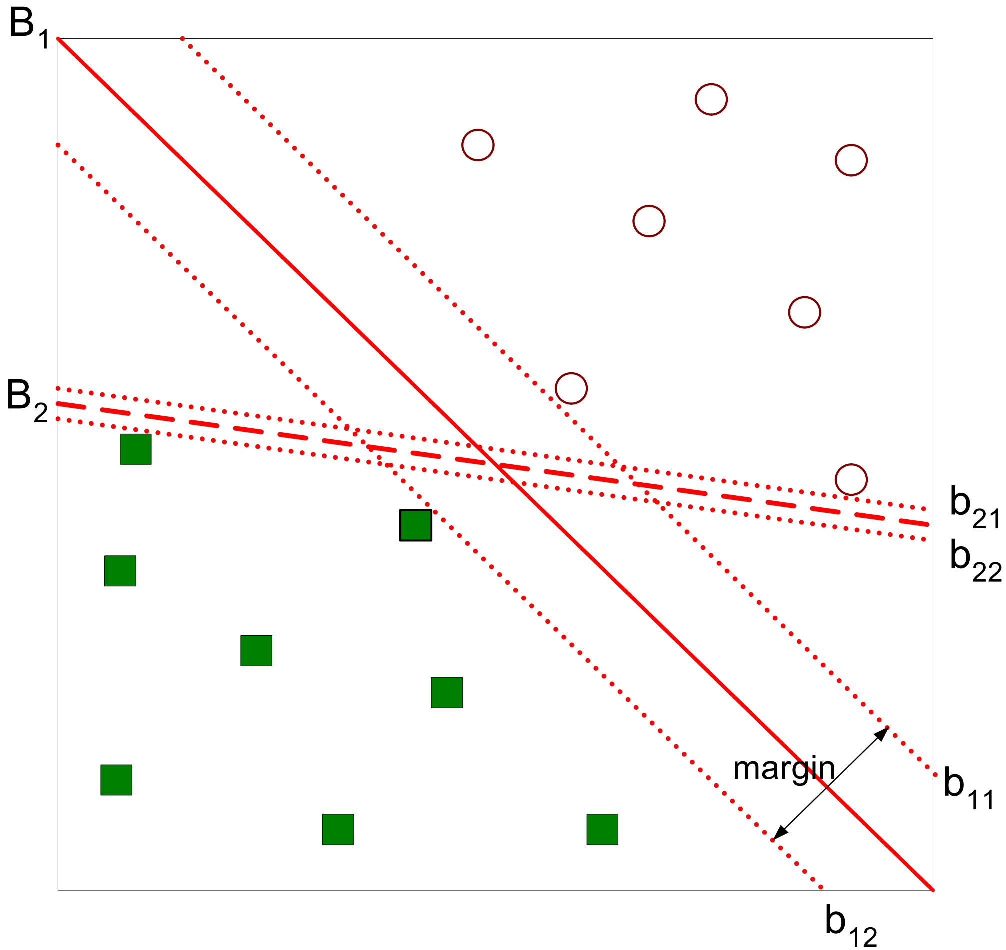
SUPPORT VECTOR MACHINES



Which one is better? B_1 or B_2?   ??????????

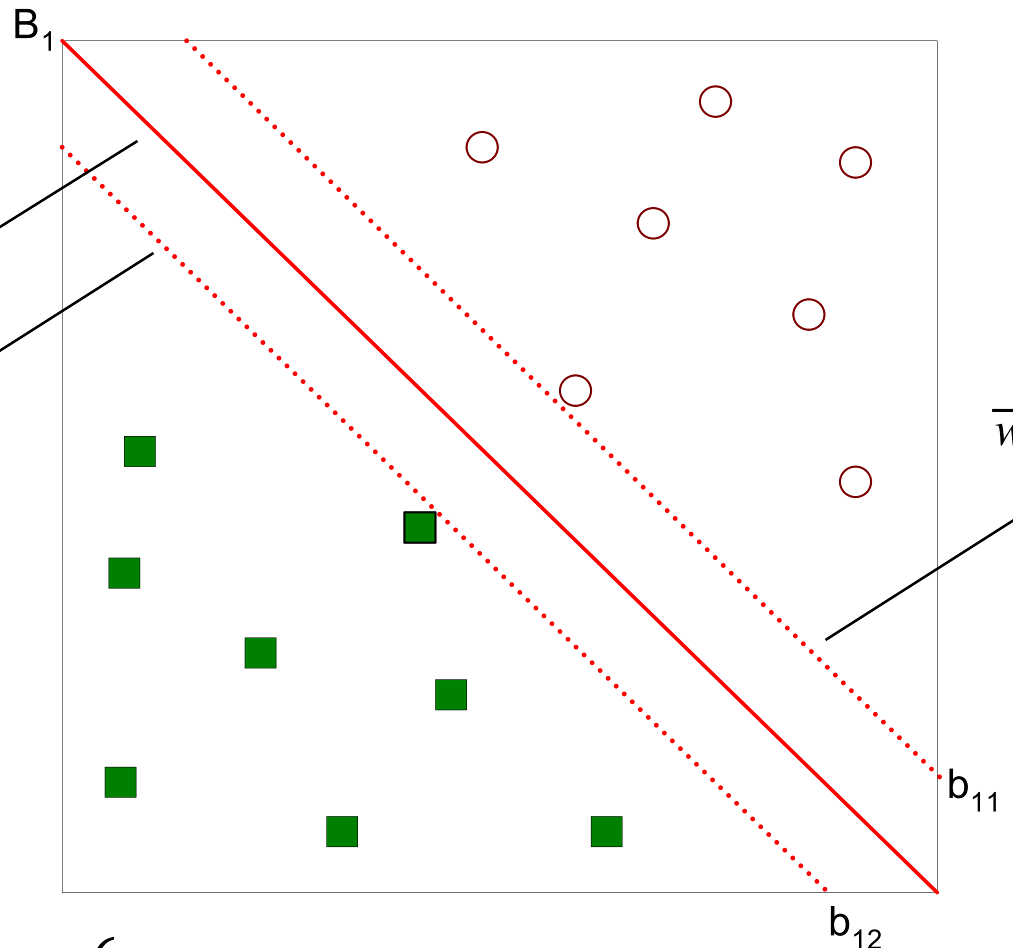
How do you define better?   ??????????

SUPPORT VECTOR MACHINES



Find hyperplane maximizes the margin → B_1 is better than B_2

SUPPORT VECTOR MACHINES



$$\vec{w} \cdot \vec{x} + b = 0$$

$$\vec{w} \cdot \vec{x} + b = -1$$

$$\vec{w} \cdot \vec{x} + b = +1$$

$$f(x) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 1 \\ 0, & \text{if } \vec{w} \cdot \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{||\vec{w}||^2}$$

SUPPORT VECTOR MACHINES

We want to maximize: $$\text{Margin} = \frac{2}{||\vec{w}||^2}$$

Which is equivalent to minimizing: $$L(w) = \frac{||\vec{w}||^2}{2}$$

But subject to the following constraints:

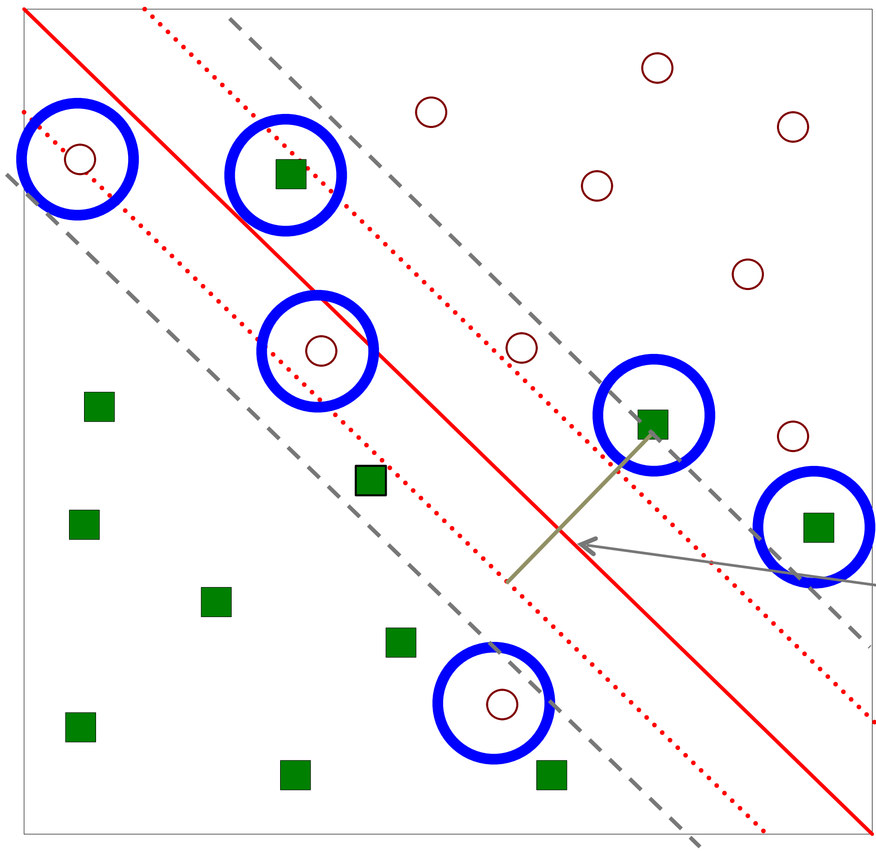
$$\vec{w} \cdot \vec{x} + b \geq 1 \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x} + b \leq -1 \text{ if } y_i = -1$$

This is a constrained optimization problem

- Numerical approaches to solve it (e.g., quadratic programming)

SUPPORT VECTOR MACHINES

What if the problem is not linearly separable?



Apply some sort of penalty

$$\frac{\xi_i}{\|w\|}$$

SUPPORT VECTOR MACHINES

What if the problem is not linearly separable?

- Introduce slack variables
- Need to minimize:

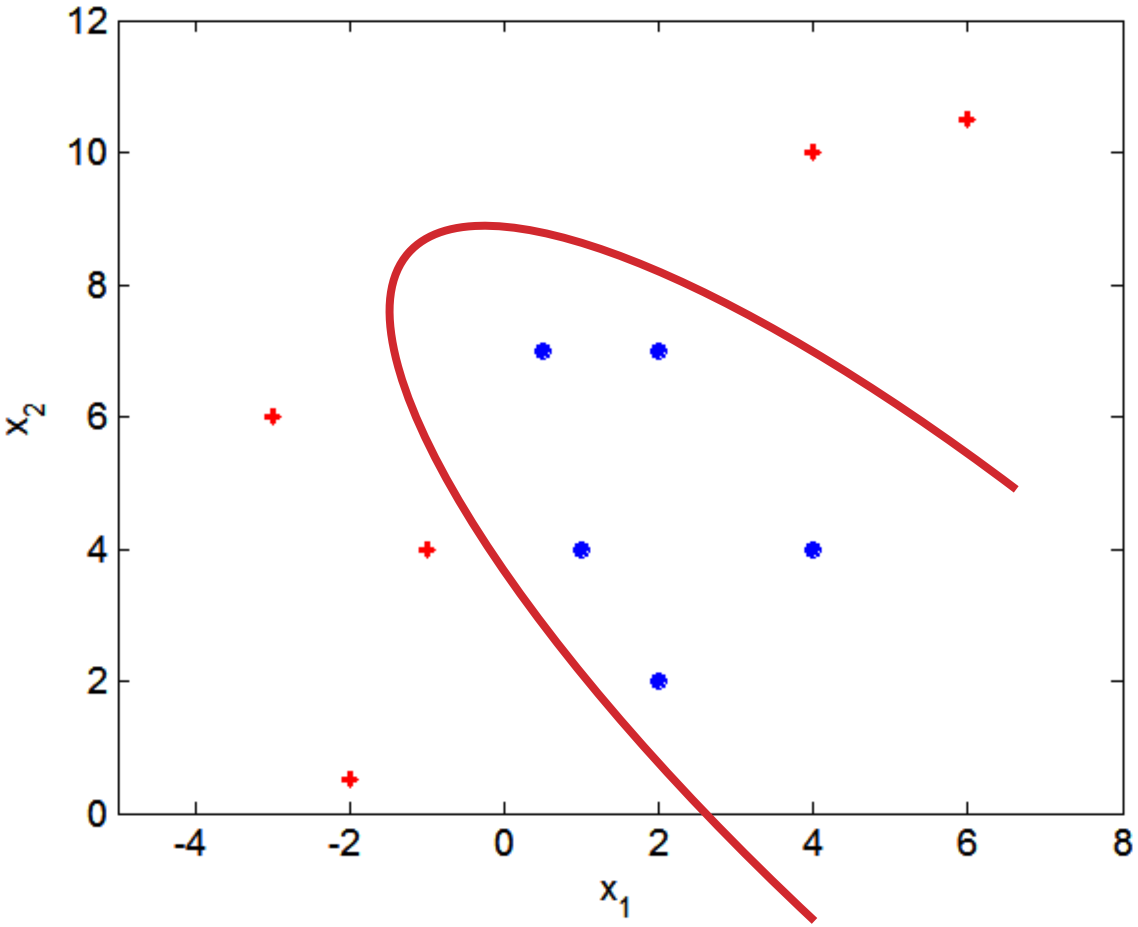
$$L(w) = \frac{||\vec{w}||^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^K\right)$$

- Subject to:

$$\vec{w} \cdot \vec{x} + b \geq 1 - \xi_i \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x} + b \leq -1 + \xi_i \text{ if } y_i = -1$$

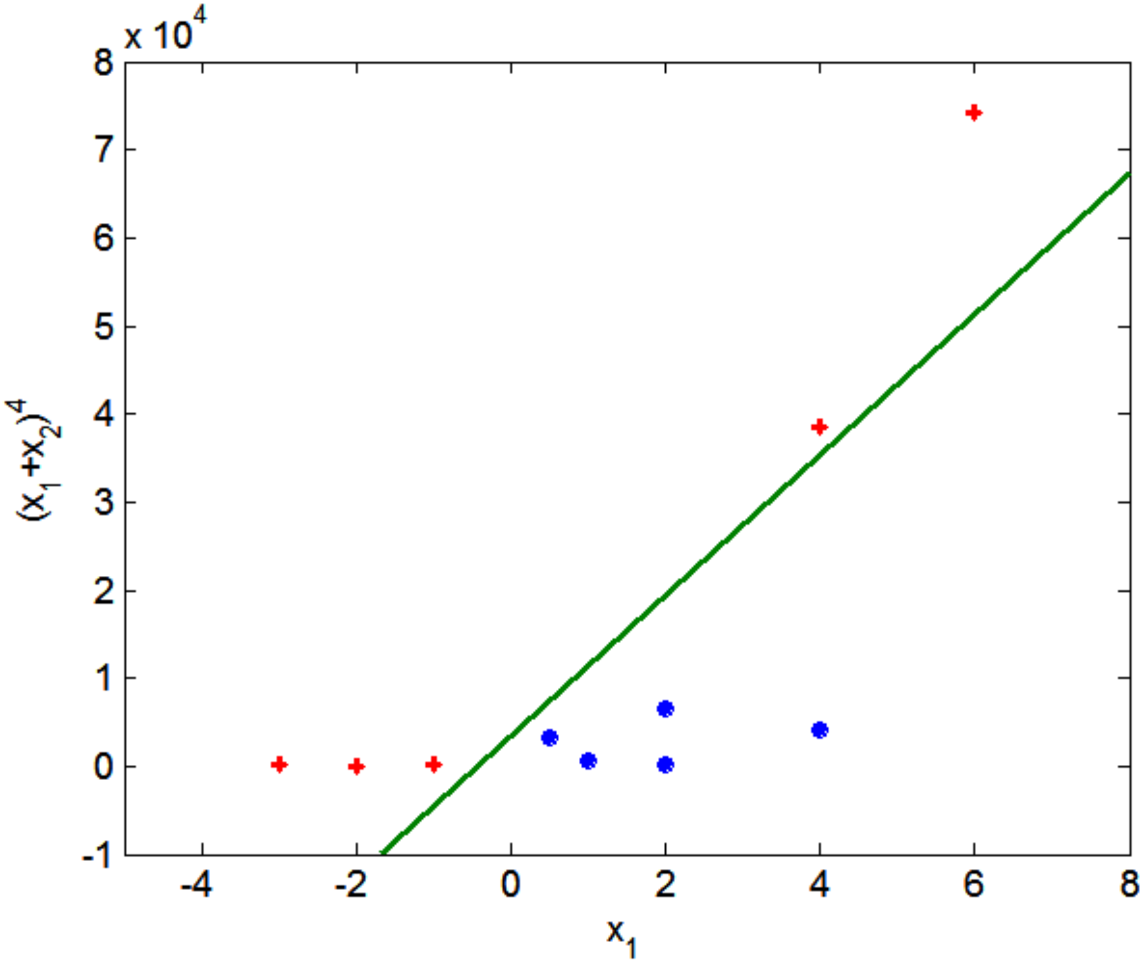
NONLINEAR SUPPORT VECTOR MACHINES

What if the decision boundary is not linear?



NONLINEAR SUPPORT VECTOR MACHINES

Transform data into higher dimensional space



SVMS IN SCIKIT-LEARN

```
from sklearn import svm

# Fit a default SVM classifier to fake data
X = [[0, 0], [1, 1]]
y = [0, 1]
clf = svm.SVC()
clf.fit(X, y)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma='auto',
kernel='rbf', max_iter=-1, probability=False,
random_state=None, shrinking=True, tol=0.001,
verbose=False)
```

Lots of defaults used for hyperparameters – can use cross validation to search for good ones

MODEL SELECTION IN SCIKIT-LEARN

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

# ... Load some raw data into X and y ...
# Split the dataset in two equal parts
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.5, random_state=0)
```

```
# Pick values of hyperparameters you want to consider
tuned_parameters = [{'kernel': ['rbf'],
                         'gamma': [1e-3, 1e-4],
                         'C': [1, 10, 100, 1000]},
                     {'kernel': ['linear'],
                         'C': [1, 10, 100, 1000]}
                    ]
```

MODEL SELECTION IN SCIKIT-LEARN

```
# Perform a complete grid search + cross validation
# for each of the hyperparameter vectors
clf = GridSearchCV(SVC(C=1),
                   tuned_parameters,
                   cv=5,
                   scoring='precision')
clf.fit(X_train, y_train)
```

```
# Now that you've selected good hyperparameters via CV,
# and trained a model on your training data, get an
# estimate of the "true error" on your test set
y_true, y_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, y_pred))
```