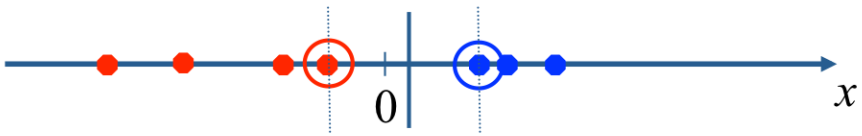
Support Vector Machines: Part 2

Outline

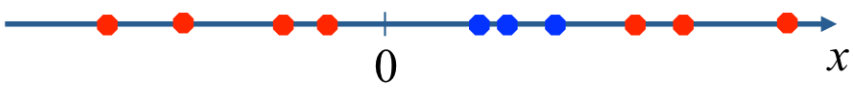
- pre: Linear Classifiers and Hyperplanes
- SVM overview: Geometric Interpretation
- Linear SVM: the Original
- Linear SVM: Soft Margins
- Tomorrow Part II: Non-linear SVMs (kernels)

Non-linear SVMs

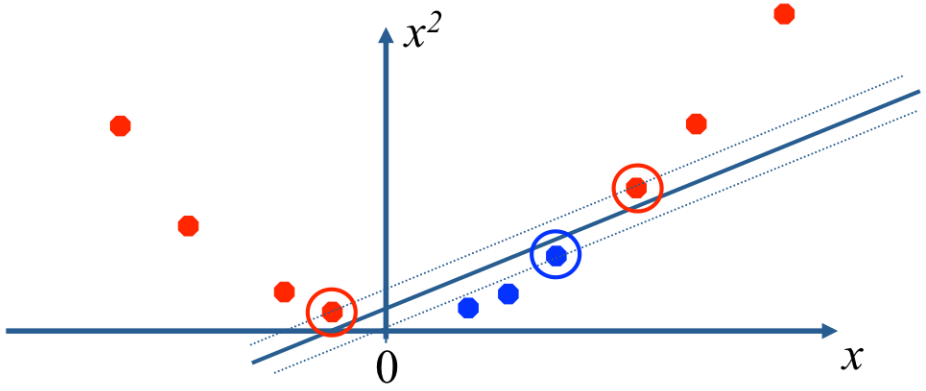
- Datasets that are linearly separable with some noise work out great:

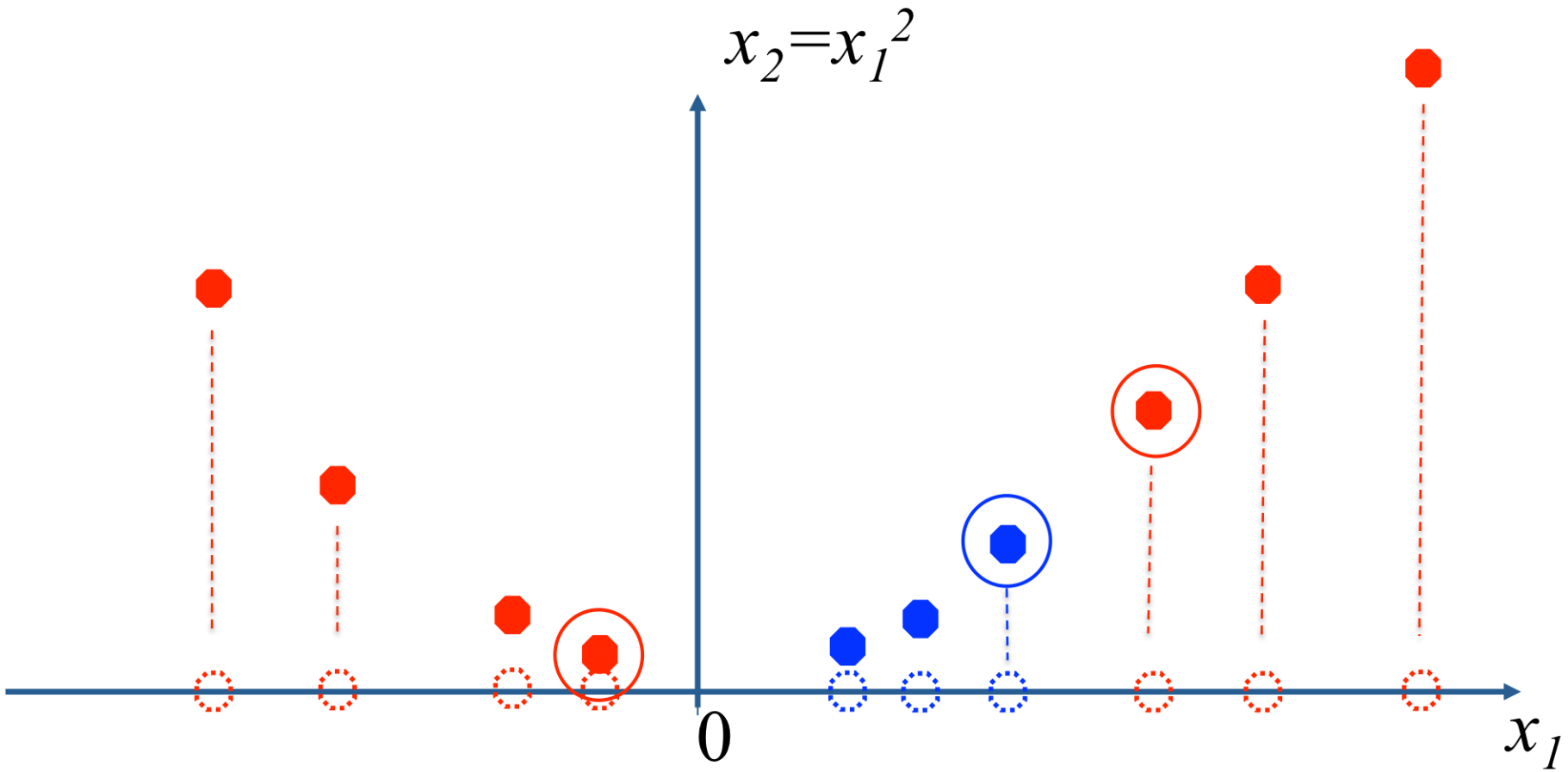


- But what are we going to do if the dataset is just too hard?



- How about… mapping data to a higher-dimensional space:



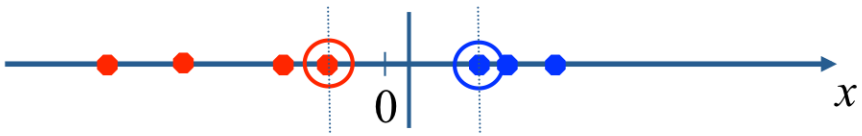


$f : \boldsymbol{R} \rightarrow \boldsymbol{R^2}$

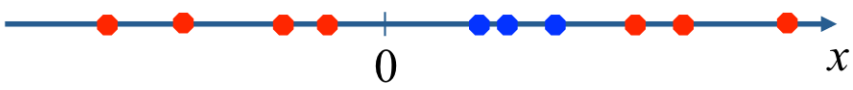
$f(\mathbf{x_1}) \rightarrow f(\mathbf{x_1}, \mathbf{x_1^2})$

Non-linear SVMs

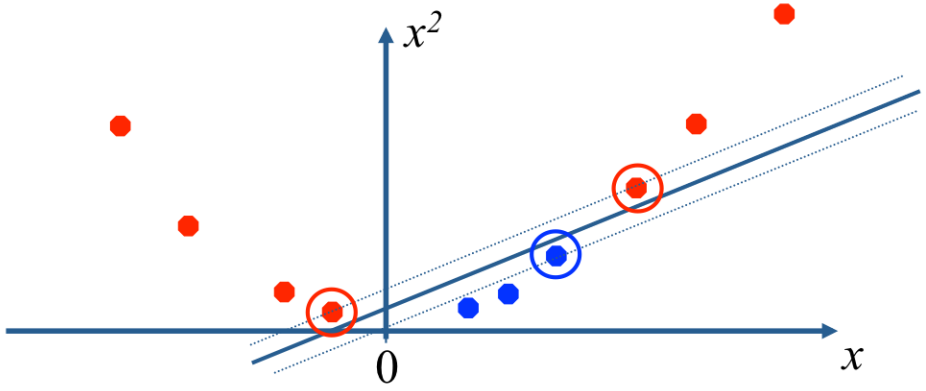
- Datasets that are linearly separable with some noise work out great:

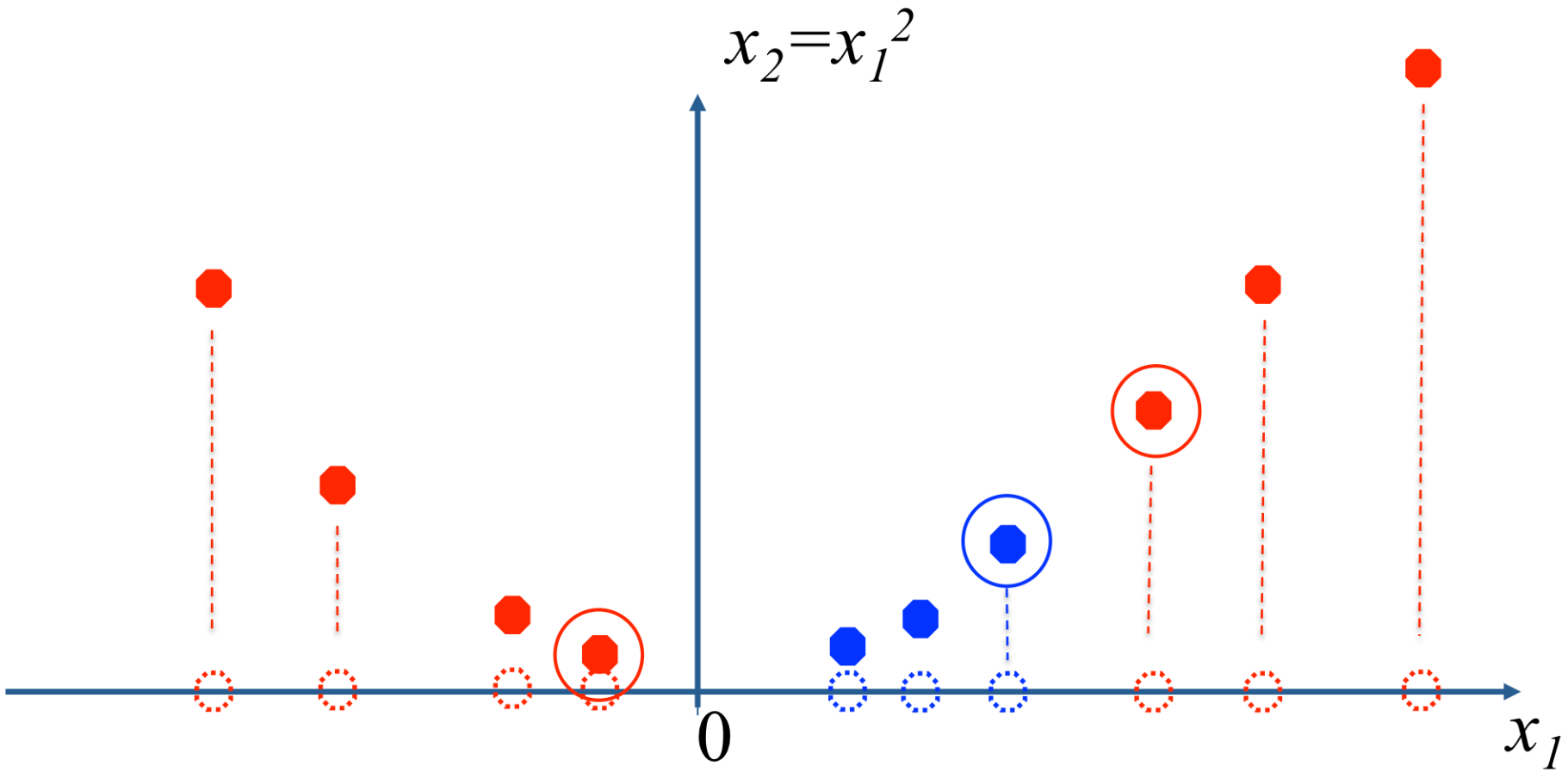


- But what are we going to do if the dataset is just too hard?



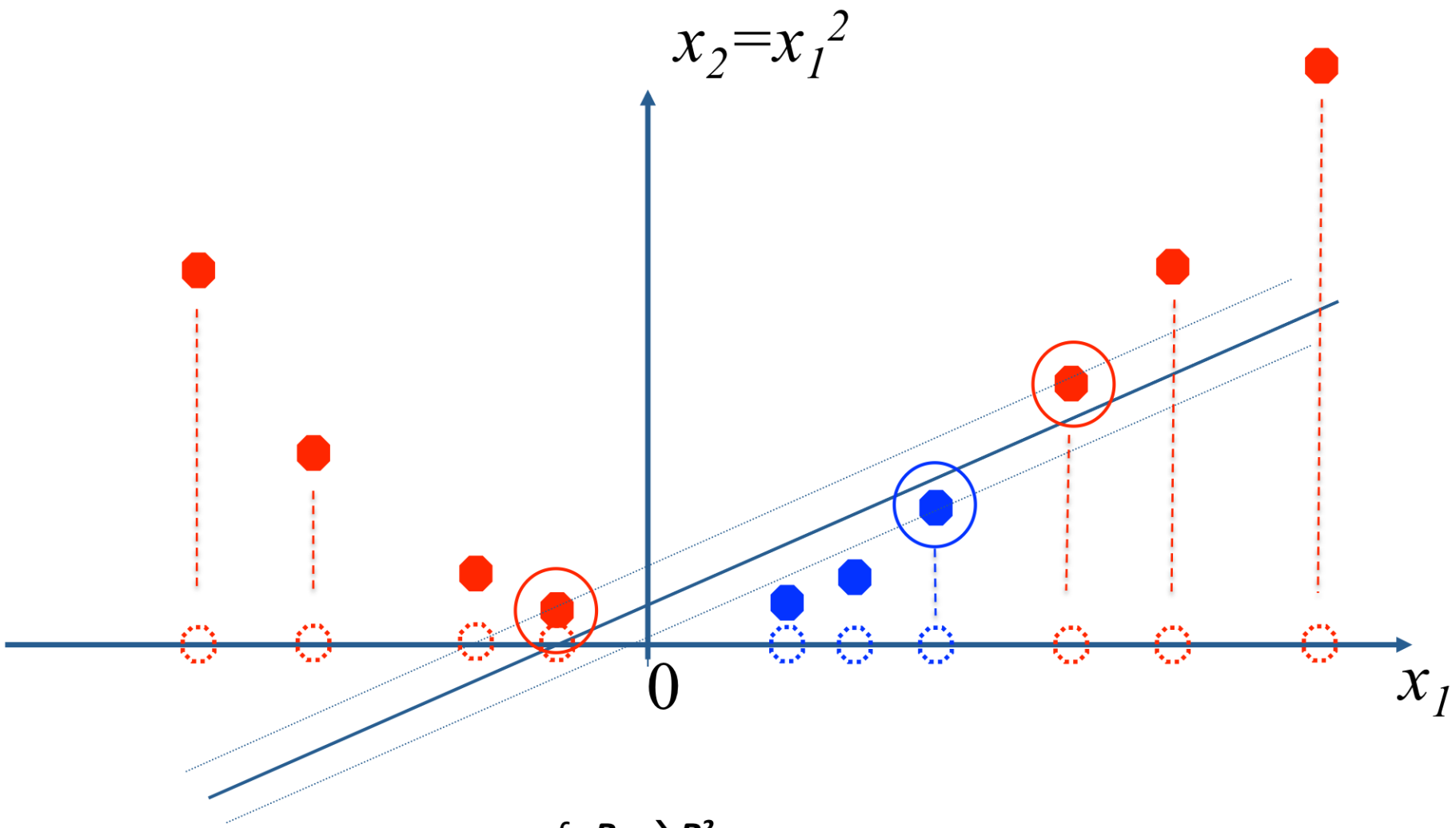
- How about… mapping data to a higher-dimensional space:





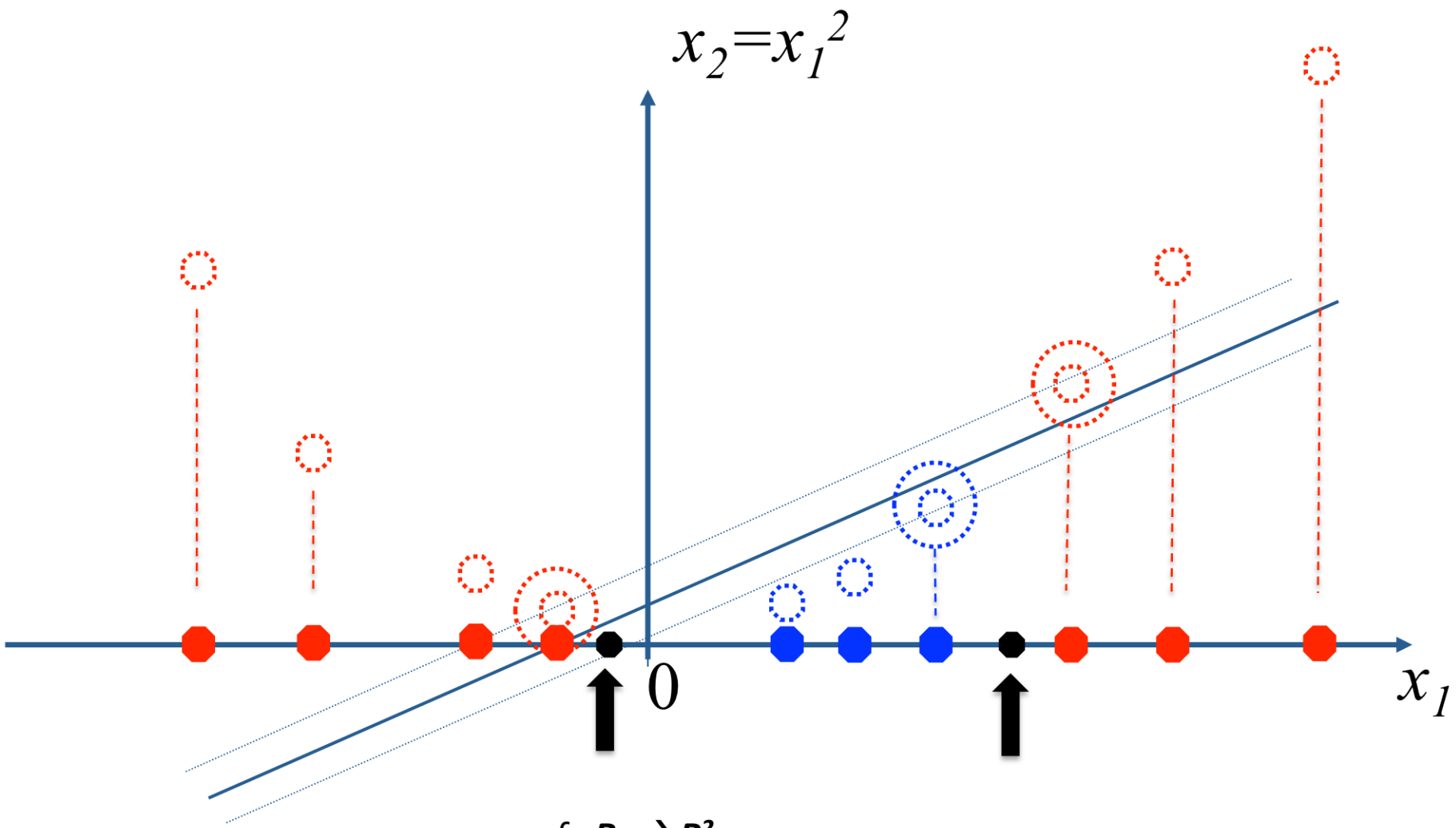
$f : \boldsymbol{R} \rightarrow \boldsymbol{R^2}$

$f(\mathbf{x_1}) \rightarrow f(\mathbf{x_1}, \mathbf{x_1^2})$



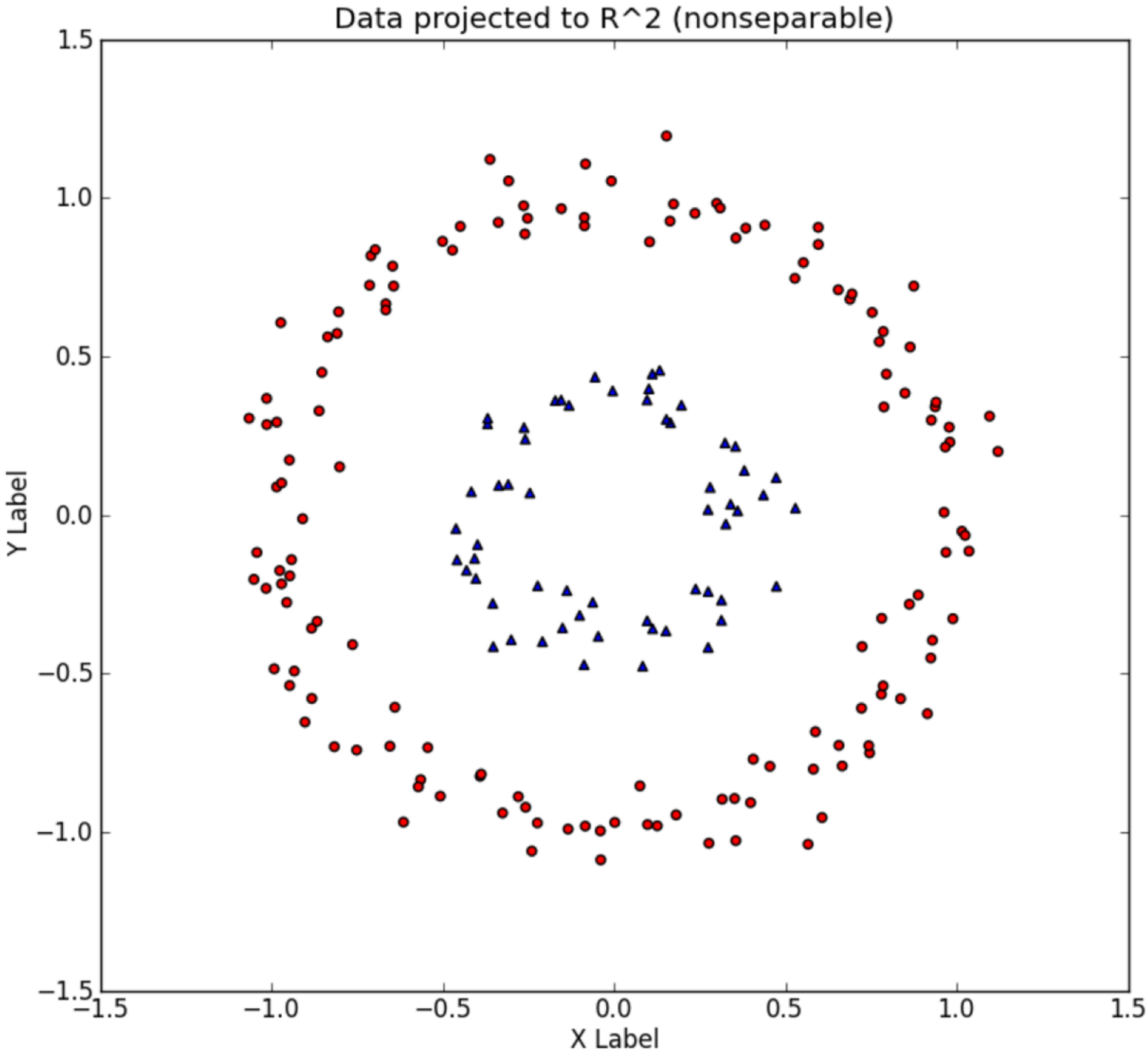
$f : \boldsymbol{R} \rightarrow \boldsymbol{R^2}$

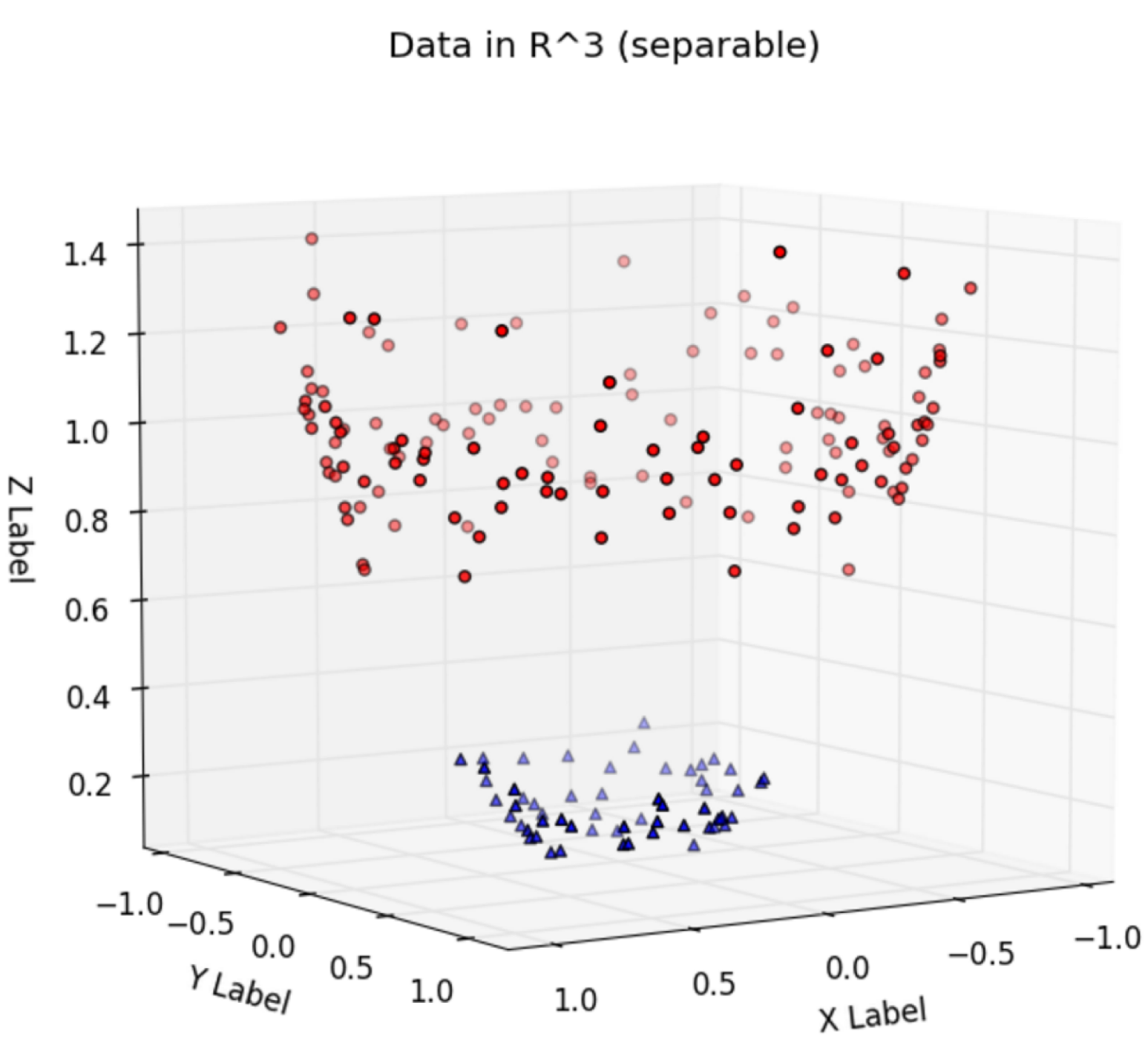
$f(\mathbf{x_1}) \rightarrow f(\mathbf{x_1}, \mathbf{x_1^2})$

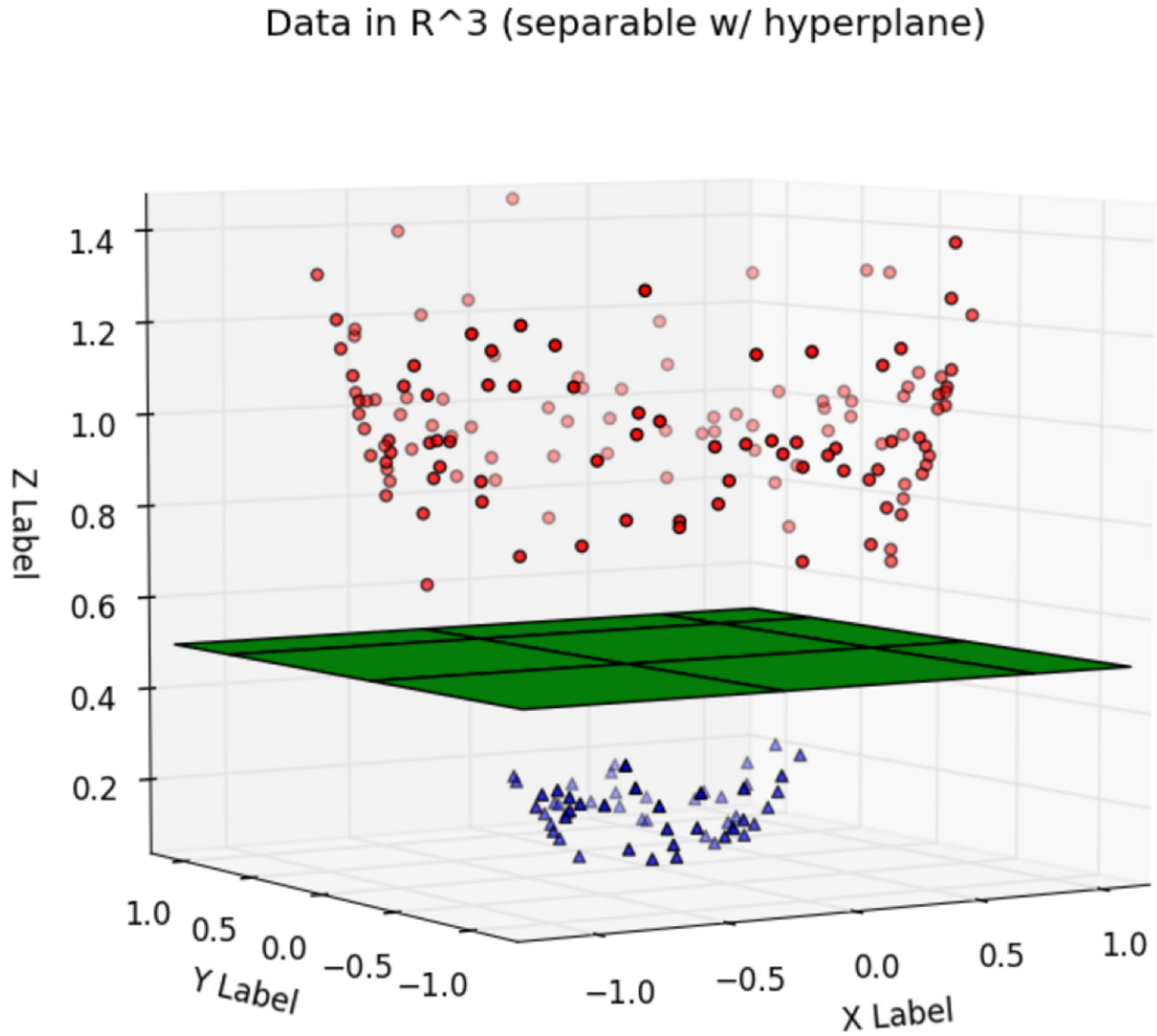


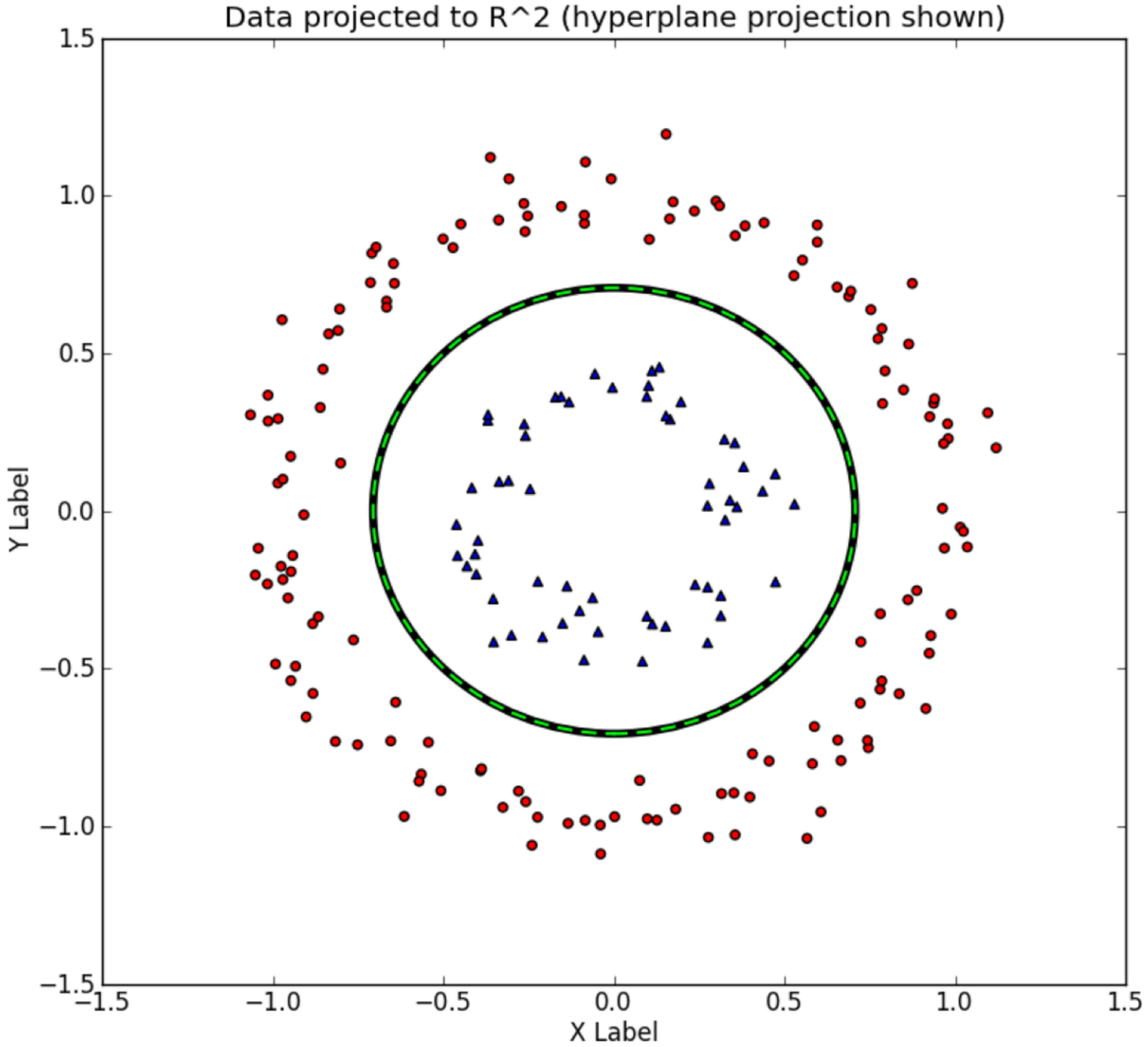
$f : \mathbf{R} \rightarrow \mathbf{R^2}$

$f(\mathbf{x_1}) \rightarrow f(\mathbf{x_1}, \mathbf{x_1^2})$


Data projected to R^2 (nonseparable)
Y Label
X Label
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Data in R^3 (separable)
Z Label
1.4
1.2
1.0
0.8
0.6
0.4
0.2
−1.0
−0.5
0.0
0.5
1.0
Y Label
1.0
0.5
0.0
−0.5
−1.0
X Label


Data in R^3 (separable w/ hyperplane)
Z Label
1.4
1.2
1.0
0.8
0.6
0.4
0.2
1.0
0.5
0.0
−0.5
−1.0
Y Label
−1.0
−0.5
0.0
0.5
1.0
X Label


Data projected to R^2 (hyperplane projection shown)
Y Label
X Label
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

“Kernel Trick”: there’s more

Large data transformations will incur serious computational and memory problems

There exist functions that, given two vectors, implicitly computes the dot product between them in a higher-dimensional space without explicitly transforming them. Such functions are called kernel functions.

Non-linear SVMs: Summary

A kernel function is a function that is equivalent to an inner product in some feature space

Can efficiently learn nonlinear decision boundaries (replacing all dot products in the SVM computation with kernels)

Thus, a kernel function implicitly maps data to a high-dimensional space (without the need to raise dimensionality explicitly).

Interview Time: Kernels

What did you choose this kernel?

- Unfortunately, choosing the 'correct' kernel is a nontrivial task, and may depend on the specific task and dataset at hand.

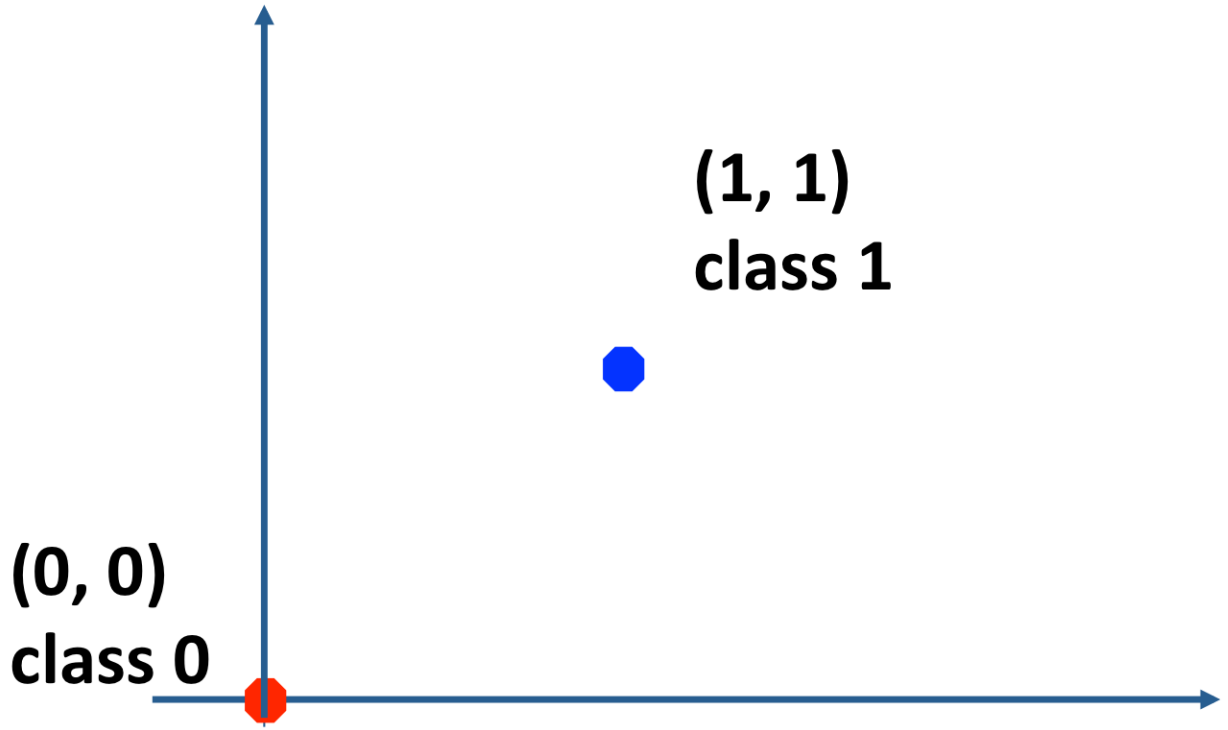
- No matter which kernel you choose, you will need to tune the kernel parameters to get good performance from your classifier.

- Popular parameter-tuning techniques include K-Fold Cross Validation

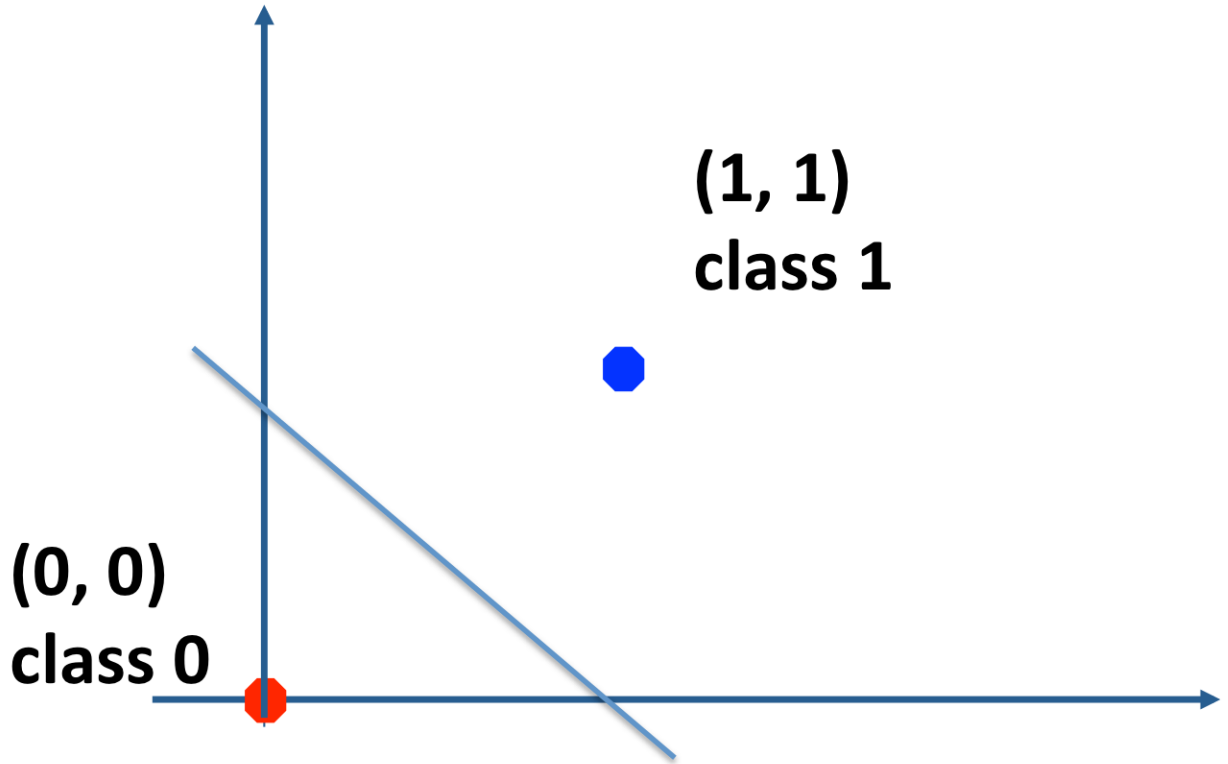
Questions?

Using sklearn and best practices

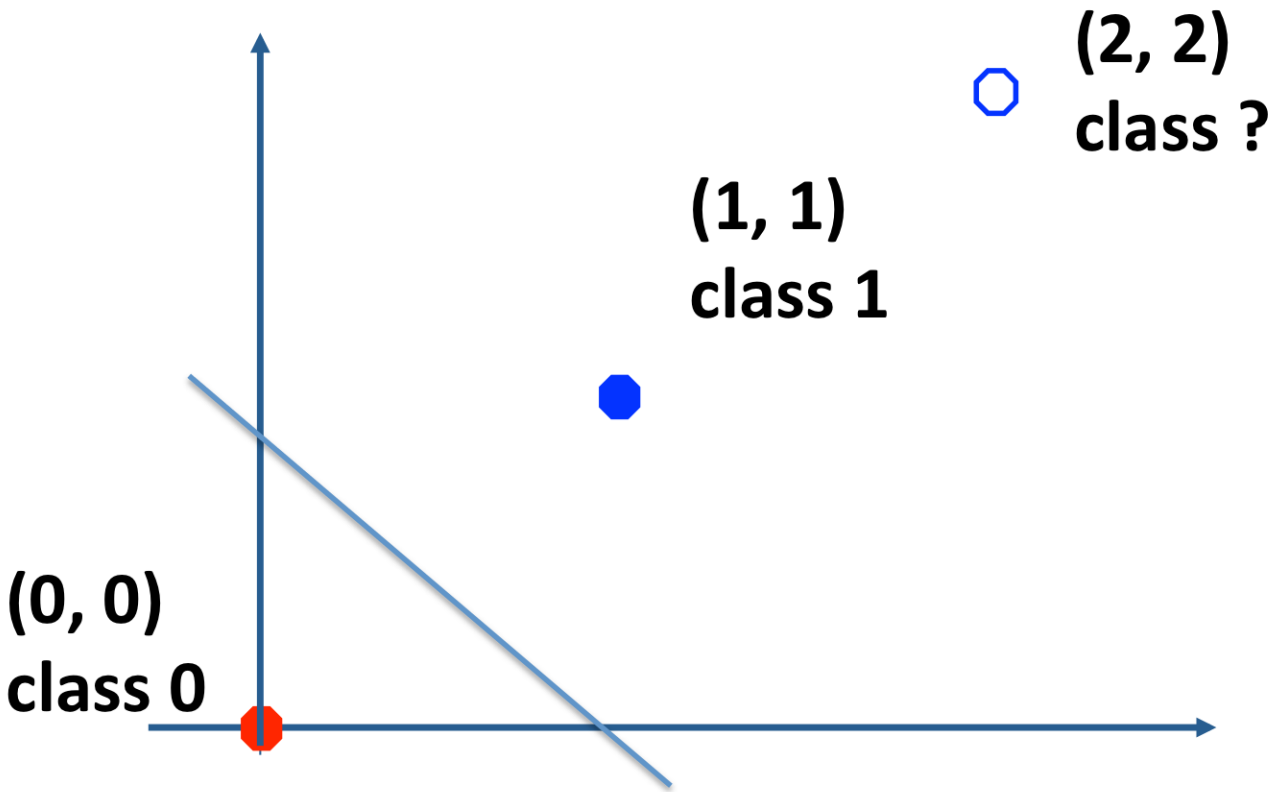
```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
```



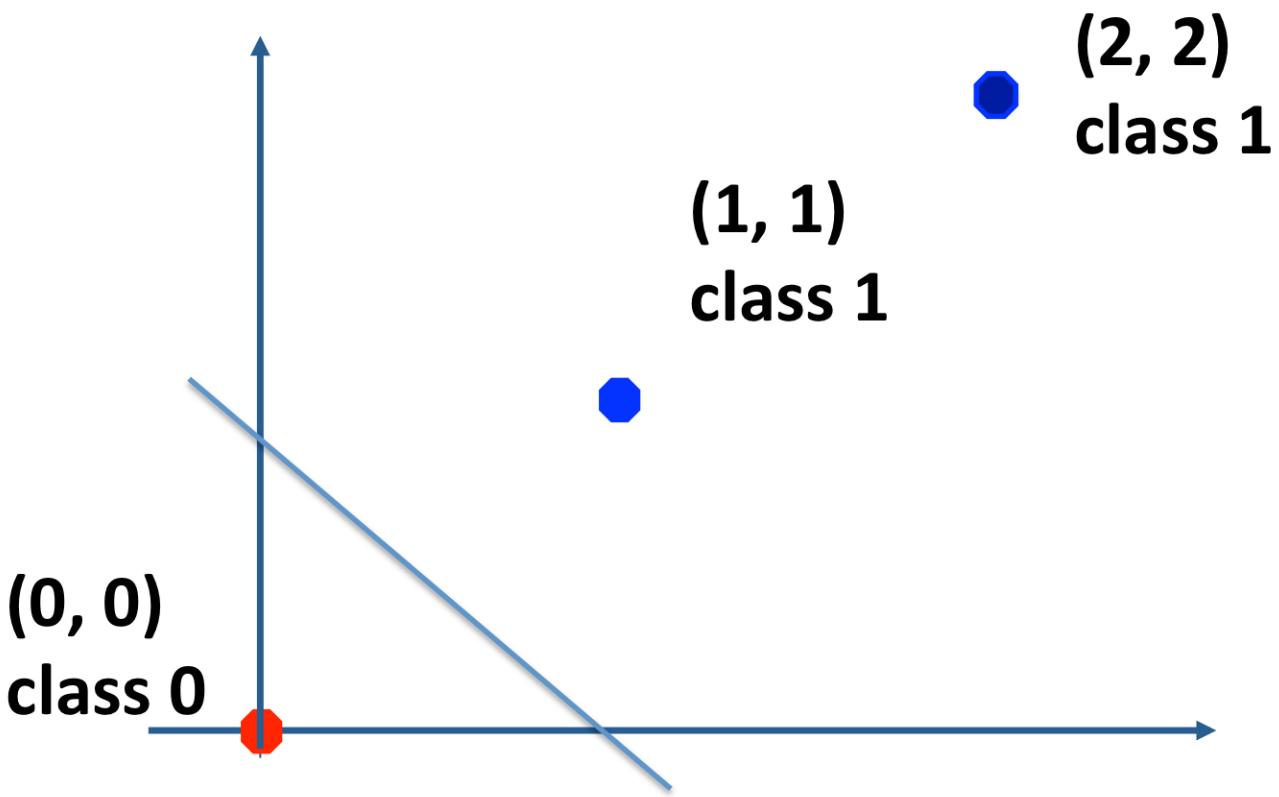
```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
```



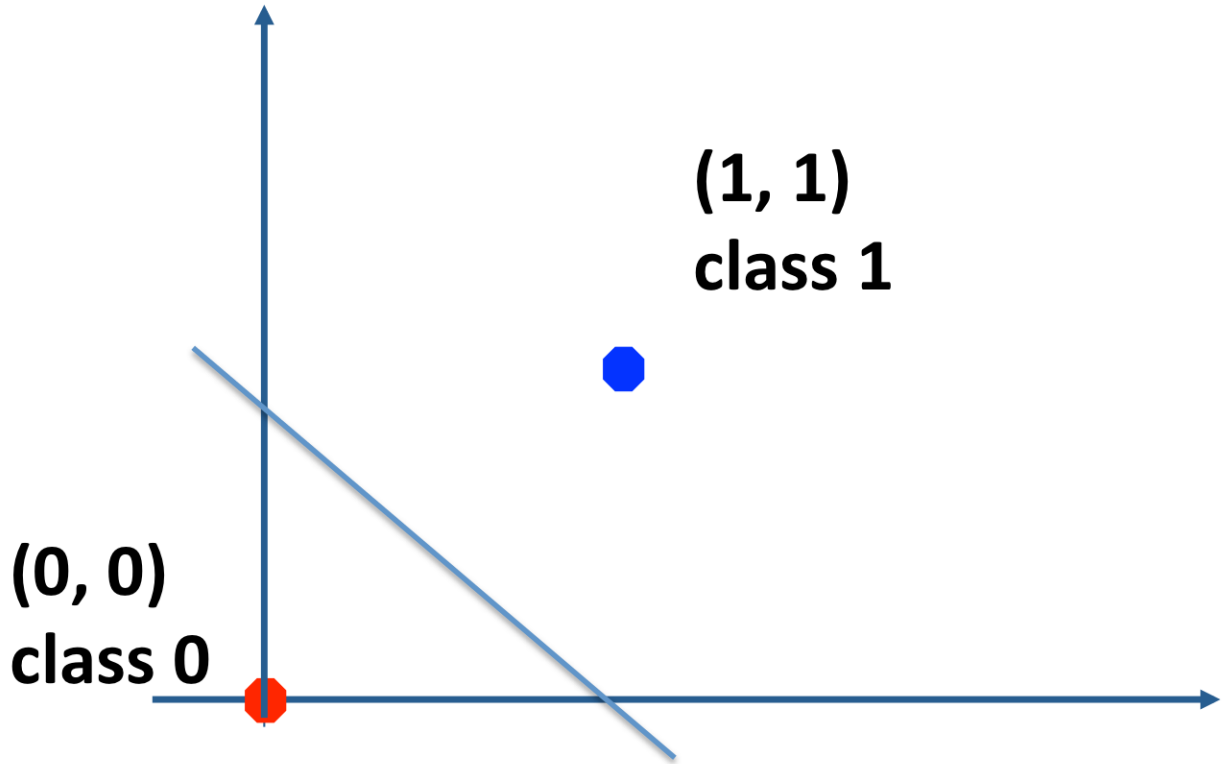
```
>>> clf.predict([[2., 2.]])
array([1])
```



```
>>> clf.predict([[2., 2.]])
array([1])
```

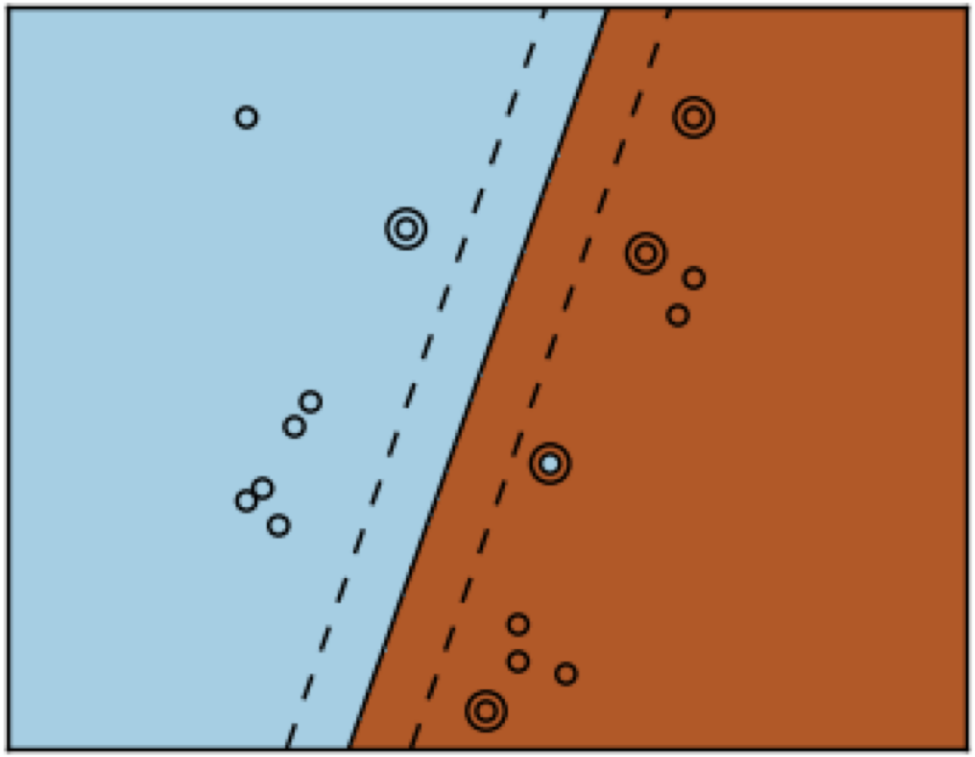


```
>>> # get support vectors
>>> clf.support_vectors_
array([[ 0.,  0.],
       [ 1.,  1.]])
```



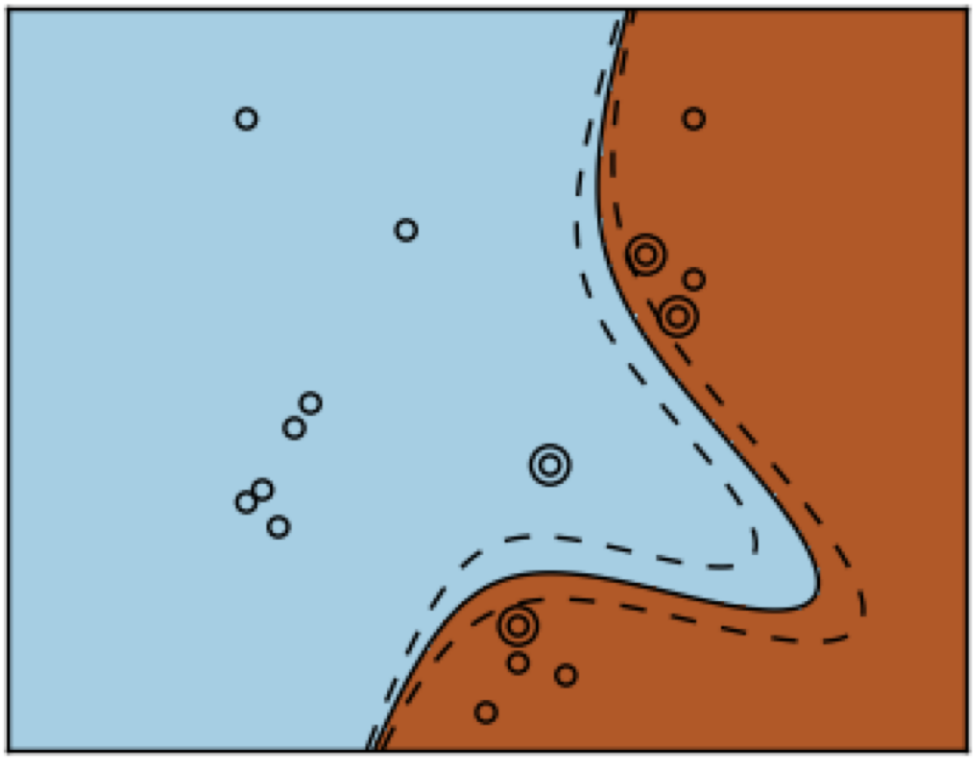
Linear kernel

```
>>> svc = svm.SVC(kernel='linear')
```



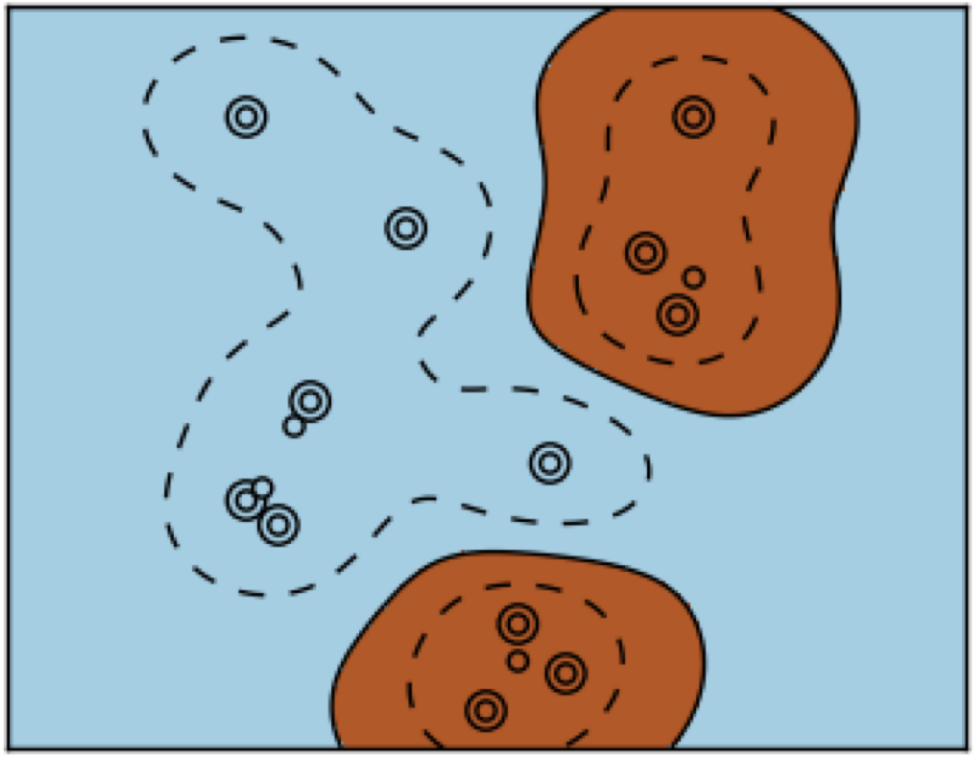
Polynomial kernel

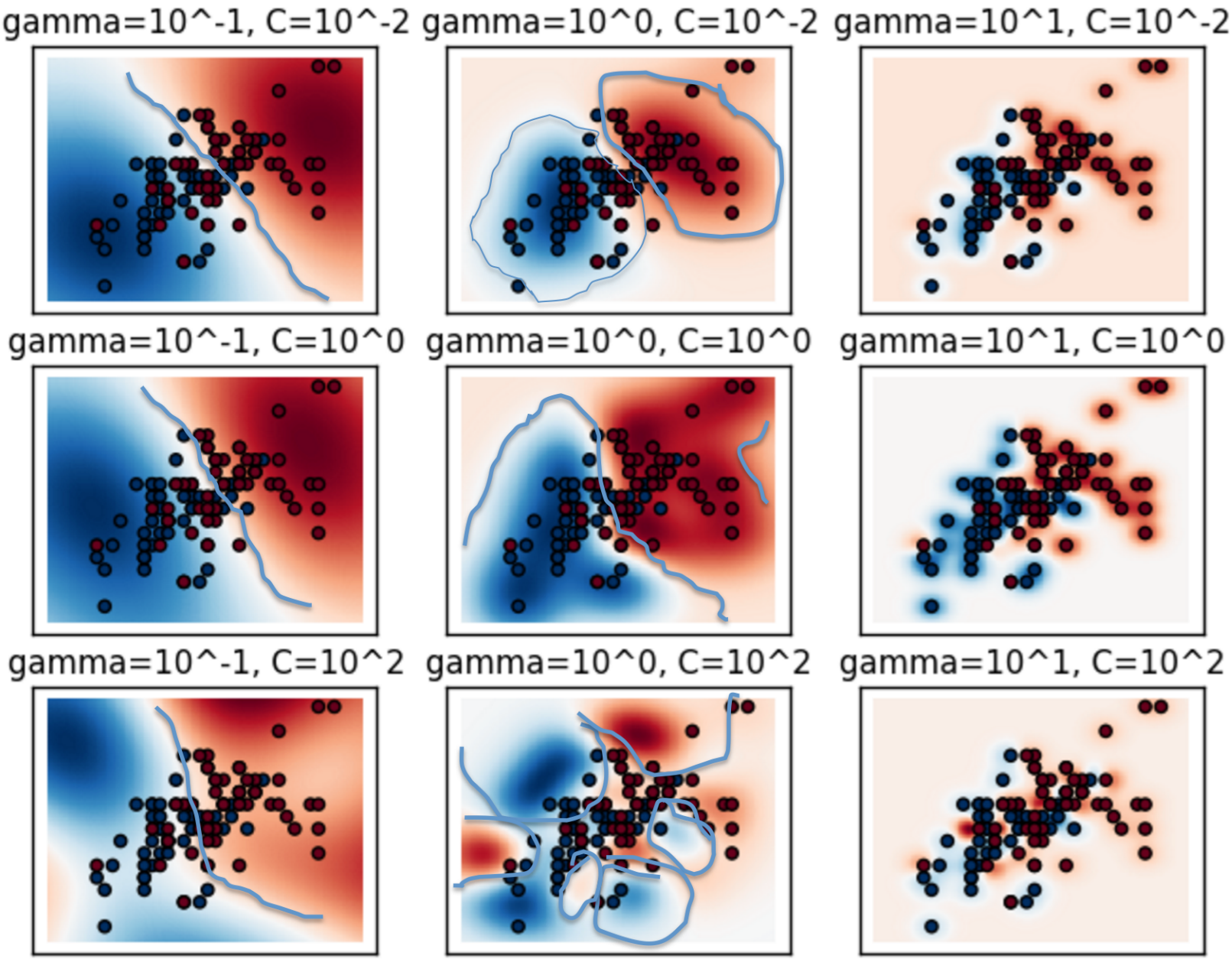
```
>>> svc = svm.SVC(kernel='poly',
...               degree=3)
>>> # degree: polynomial degree
```



RBF (Gaussian) kernel

```
>>> svc = svm.SVC(kernel='rbf')
>>> # gamma: inverse of size of
>>> # radial kernel
```




gamma=10^-1, C=10^-2
gamma=10^0, C=10^-2
gamma=10^1, C=10^-2
gamma=10^-1, C=10^0
gamma=10^0, C=10^0
gamma=10^1, C=10^0
gamma=10^-1, C=10^2
gamma=10^0, C=10^2
gamma=10^1, C=10^2

Interview Time: Implementation

How did you choose your parameters for your RBF kernel?

Interview Time: Implementation

How did you choose your parameters for your RBF kernel?

- proper choice of C and gamma is critical to the SVM's performance
- use ***sklearn.grid_search.GridSearchCV*** with C and gamma spaced exponentially apart

Final thoughts

Normalize before SVM (not scale-invariant)

Why not always SVM?

When to use?

When to use SVMs, Linear Regression, etc.

A lot of features, but small data
(10K features, 1000 training examples)

Best you can do is a simple model, go for Linear Regression or LinearSVC

When to use SVMs, Linear Regression, etc.

Few features, decent data
(5-100 features, 10K training examples)

Go for a Gaussian Kernel SVC, rock the hell out of it.

When to use SVMs, Linear Regression, etc.

Few features, lots of data
(5-100 features, 100K training examples)

With that much data, you can extract more from weaker features. Add more features. Use Linear Regression or LinearSVC

(Gaussian kernel SVC is too slow with large data)

Questions?