# *Support Vector Machines 2*

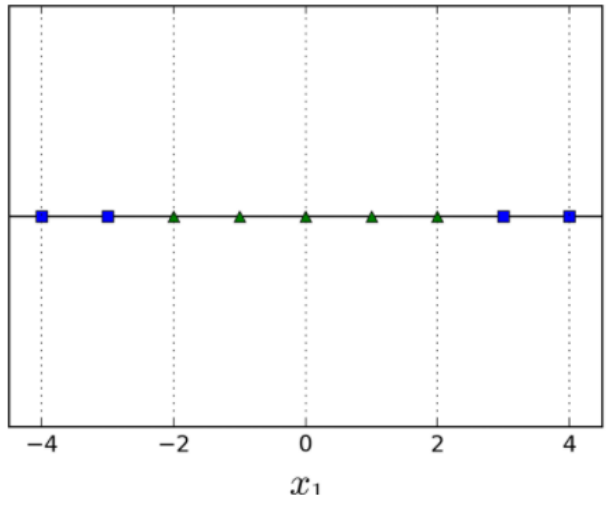
Glenn Bruns
CSUMB

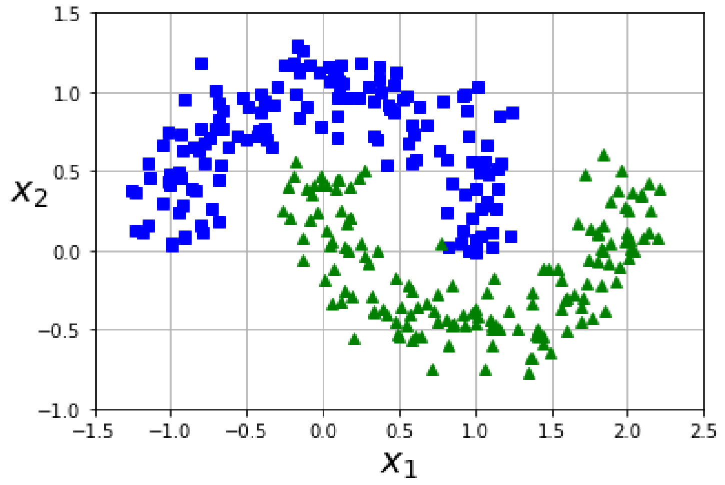
## Learning outcomes

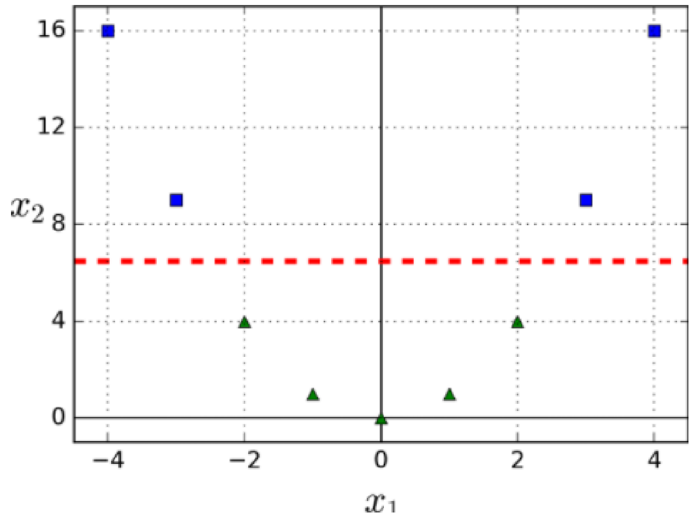
After this lecture you should be able to:

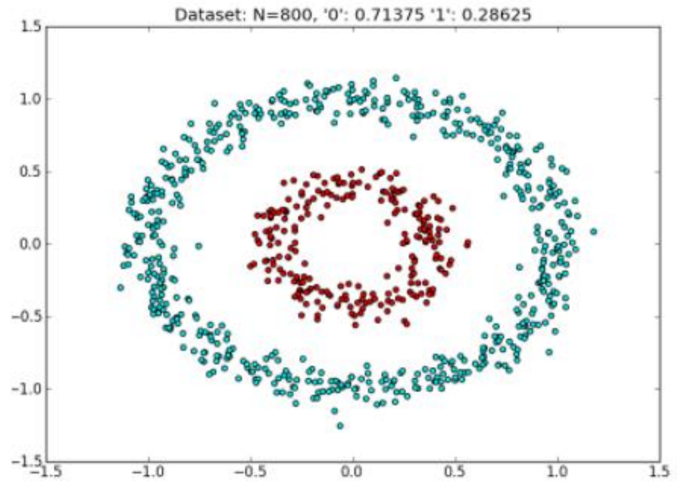
- ❑ explain the benefit of SVM when using polynomial or similarity features
- ❑ choose, and use, SVM kernels
- ❑ use SVMs for both classification and regression

# Classes are often not linearly separable









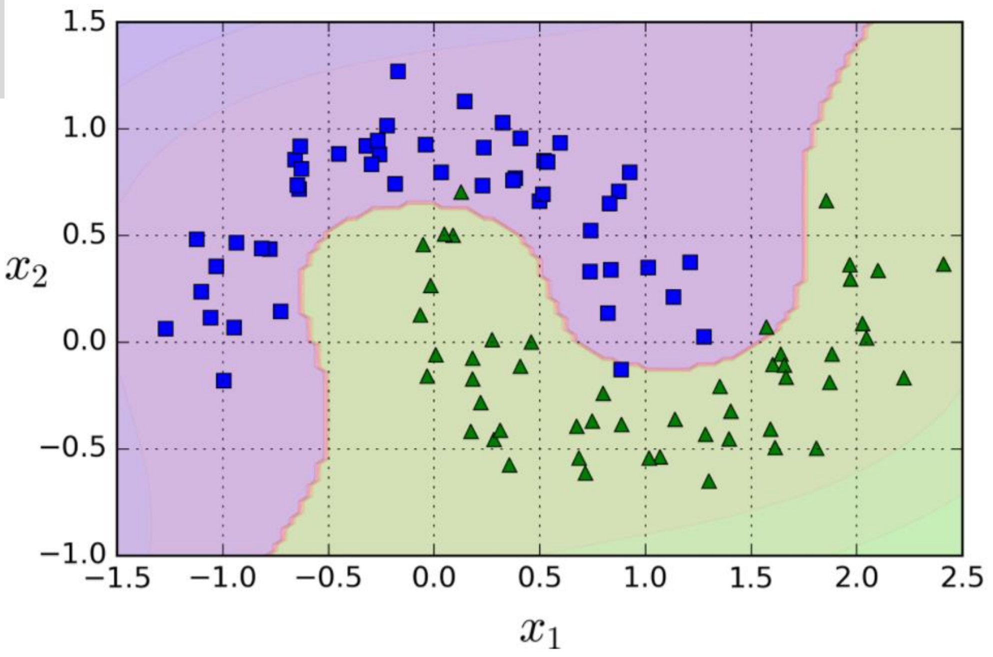
sources: Geron text, eric-kim.net, sebastianraschka.com

# Add polynomial features

```
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

polynomial_svm_clf = Pipeline((
   ("poly_features", PolynomialFeatures(degree=3)),
   ("scaler", StandardScaler()),
   ("svm_clf", LinearSVC(C=10, loss="hinge"))
))

polynomial_svm_clf.fit(X, y)
```



source: Geron text

## Details of PolynomialFeatures

If our features are a,b and the degree parameter is 2, we get all polynomial combinations of the features with degree at most 2:

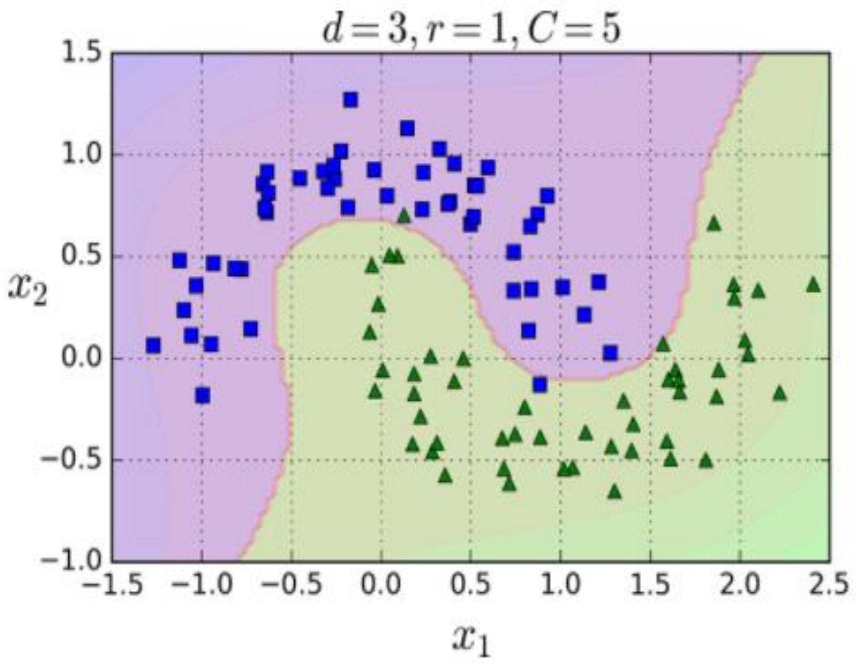
$$1, a, b, a^2, ab, b^2$$

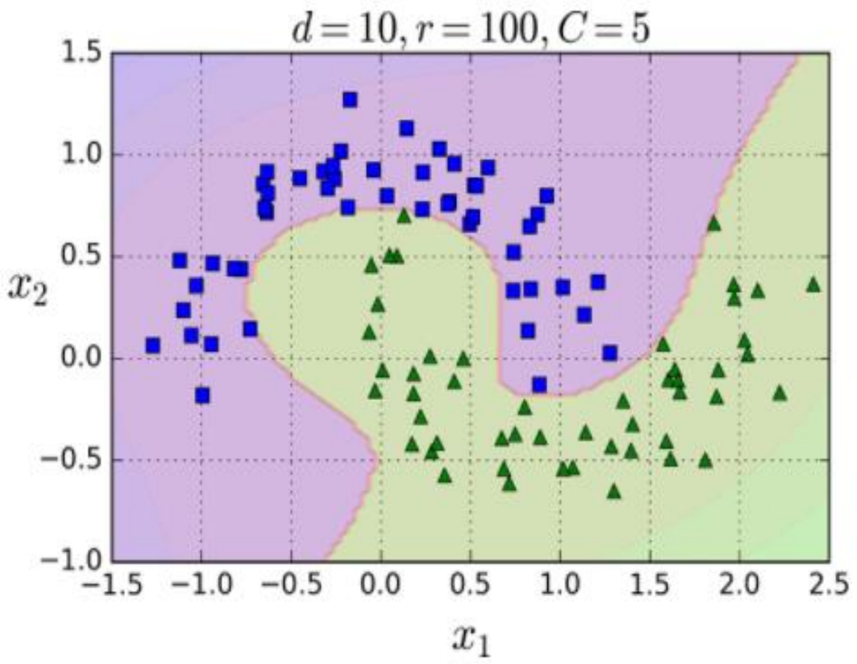
```
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(2)
>>> poly.fit_transform(X)
array([[  1.,   0.,   1.,   0.,   0.,   1.],
       [  1.,   2.,   3.,   4.,   6.,   9.],
       [  1.,   4.,   5.,  16.,  20.,  25.]])
>>> poly = PolynomialFeatures(interaction_only=True)
>>> poly.fit_transform(X)
array([[  1.,   0.,   1.,   0.],
       [  1.,   2.,   3.,   6.],
       [  1.,   4.,   5.,  20.]])
```

# Polynomial features with a “kernel”

```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline((
    ("scaler", StandardScaler()),
    ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
))
poly_kernel_svm_clf.fit(X, y)
```

support vector classifier with a “polynomial kernel”

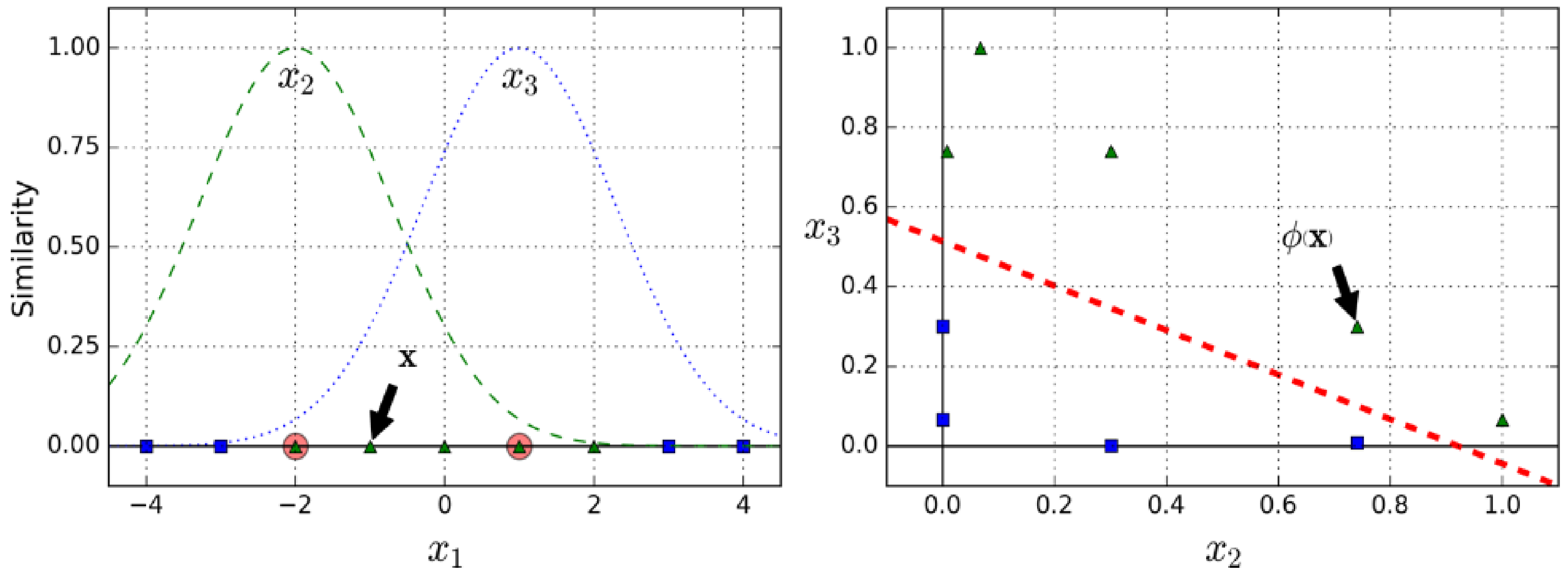




degree: maximum polynomial degree,
r/coef0: controls influence of higher degrees vs. lower degrees

source: Geron text

# Similarity features

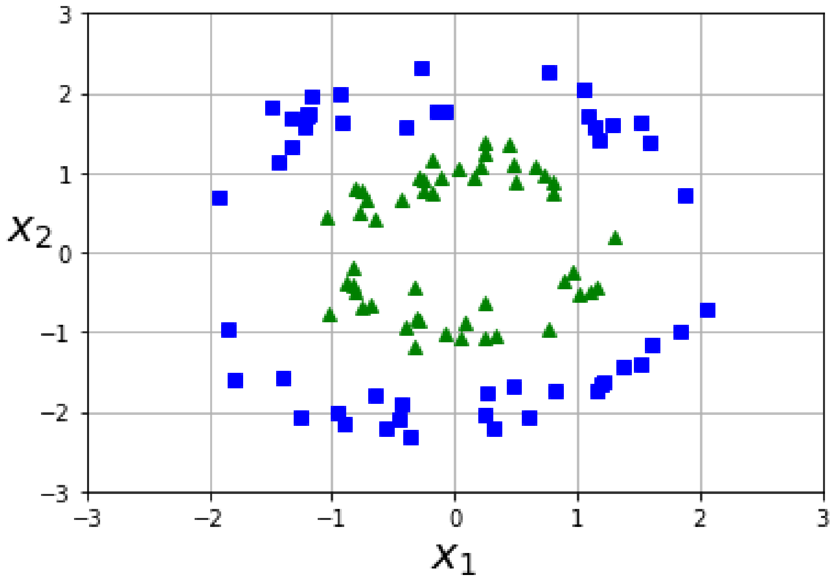


For training example $\mathbf{x}$, value of new feature $x_2$ is the "similarity" of $\mathbf{x}$ to the training point at -2.

Similarity is measured using the Gaussian distribution.

What is the value of feature $x_3$ for the blue square with $x_1 = 3$?

# Similarity features example



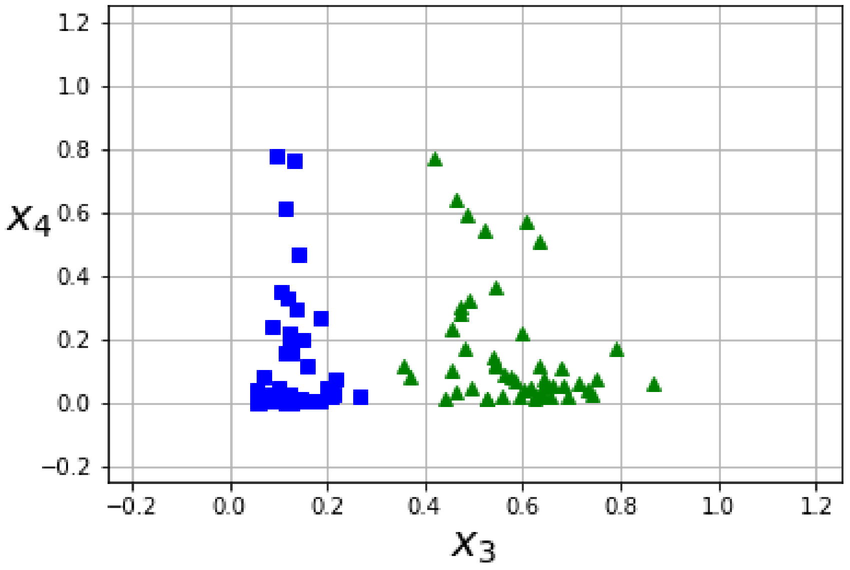
What should we use as "landmark points" for this example?

What about the points at:

(2, 0)

(0, 0)

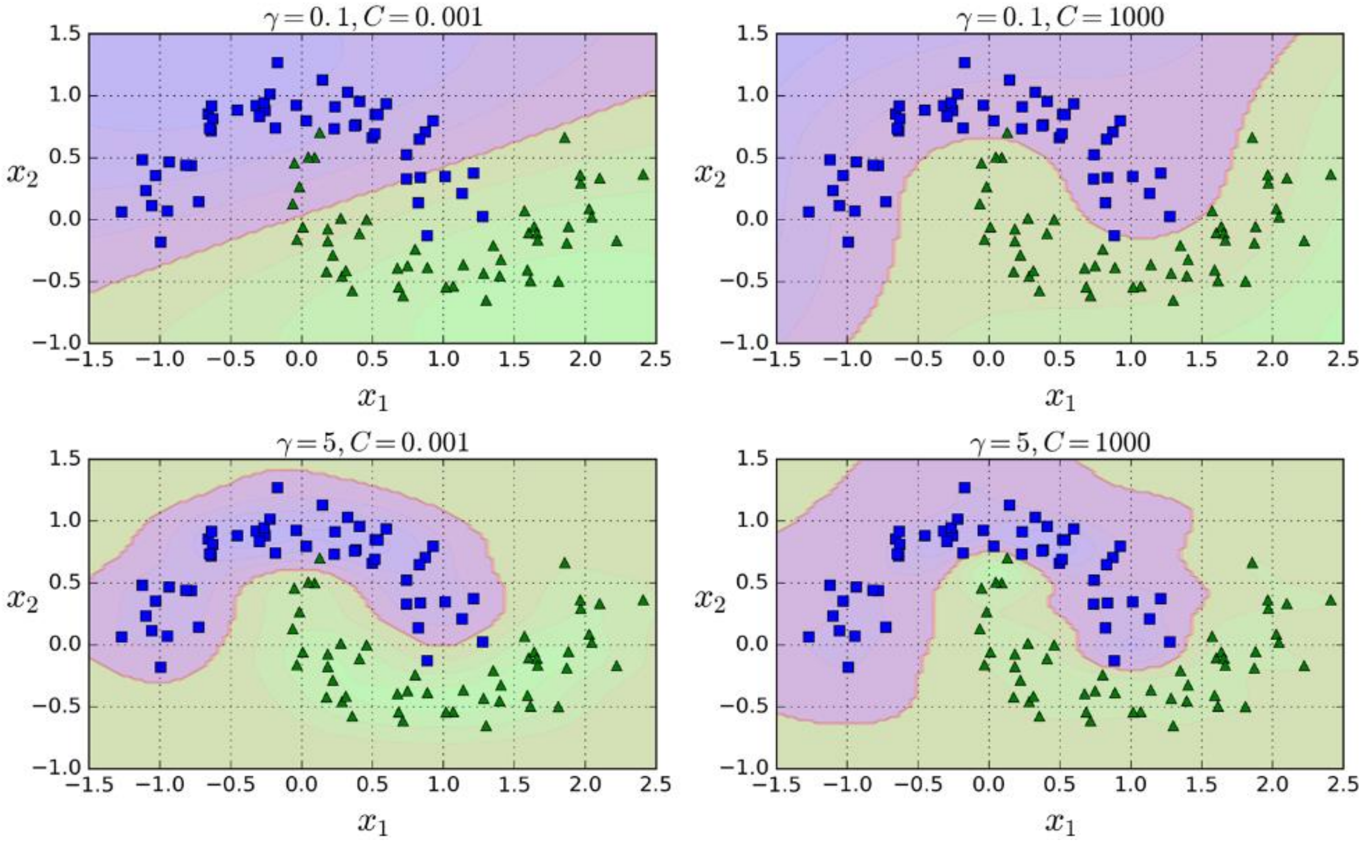
?



# Similarity features with a kernel

```
rbf_kernel_svm_clf = Pipeline((
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ))
rbf_kernel_svm_clf.fit(X, y)
```

support vector classifier with a "Gaussian RBF kernel"



$\gamma$: larger value makes bell-shaped curver narrower; reduces landmark's "range of influence"

source: Geron text

## Kernels and the kernel trick

Most popular kernels: polynomial and Gaussian RBF, by far.

Kernel trick: you get the same result as adding polynomial features without actually adding them.

Same with features related to other kernels.

## Practical guide to SV classification

See Hsu, Chang, and Lin, “A Practical Guide to Support Vector Classification”, 2016.

Recommended practice:

1. transform data to numeric matrix
2. scale the data
3. consider using the RBF kernel
4. use cross-validation to find the best values of parameters $C$ and $\gamma$
5. use the best values of $C$ and $\gamma$ for training with the whole training set
6. test

## Regression with SVMs

SVMs support regression by a modification of the cost function.

Recall: cost function in linear regression:

- sum of squared residuals

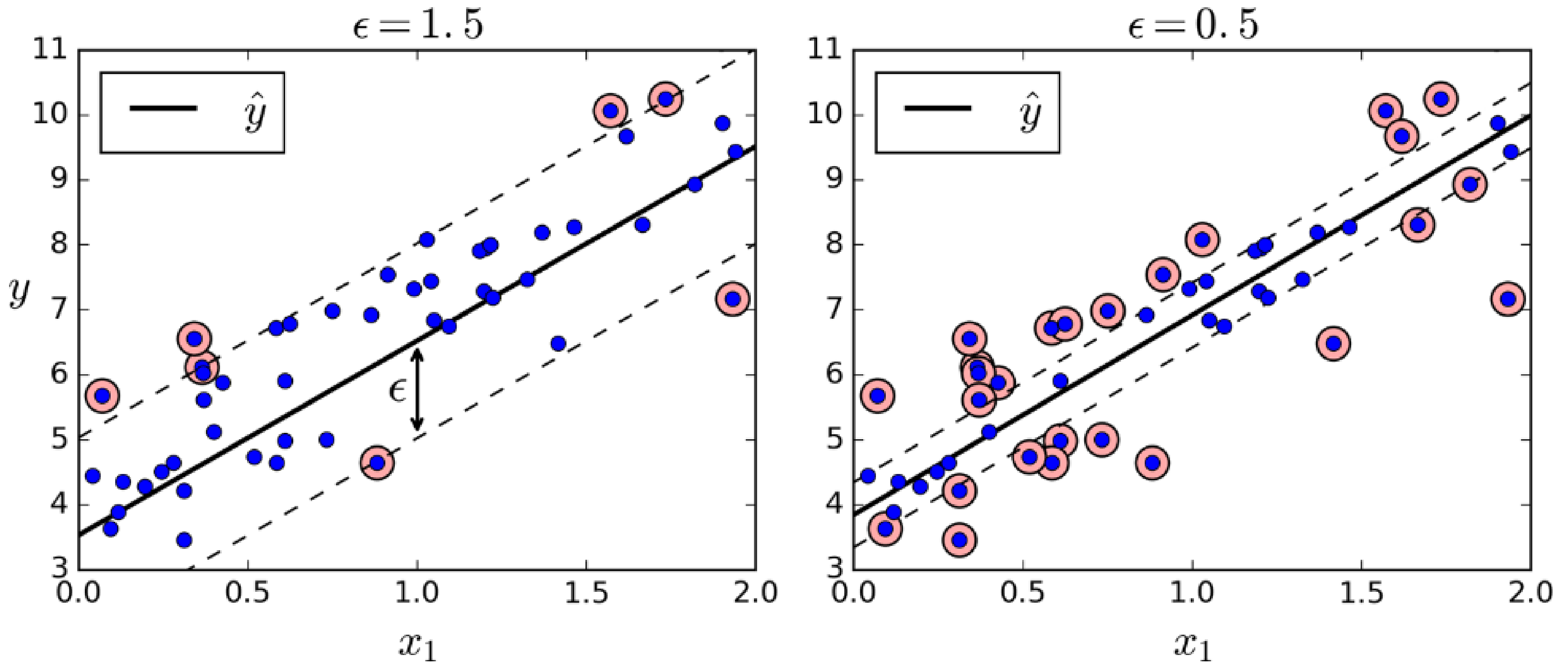
Cost function for SVM regression:

- sum of residuals having an absolute value greater than some constant

Alternative way of saying it:

- try to keep instances on the street; limit instances off the street

# Regression with SVMs



Only “off the street” training instances influence the fit.

Kernels, as before, can be used on regression problems with non-linear data.

See LinearSVR in sklearn.

## Computational cost (sklearn)

| | time complexity | scaling required | kernel trick |
|---|---|---|---|
| LinearSVC | $O(m \times n)$ | Yes | No |
| SGDClassifier | $O(m \times n)$ | Yes | No |
| SVC | $O(m^2 \times n)$ to $O(m^3 \times n)$ | Yes | Yes |

m = # training instances, n = # features

- SVC:
  - scales well with number of features
  - good for small to medium data sets
  - gets very slow as the number of training instances increases

## Summary

SVMs can handle non-linear decision boundaries.

The “kernel trick” make it possible to get the benefit of polynomial features without the usual cost.

Another popular kernel (Gaussian RBF kernel) supports similarity features.

SVMs can also be used for regression.