Support Vector Machines

Igor Wojnicki

March 17, 2023

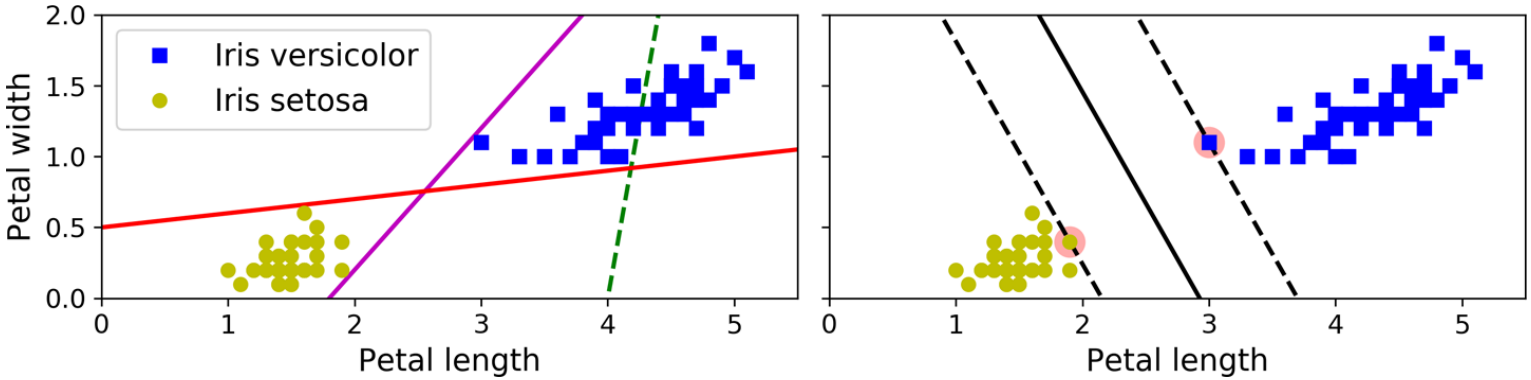
Plan prezentacji

Klasyfikacja

Regresja

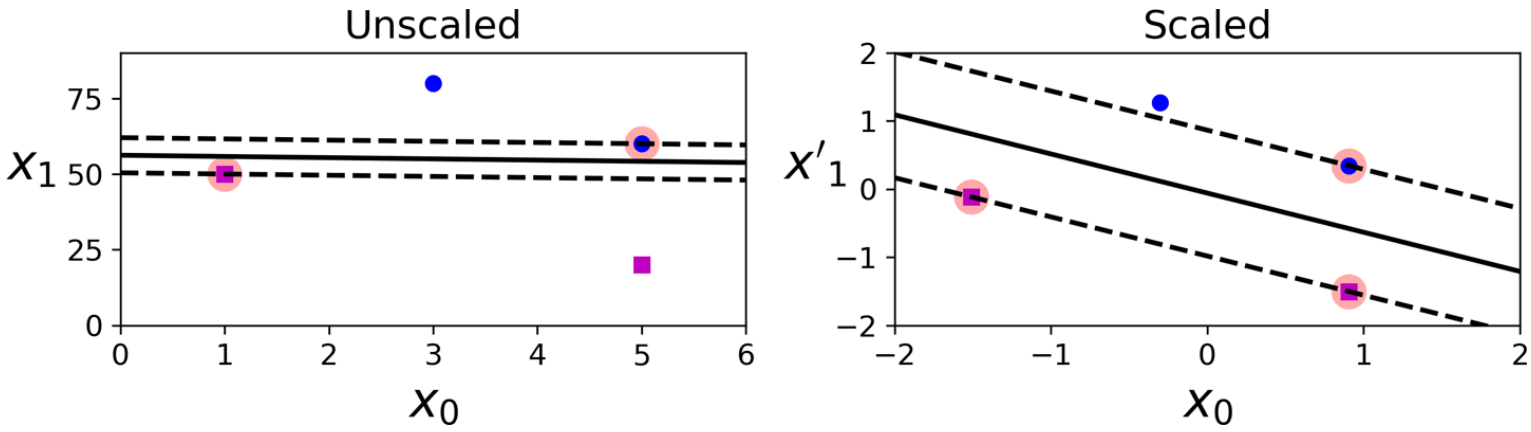
Jak działa SVM?

Maszyna Wektorów Nośnych



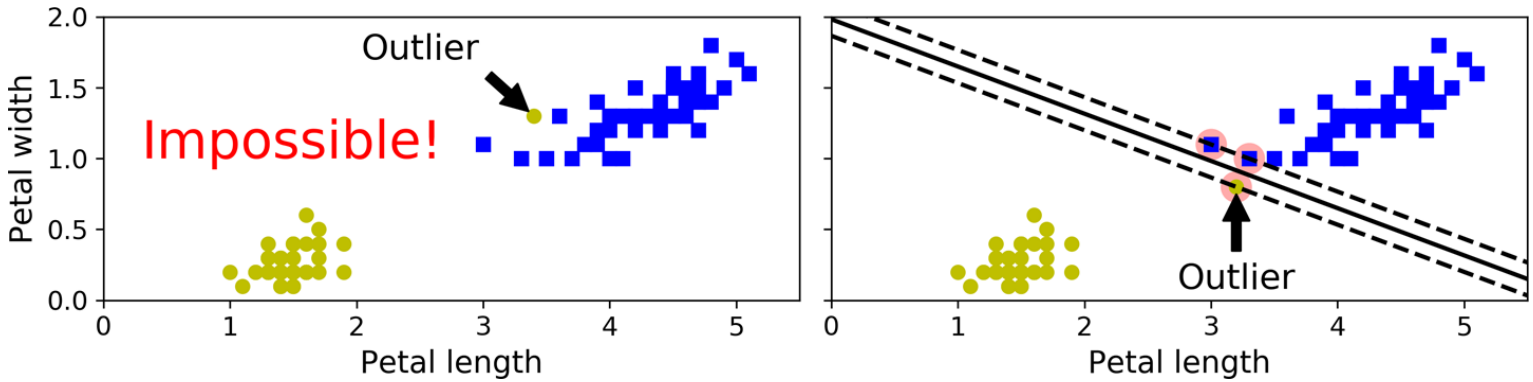
- liniowa separacja (. . . i nie tylko)
- liniowe klasyfikatory
- margines klasyfikacji: utworzenie jak najszerszej ulicy pomiędzy klasami
- dodanie nowych instancji poza "ulicą" nie wpływa na klasyfikacje
- granica klas jest oparta (supported) o instancje leżące na granicy "ulicy"
- instancje te nazywane są *support vectors* (wektory nośne)

Uwaga na skalę cech



- Klasyfikacja na oryginalnych wartościach cech może być marna...

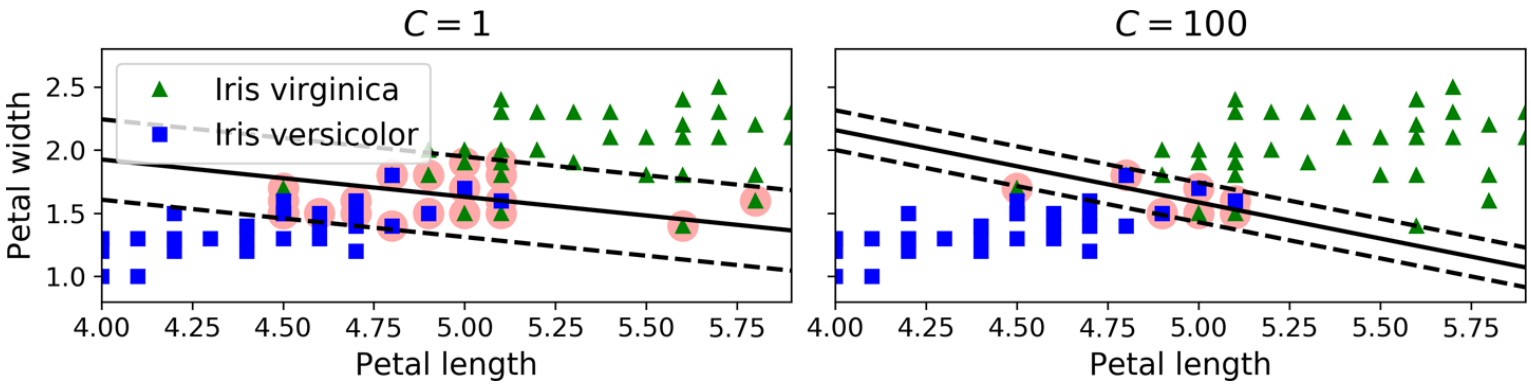
Problemy z klasyfikacją, elementy/wartości odstające, outliers



- Gdzie powinna być droga?
- Jaka szeroka?

Soft Margin Classification

- Równowaga:
  - maksymalna szerokością "drogi"
  - ograniczenie naruszeń marginesu klasyfikacji



- hiperparametr C
- jeżeli *overfitting* należy zmniejszyć C

SVM, uczenie

```
1  import numpy as np
2  from sklearn import datasets
3  from sklearn.pipeline import Pipeline
4  from sklearn.preprocessing import StandardScaler
5  from sklearn.svm import LinearSVC
6  iris = datasets.load_iris()
7  X = iris["data"][:, (2, 3)]  # długość i szerokość płatka
8  y = (iris["target"] == 2).astype(np.int8)  # Iris virginica
9  svm_clf = Pipeline([
10         ("scaler", StandardScaler()),
11         ("linear_svc", LinearSVC(C=1,
12                                  loss="hinge",
13                                  random_state=42)),
14     ])
15 svm_clf.fit(X, y)
```

- automatyczne skalowanie
- funkcja kary: ujemna odległość od granicy

SVM, klasyfikacja

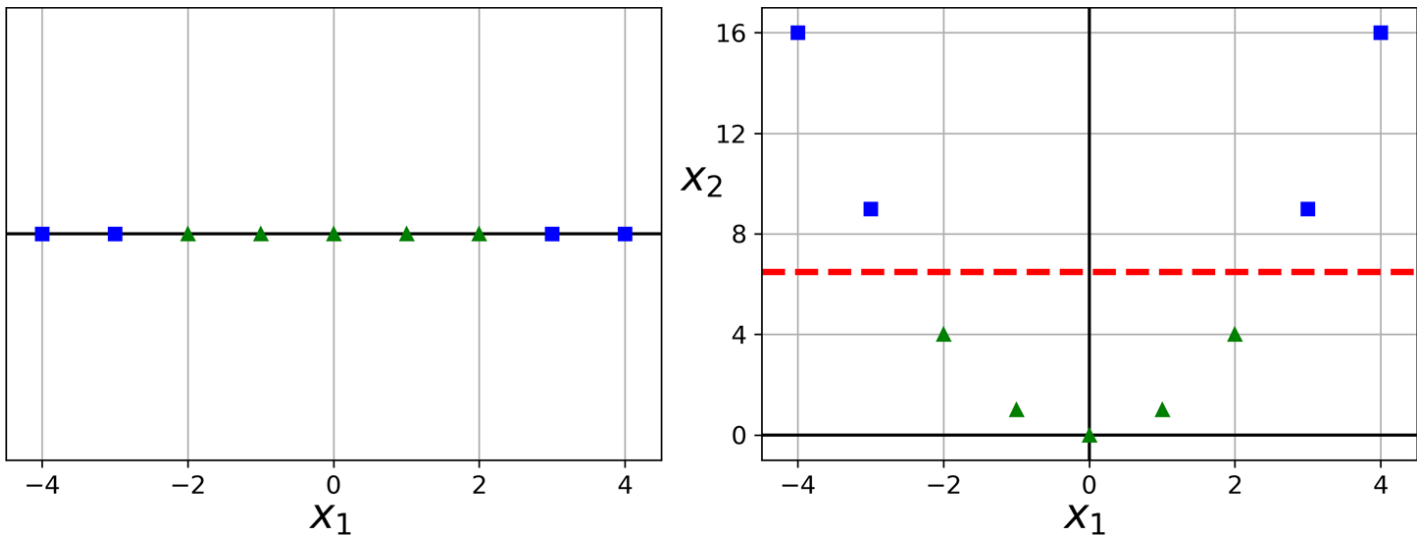
```
1  print(svm_clf.predict([[5.5, 1.7],
2                         [4.5, 1.7]]))
```

```
[1 0]
```

- kwiatek o długości płatka 5.5 cm i szerokości 1.7 cm to *Iris virginica*
- kwiatek o długości płatka 4.5 cm i szerokości 1.7 cm to nie jest *Iris virginica*
- brak informacji o prawdopodobieństwie, w porównaniu z Regresją Logistyczną.

SVM, klasyfikacja nieliniowa

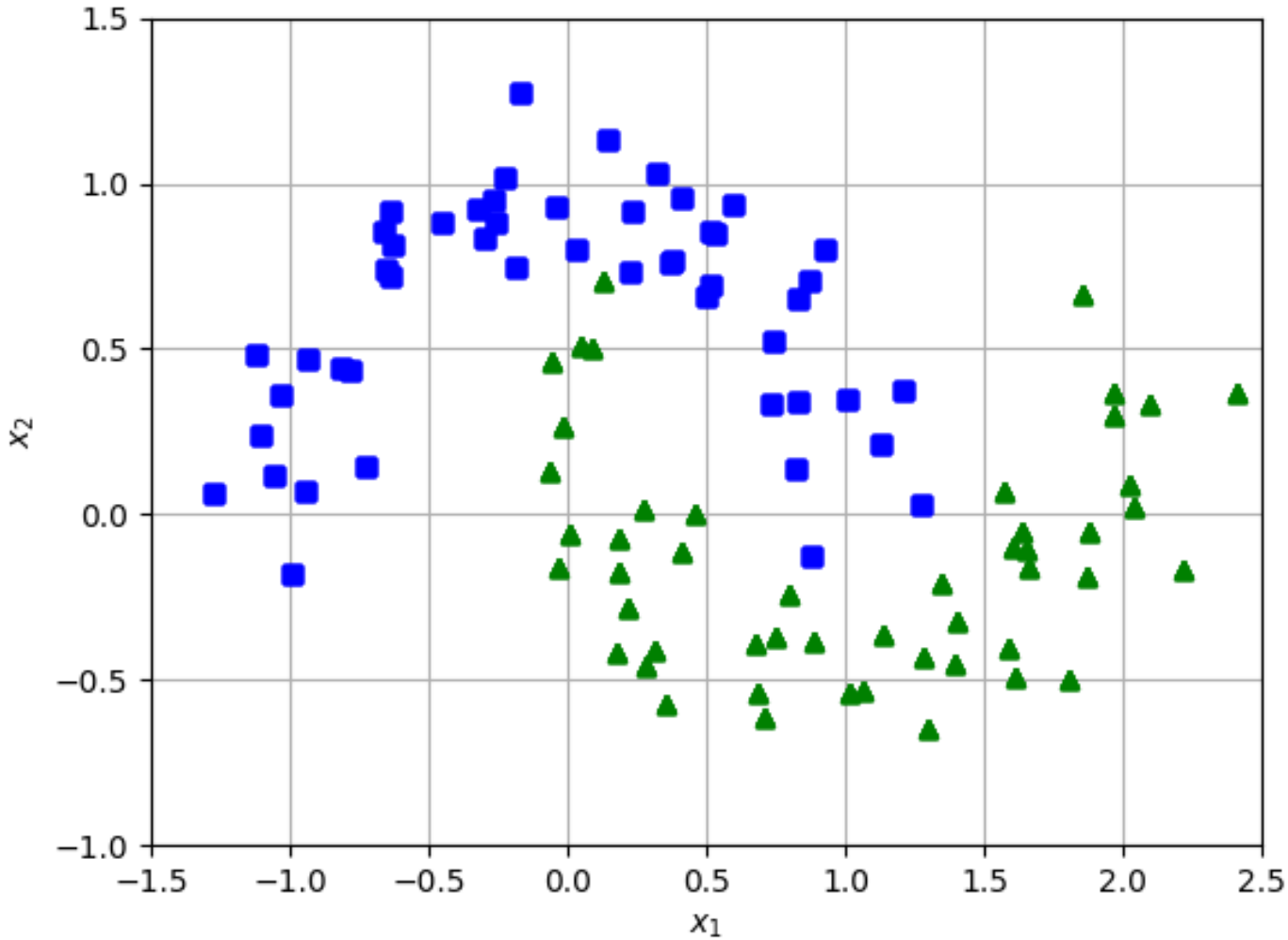
- Co jeżeli instancji nie można odseparować hiperpłaszczyzną?
- Dodać więcej cech.



SVM, nieliniowa, dane

```
1  from sklearn.datasets import make_moons
2  import matplotlib.pyplot as plt
3
4  X, y = make_moons(n_samples=100, noise=0.15,
5                    random_state=42)
6
7  def plot_dataset(X, y, axes, file):
8      plt.plot(X[:, 0][y==0], X[:, 1][y==0], "bs")
9      plt.plot(X[:, 0][y==1], X[:, 1][y==1], "g^")
10     plt.axis(axes)
11     plt.grid(True, which='both')
12     plt.xlabel("$x_1$")
13     plt.ylabel("$x_2$")
14     plt.savefig(f)
15
16 f = "moons_dataset.png"
17 plot_dataset(X, y, [-1.5, 2.5, -1, 1.5], f)
18 print(f)
```

Moon dataset



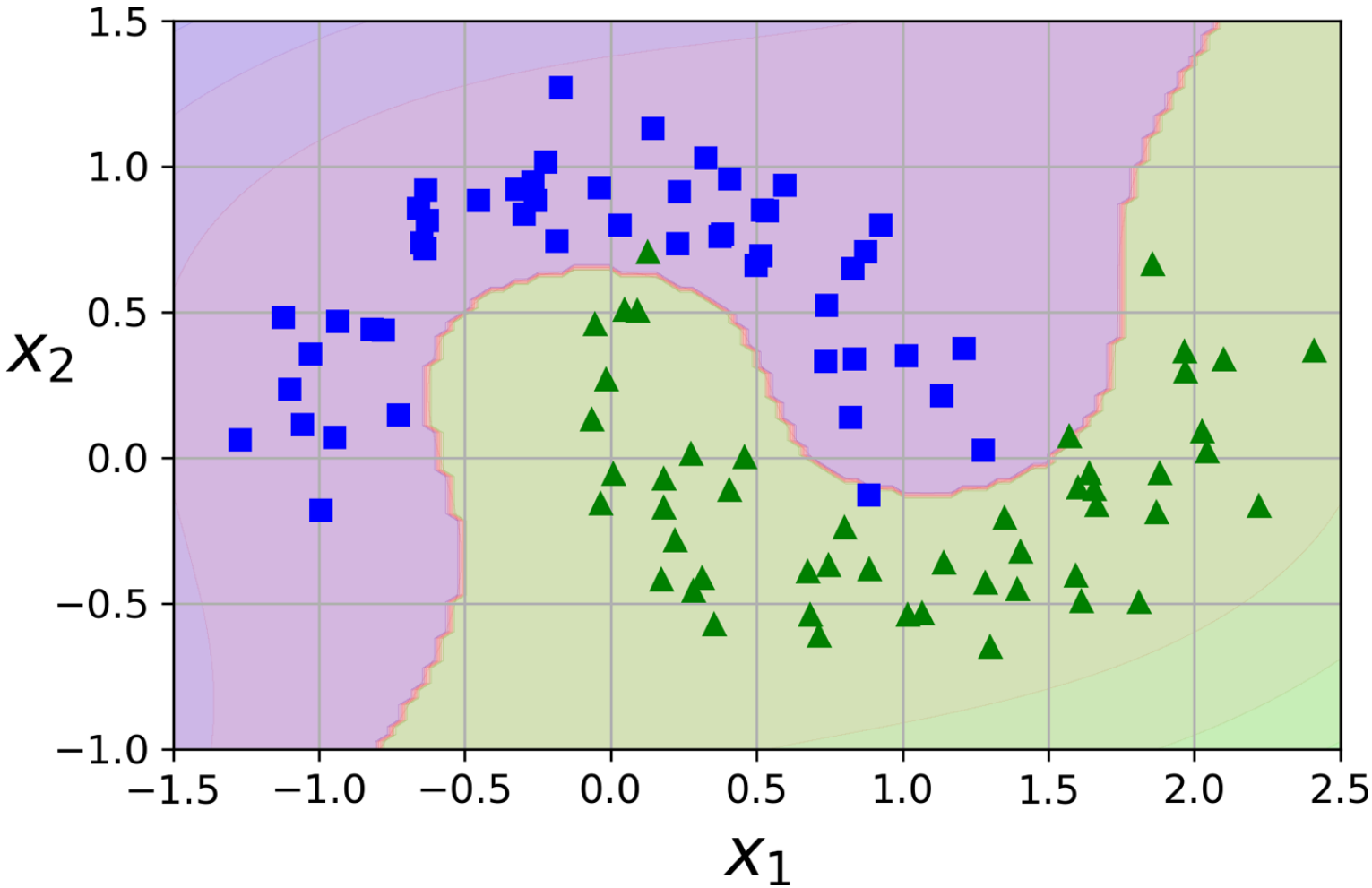
SVM, nielionowa, uczenie

```
1  from sklearn.pipeline import Pipeline
2  from sklearn.preprocessing import PolynomialFeatures
3
4  polynomial_svm_clf = Pipeline([
5          ("poly_features", PolynomialFeatures(degree=3,
6                                               include_bias=Fa
7          ("scaler", StandardScaler()),
8          ("svm_clf", LinearSVC(C=10, loss="hinge",
9                                max_iter=3000, # zbieżność
10                               random_state=42))
11     ])
12
13 polynomial_svm_clf.fit(X, y)
```

SVM, nielioniowa, klasyfikacja

```
1  print(polynomial_svm_clf.predict([[0.5, 0],
2                                    [1, 0]]))
```

```
[1 0]
```



SVM, nieliniowa klasyfikacja, raz jeszcze

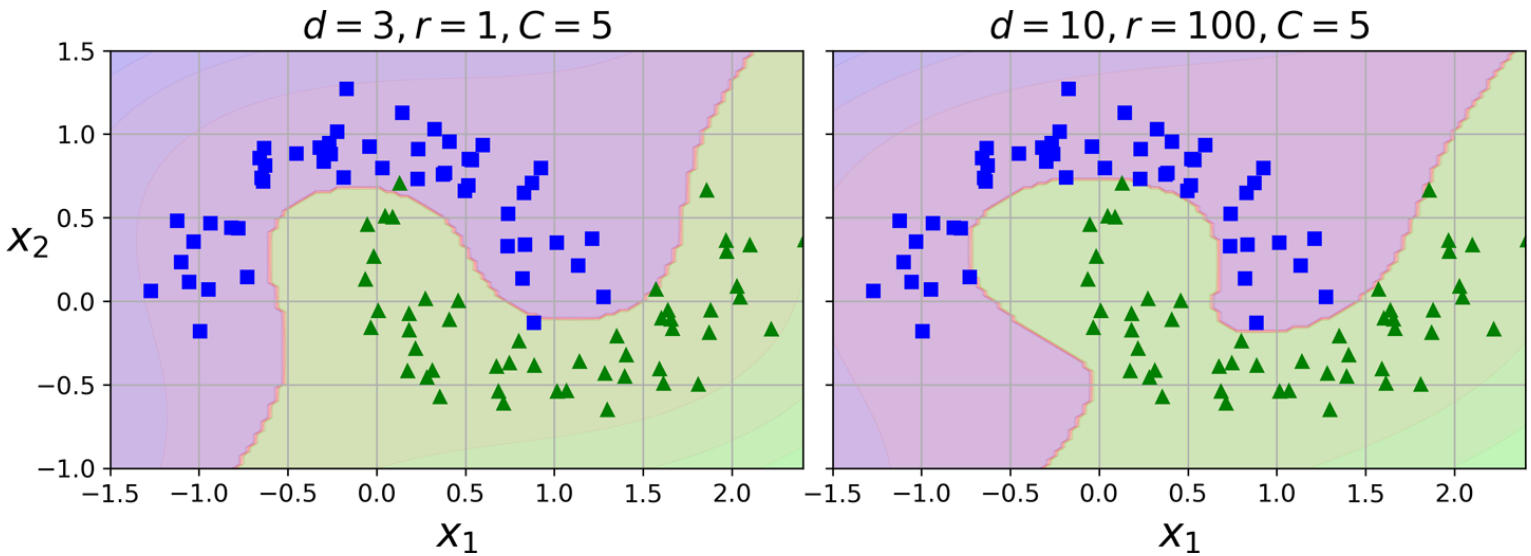
```
from sklearn.svm import SVC

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)
print(poly_kernel_svm_clf.predict([[0.5, 0],
                                   [1, 0]]))
```

```
[1 0]
```

- szybsze niż dodawanie cech
- są również inne *kernels*
- wielomian wyższego rzędu -> overfitting
- wielomian niższego rzędu -> underfitting
- hiperparametr *coef0* – im większy tym większ wpływ wielomianów wysokiego rzędu

SVM, nieliniowa klasyfikacja, raz jeszcze



Złożonośc obliczeniowa

- m – liczba instancji, n – liczba cech
- LinearSVC, $O(m * n)$,
- SVC, $O(m^2 * n)$ – $O(m^3 * n)$

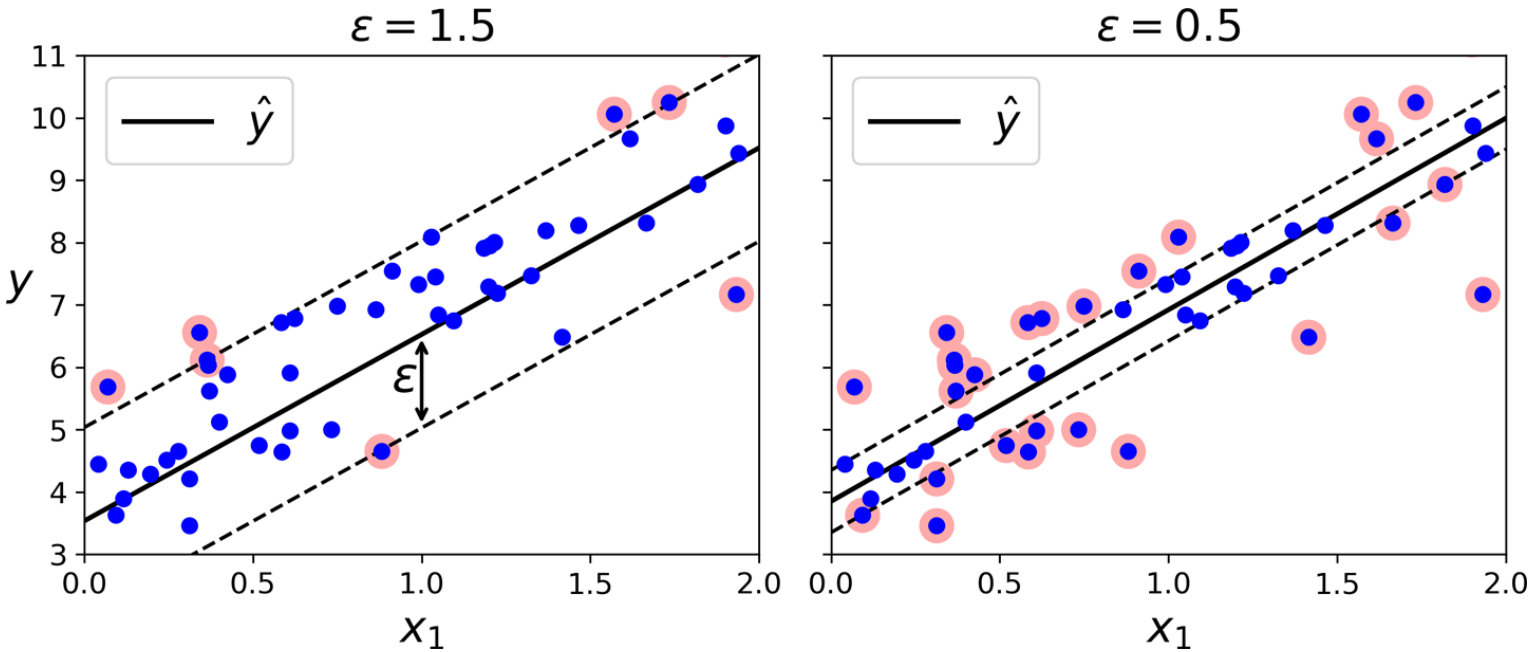
Plan prezentacji

Klasyfikacja

Regresja

SVM, regresja liniowa

- Dopasowanie jak największej liczby instancji w "ulicy" minimalizując naruszenie marginesów ϵ.



SVM, regresja liniowa, dane

```
np.random.seed(42)
m = 50
X = 2 * np.random.rand(m, 1)
y = (4 + 3 * X + np.random.randn(m, 1)).ravel()
```

SVM, regresja liniowa, uczenie

```
1  from sklearn.svm import LinearSVR
2
3  svm_reg = LinearSVR(epsilon=1.5, random_state=42)
4  svm_reg.fit(X, y)
```

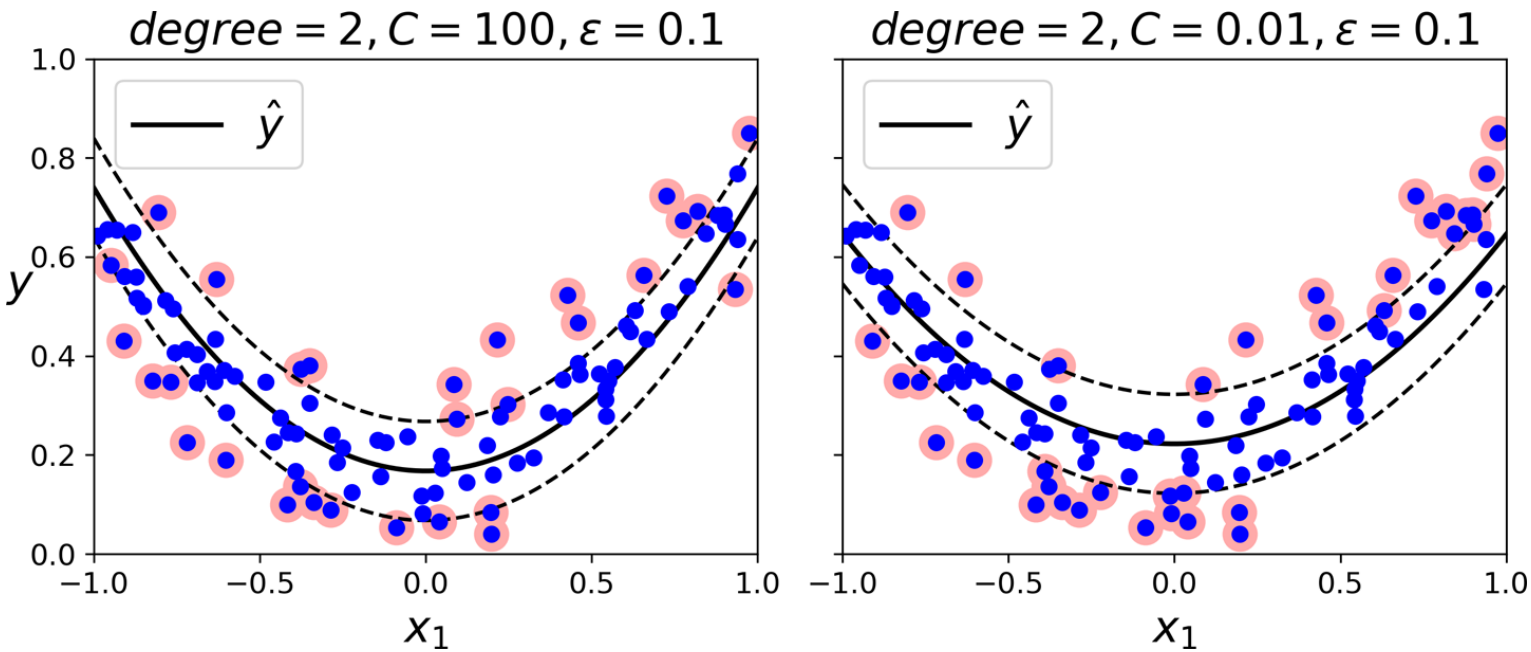
SVM, regresja liniowa, predykcja

```
1  print(svm_reg.predict([[1],[2]]))
```

```
[6.52640746 9.51919121]
```

SVM, regresja nielinowa, dane

```
np.random.seed(42)
m = 100
X = 2 * np.random.rand(m, 1) - 1
y = (0.2 + 0.1 * X + 0.5 * X**2 +
     np.random.randn(m, 1)/10).ravel()
```



SVM, regresja nielinowa, uczenie

```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2,
                   C=100, epsilon=0.1, gamma="scale")
svm_poly_reg.fit(X, y)
```

SVM, regresja nielinowa, predykcja

```
1  print(svm_poly_reg.predict([[0],[-1]]))
```

```
[0.16764293 0.73995101]
```