Linear models and Nearest Neighbors

Andreas Müller

Supervised Learning

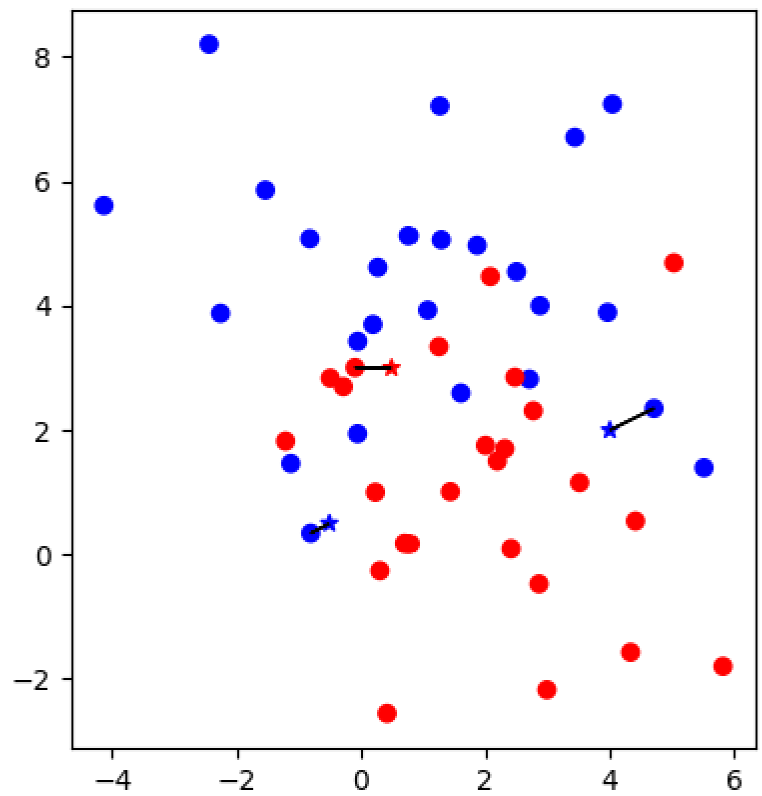
$$(x_i, y_i) \propto p(x, y)$$ i.i.d.

$$x_i \in \mathbb{R}^n$$

$$y_i \in \mathbb{R}$$

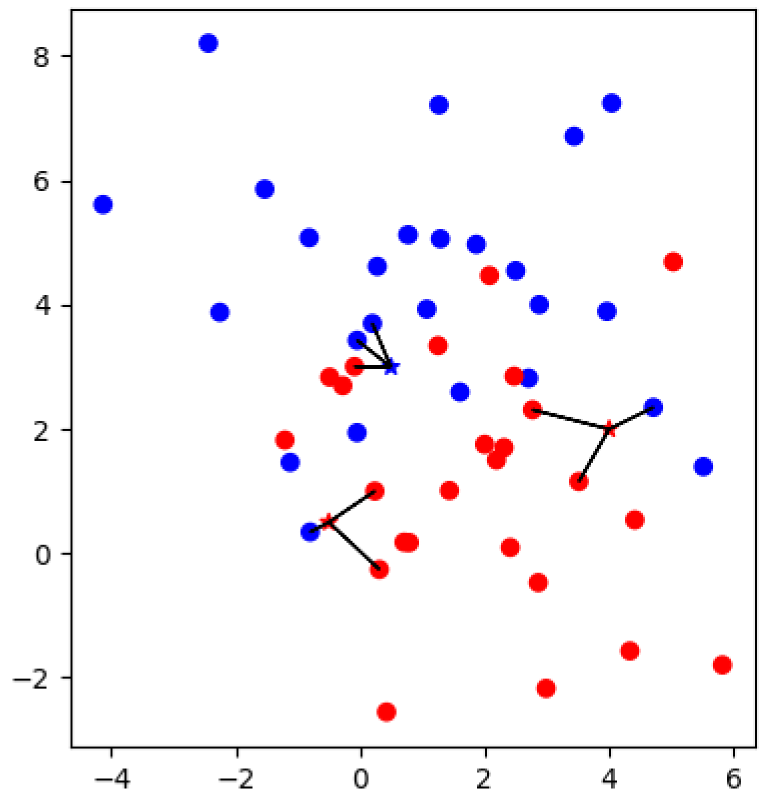
$$f(x_i) \approx y_i \qquad f(x) \approx y$$

Nearest neighbors

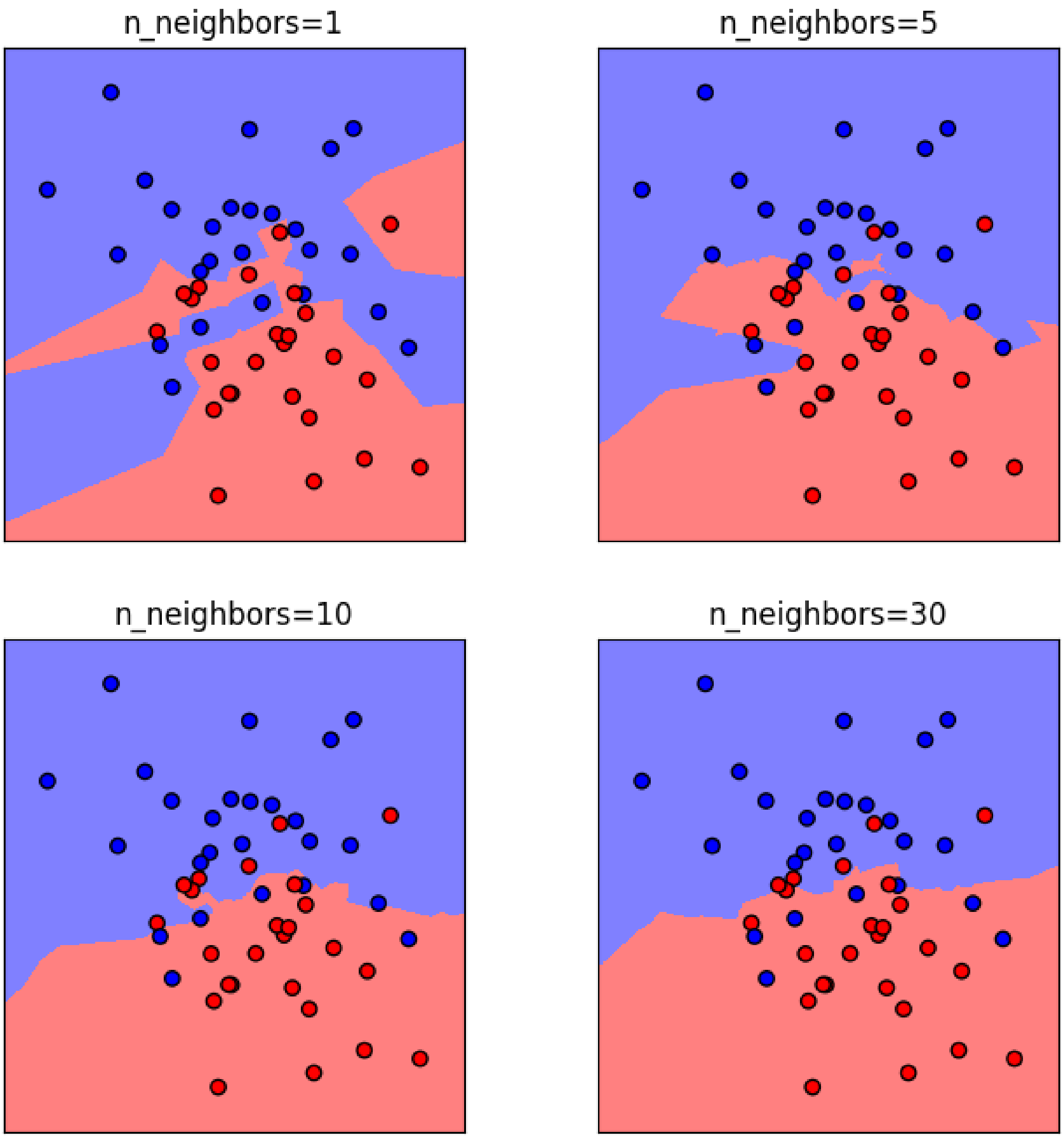


$$f(x) = y_i, i = \text{argmin}_j ||x_j - x||$$

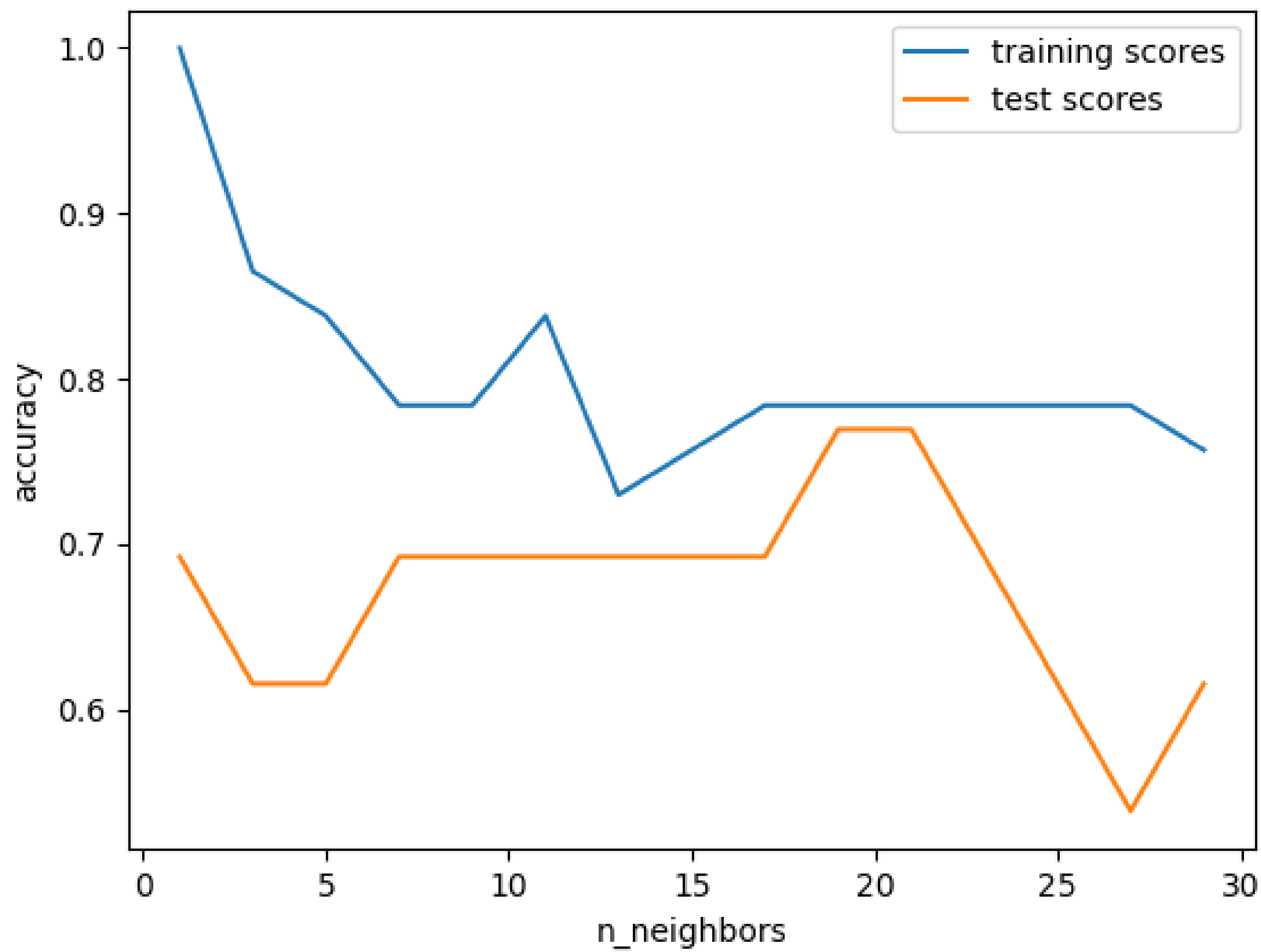
Nearest neighbors



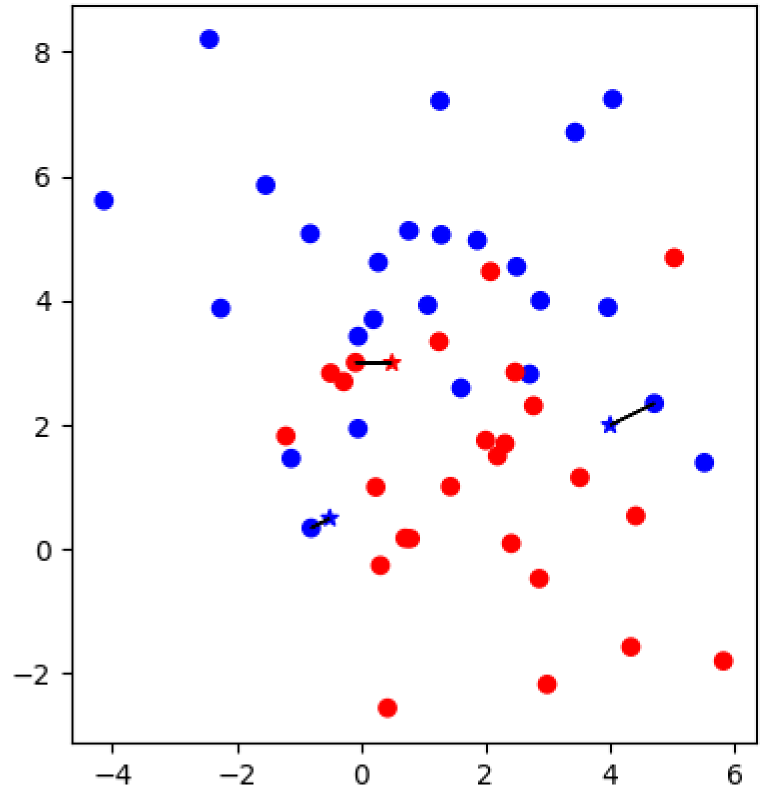
Influence of n_neighbors



Model Complexity



Nearest neighbors



$$f(x) = y_i, i = \mathrm{argmin}_j ||x_j - x||$$

training set

$$X = \begin{pmatrix} 1.1 & 2.2 \\ 6.7 & 0.5 \\ 2.4 & 9.3 \\ 1.5 & 0.0 \\ 0.5 & 3.5 \\ 5.1 & 9.7 \\ 3.7 & 7.8 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

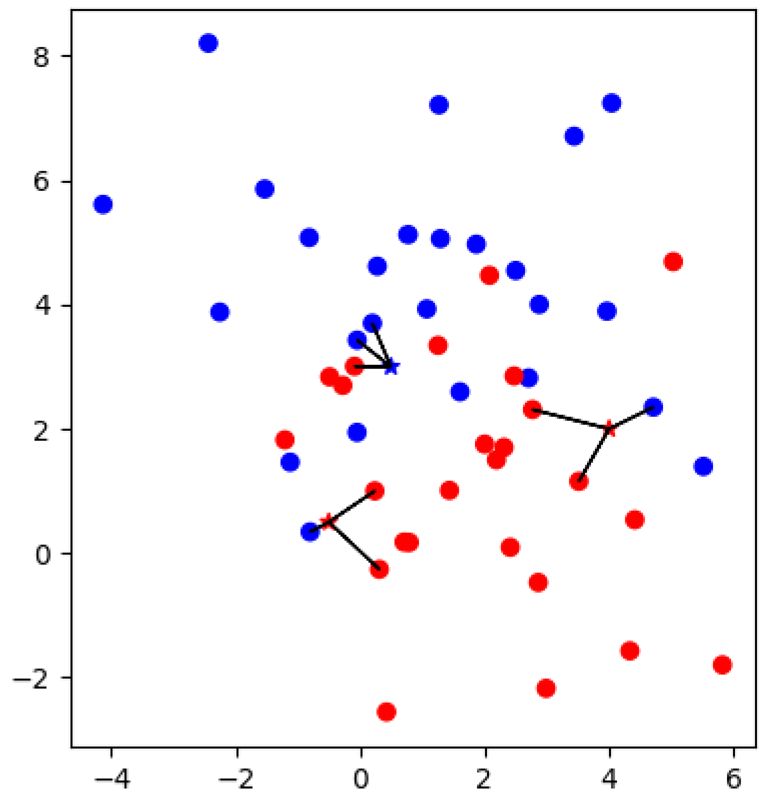
test set

Implementing KNN in scikit-learn

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("accuracy: {:.2f}".format(knn.score(X_test, y_test)))
```

```
accuracy: 0.77
```

Nearest neighbors



Overfitting and Underfitting

Training

Sweet spot

Accuracy

Generalization

Underfitting

Overfitting

Model complexity

Computational Properties Neighbors

- fit: no time
- memory: O(n * p)
- predict: O(n * p)

Kd-tree

- fit: O(p * n log n)
- memory: O(n * p)
- predict:

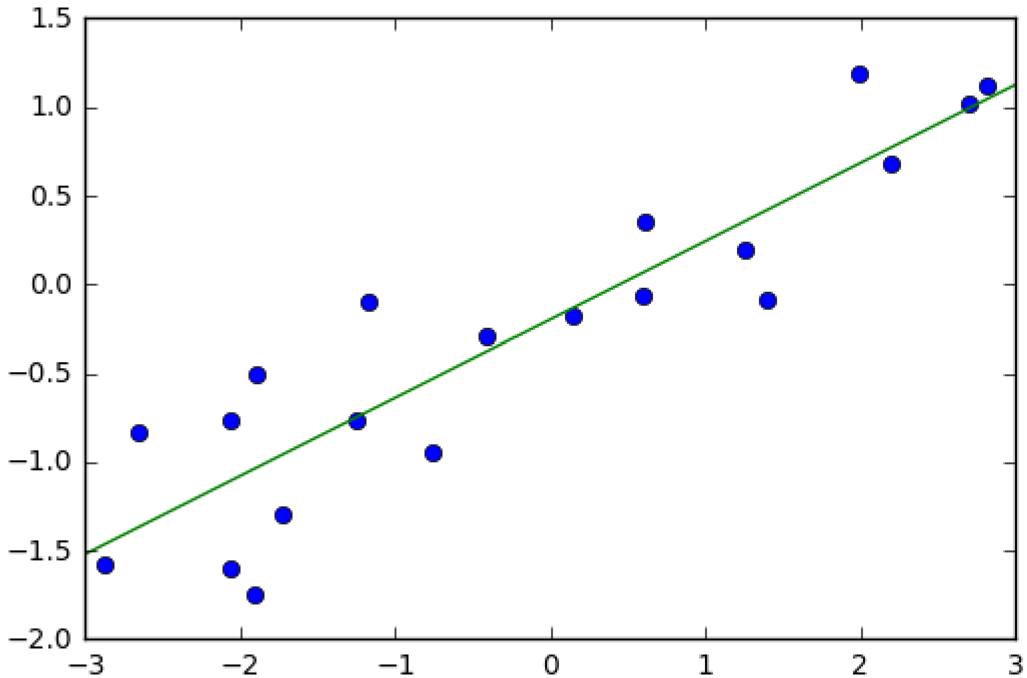
  O(k * log(n))

  FOR FIXED p!

n = n_samples
p = n_features

Linear models for regression

Linear Models for Regression



$$\hat{y} = w^T\mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

Linear Regression
Ordinary Least Squares

$$\hat{y} = w^T \mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{p} ||w^T \mathbf{x}_i - y_i||^2$$

Unique solution if $\mathbf{X} = (\mathbf{x}_1, ...\mathbf{x}_n)^T$ has full rank.

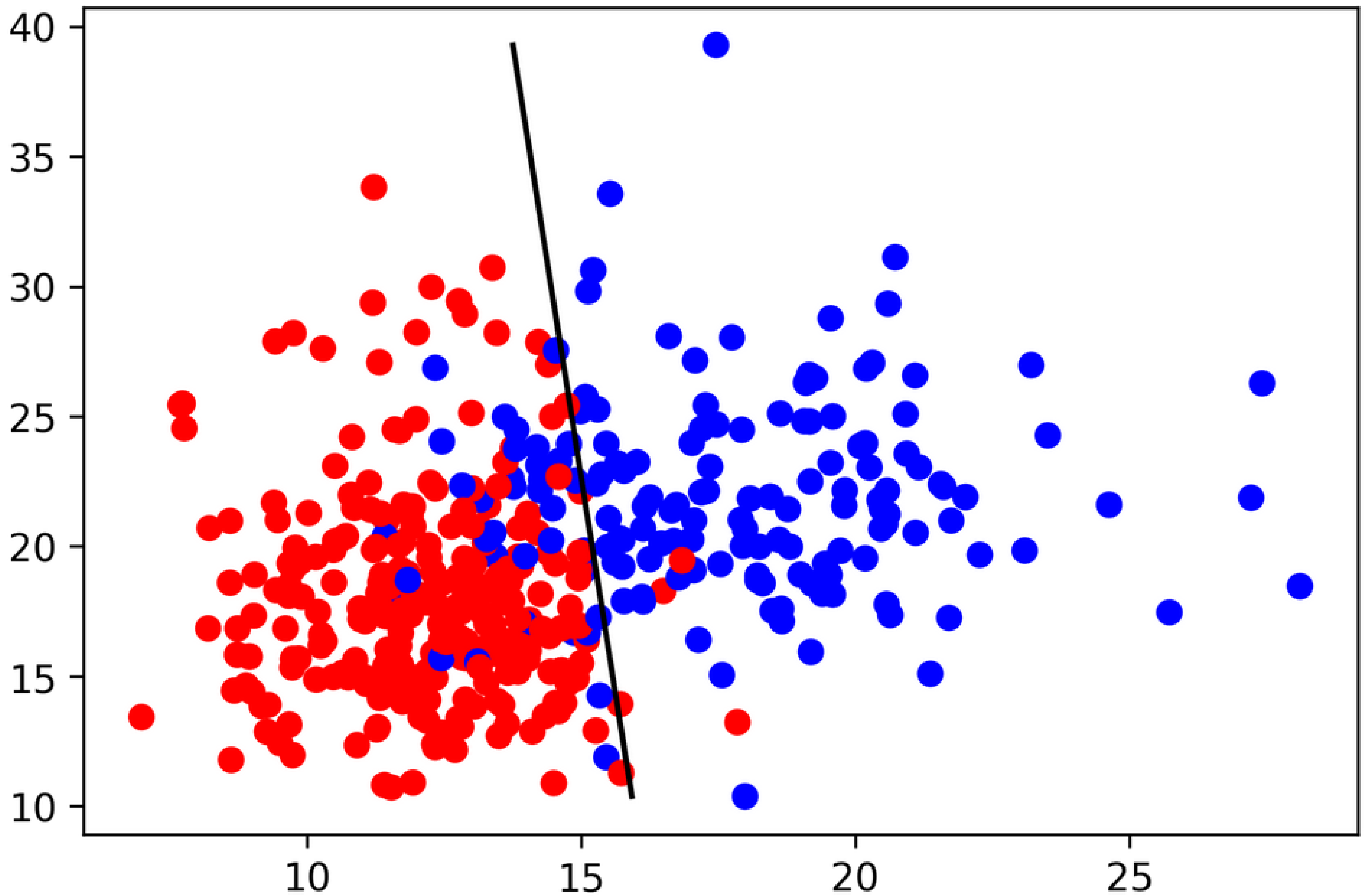
Ridge Regression

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} ||w^T x_i - y_i||^2 + \alpha ||w||^2$$

Always has a unique solution.
Has tuning parameter alpha

Linear Models for Classification

Linear models for **binary** classfiication

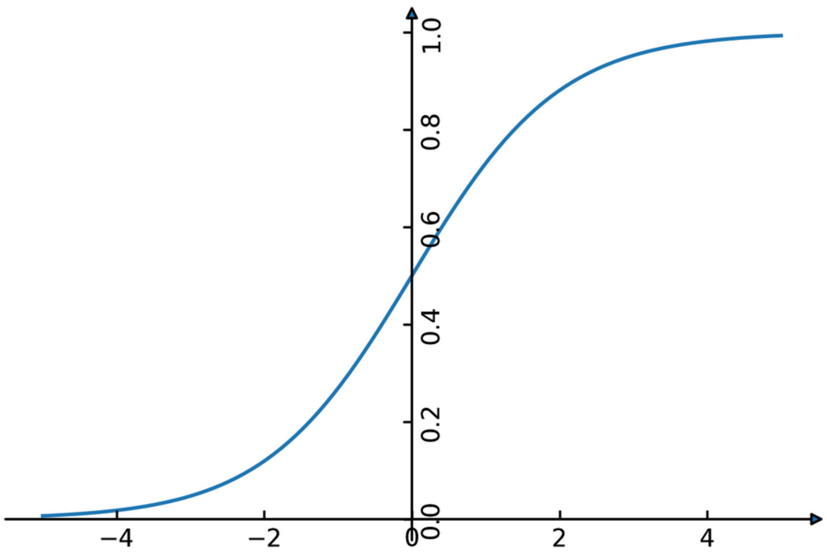


$$\hat{y} = \mathrm{sign}(w^T\mathbf{x} + b) = \mathrm{sign}(\sum_i w_i x_i + b)$$

Logistic Regression

$$\min_{w \in \mathbb{R}^p} -\sum_{i=1}^{n} \log(\exp(-y_i w^T \mathbf{x}_i) + 1)$$

$$p(y|\mathbf{x}) = \frac{1}{1 + e^{-w^T \mathbf{x}}}$$

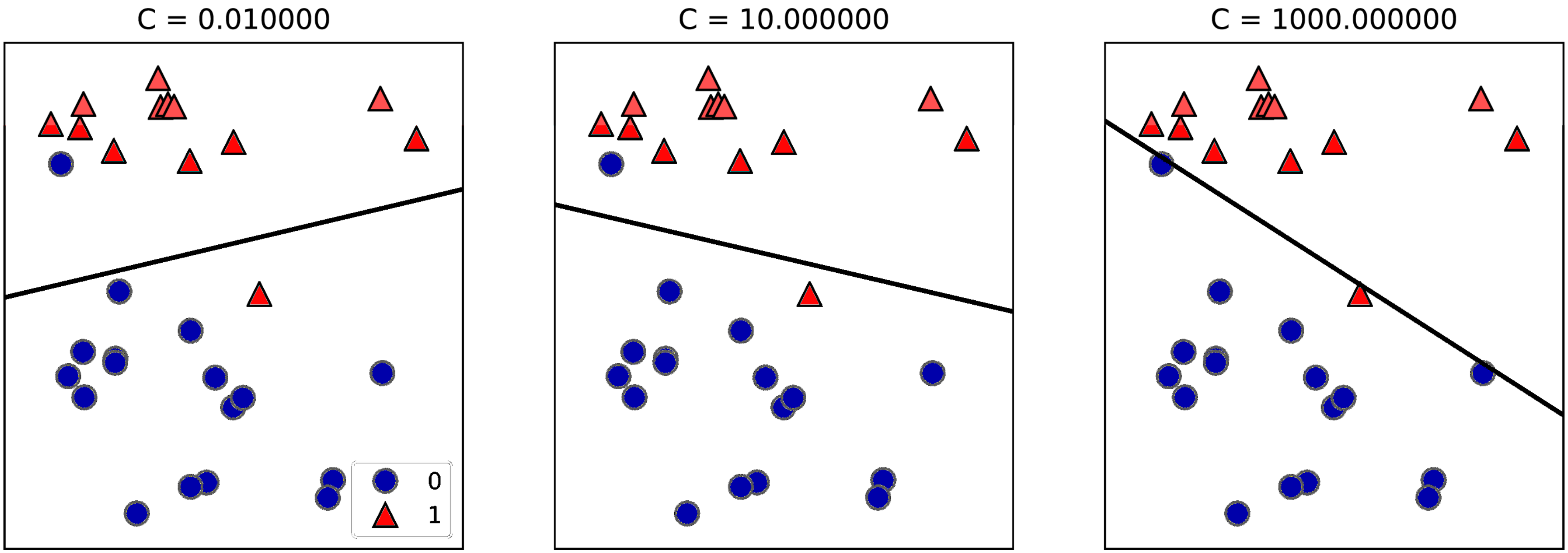


$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

Penalized Logistic Regression

$$\min_{w \in \mathbb{R}^p} -C \sum_{i=1}^{n} \log(\exp(-y_i w^T \mathbf{x}_i) + 1) + ||w||_2^2$$

C is inverse to alpha (or alpha / n_samples)


C = 0.010000
C = 10.000000
C = 1000.000000
0
1

Multiclass classification

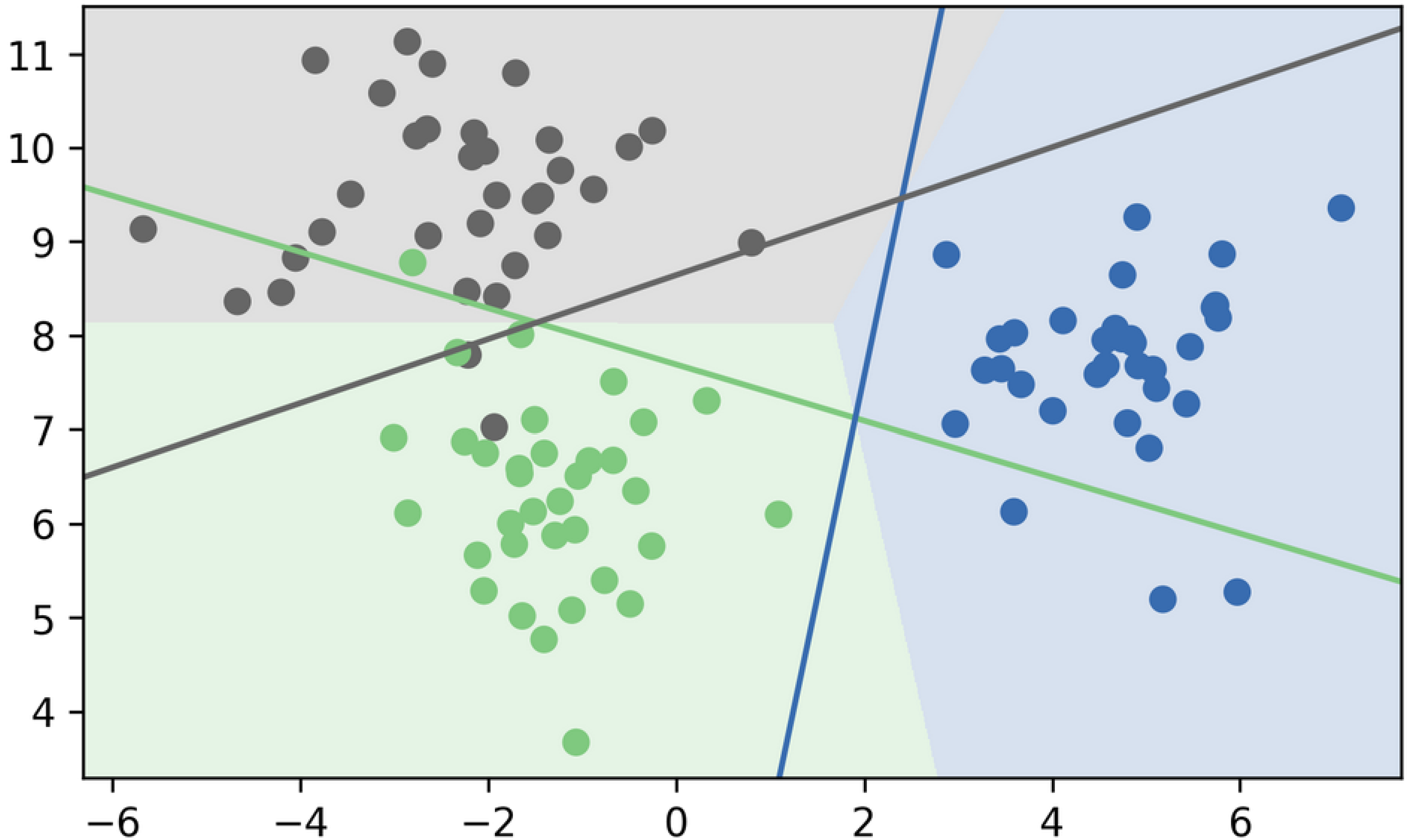
Multinomial Logistic Regression

Probabilistic multi-class model:

$$p(y = i|\mathbf{x}) = \frac{e^{-\mathbf{w}_i^T\mathbf{x}}}{\sum_{j\in Y} e^{-\mathbf{w}_j^T\mathbf{x}}}$$

$$\min_{w\in\mathbb{R}^p} -\sum_{i=1}^{n} \log(p(y = y_i|\mathbf{x}_i))$$

$$\hat{y} = \arg\max_{i\in Y} \mathbf{w}_i\mathbf{x}$$



$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i \mathbf{x}$$

Multi-Class in Practice

OvR and multinomial LogReg produce one coef per class:

```
from sklearn.datasets import load_iris
iris = load_iris()
X, y = iris.data, iris.target
print(X.shape)
print(np.bincount(y))
```

```
(150, 4)
[50 50 50]
```

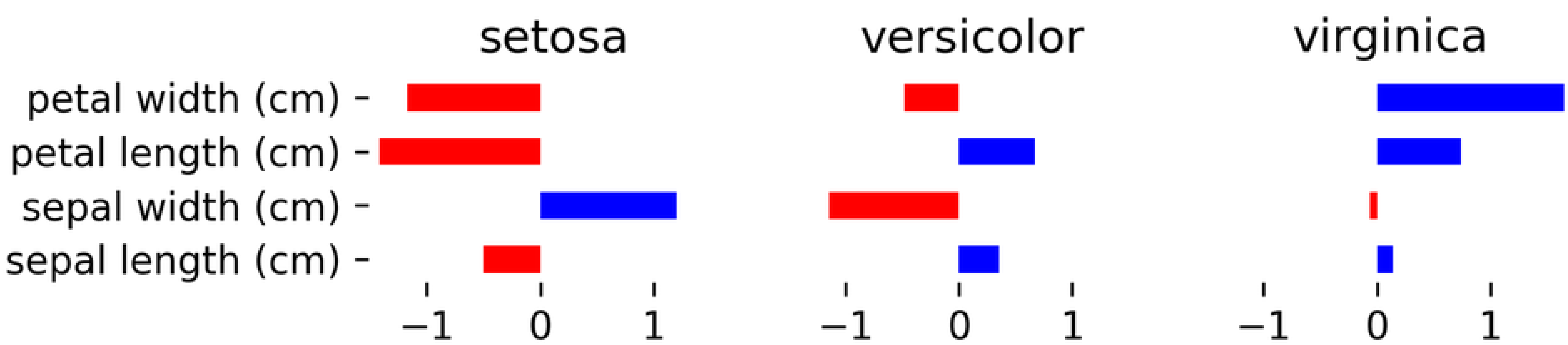
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

logreg = LogisticRegression(multi_class="multinomial", solver="lbfgs").fit(X, y)
linearsvm = LinearSVC().fit(X, y)
print(logreg.coef_.shape)
print(linearsvm.coef_.shape)
```

```
(3, 4)
(3, 4)
```

SVC would produce the same shape, but with different semantics!

```
logreg.coef_
```

```
array([[-0.42339232,  0.96169329, -2.51946669, -1.0860205 ],
       [ 0.53411332, -0.31794321, -0.20537377, -0.93961515],
       [-0.11072101, -0.64375008,  2.72484045,  2.02563566]])
```



(after centering data, without intercept)