Scikit-learn and Tour of Classifiers

Recap

Classification pipeline

- **Goal**: Train a classifier that **generalizes** well on **unseen** data
- Train (for training) / Test (unseen) split to get better estimate of generalization error via error on the test set
  - **Randomize and stratify** to get equal class distribution in test & train set
- Feature Scaling/Transformations
  - Always apply **identical** transformations to test & train set
  - **Avoid information leakage!** Standardization: compute mean/stddev **on the training set**
- **Accuracy:** 1.0 – misclassified_samples / all_samples

Scikit-learn and Tour of Classifiers

Overfitting

Train a logistic regression model – Parameter C

`03_logreg_iris.ipynb`

Code example:

- Use **Iris** data set in scikit-learn
- Use ALL features
- Split the data into **training** and **test** set (test_size=0.3, random_state=1)
- Intialise LogisticRegression class with LogisticRegression(C=100.0, random_state=1)
- Print out number of **misclassified samples**
- Print out **classification accuracy** for **training data** & **test data**
- **Does accuracy change with C?**

Train logistic regression model on cancer data set

Exercise:

- Use **Wisconsin breast cancer** data set in scikit-learn
- Use ALL features
- Split the data into **training** and **test** set (test_size=0.3, random_state=1)
- Intialise LogisticRegression class with LogisticRegression(C=100.0, random_state=1)
- Print out number of **misclassified samples**
- Print out **classification accuracy** for **training data** & **test data**
- **Does accuracy change with C?**

Overfitting / Underfitting

Overfitting

- Overfitting is a common problem in machine learning
- We want the model to **generalize** well on **unseen** data
- A model that is overfitted performs well on training data but does not generalise well to unseen data (test data)

Overfitting – Underfitting

A model that suffers from **overfitting**

- Has high variance
- High variance may be caused by too many parameters that lead to a model that is too complex given the training data

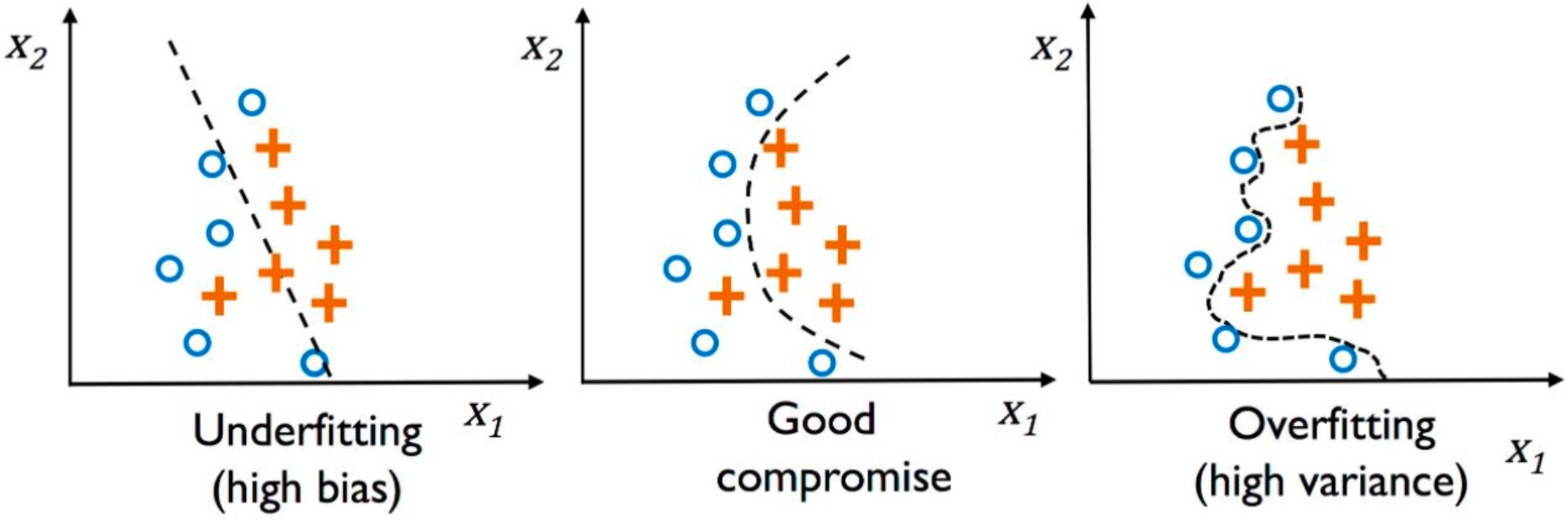
→ **poor performance** on unseen data because of high variance

A model that suffers from **underfitting**

- Has high bias
- High bias means that the model is not complex enough to capture the pattern in the training data well

→ **poor performance** on unseen data because of high bias

Overfitting – Underfitting (Examples)



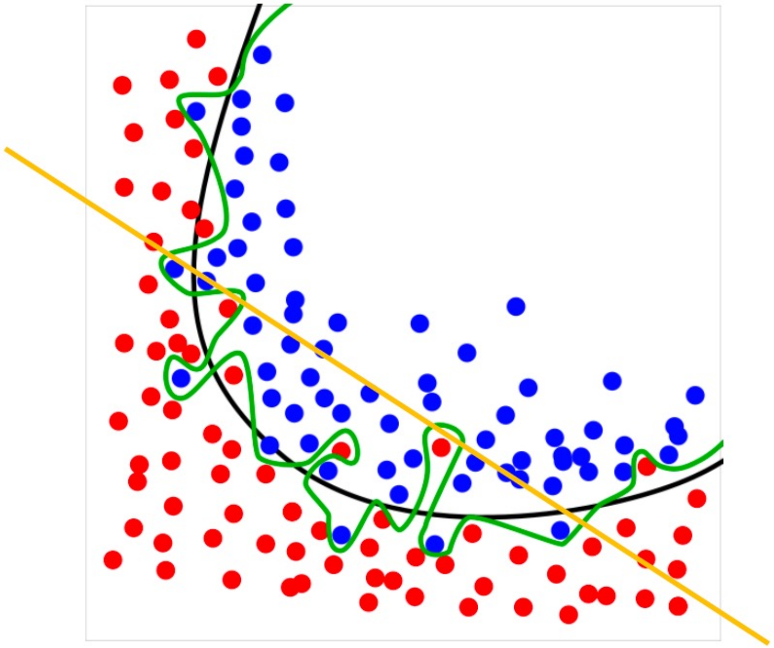
Linear decision boundary

Nonlinear decision boundary

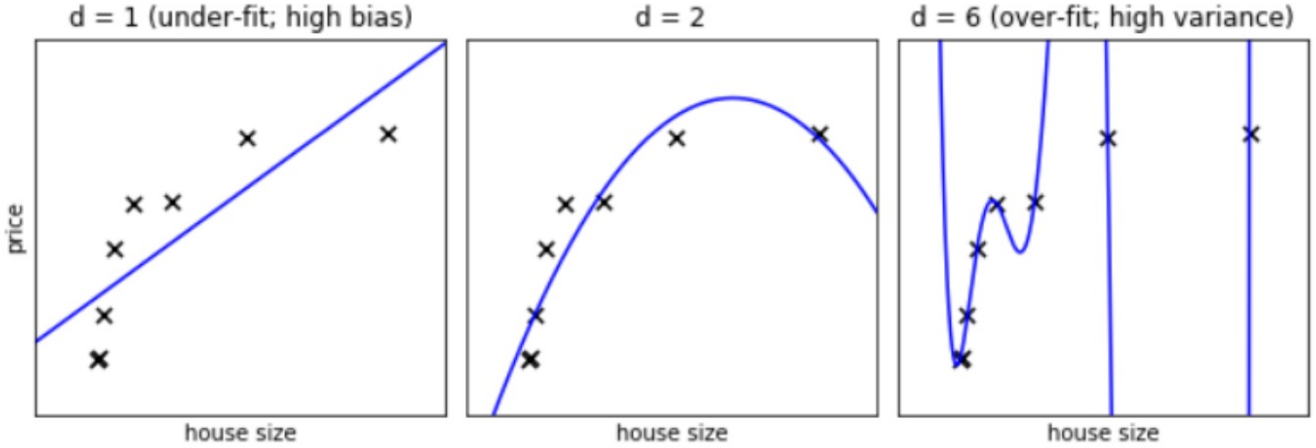
Nonlinear decision boundary

Figure from Python Machine Learning (Raschka & Mirjalili)

Overfitting – Underfitting (Examples)



Classification

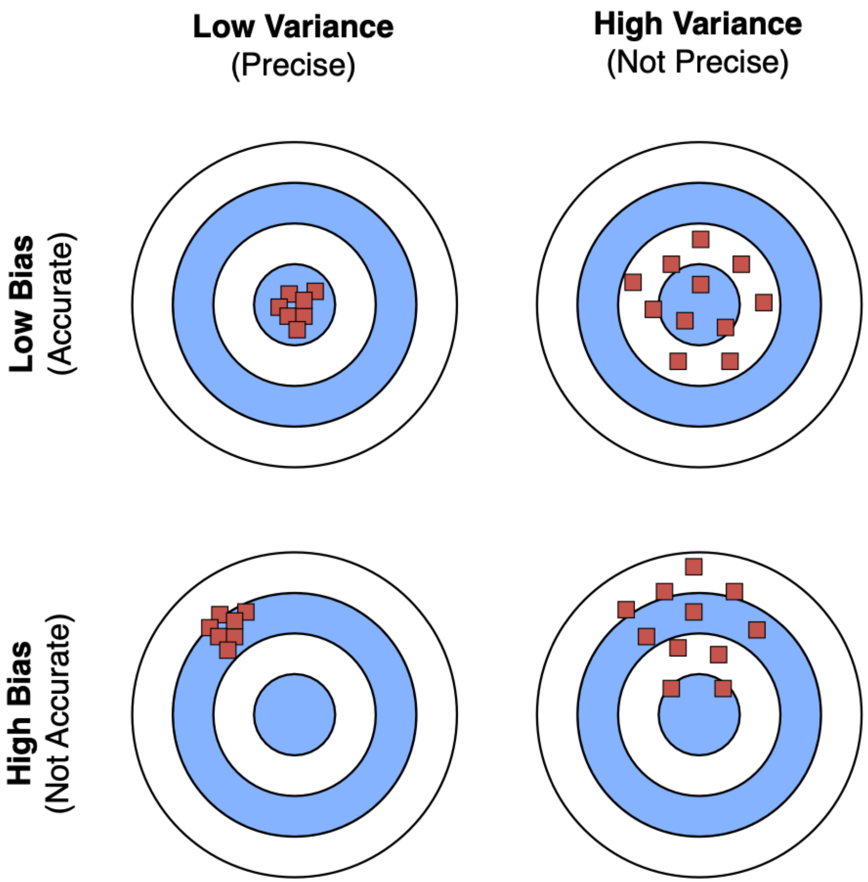


Regression

Figure from Python Machine Learning (Raschka & Mirjalili)

Bias – Variance and Overfitting – Underfitting

Overfitting



Underfitting

Figure by Sebastian Raschka Lecture Notes STAT451 FS20

Bias – Variance

Variance

- Measures **consistency** (**variability**) of the model **prediction** for a particular sample
- If model is retrained on different subsets of training data and prediction for a particular sample differ much each time → **high variance**
- The model is **sensitive** to the **randomness** in the **training data**

Bias

- Measures **how far off** the predictions are from the **correct values** in general
- If model is retrained multiple times on different **training** datasets → bias measures the **systematic error** that is **not due** to randomness

Bias – Variance

Given a true value $\boldsymbol{y}$ and an estimator $\hat{\boldsymbol{y}}$

Bias

$$Bias(\hat{\boldsymbol{y}}) := E[\hat{\boldsymbol{y}}] - \boldsymbol{y}$$

Variance

$$Var(\hat{\boldsymbol{y}}) := E[(E[\hat{\boldsymbol{y}}] - \hat{\boldsymbol{y}})^2] = E[\hat{\boldsymbol{y}}^2] - E[\hat{\boldsymbol{y}}]^2$$

$E[\hat{\boldsymbol{y}}]$ **:** the expected value (expectation) of the estimator $\hat{y}$ (here: expectation over training sets)

Bias – Variance decomposition of squared loss

Given a true value $\boldsymbol{y}$ and an estimator $\widehat{\boldsymbol{y}}$; $Bias(\widehat{\boldsymbol{y}}) = E[\widehat{\boldsymbol{y}}] - \boldsymbol{y}$; $Var(\widehat{\boldsymbol{y}}) = E[(E[\widehat{\boldsymbol{y}}] - \widehat{\boldsymbol{y}})^2]$

For $\boldsymbol{y}$ (true outcome) and $\widehat{\boldsymbol{y}}$ (estimated outcome):

using: $(a+b)^2 = a^2 + b^2 + 2ab$

Squared loss: $L = (\boldsymbol{y} - \widehat{\boldsymbol{y}})^2 = (\boldsymbol{y} - E[\widehat{\boldsymbol{y}}] + E[\widehat{\boldsymbol{y}}] - \widehat{\boldsymbol{y}})^2$

$$= (\boldsymbol{y} - E[\widehat{\boldsymbol{y}}])^2 + (E[\widehat{\boldsymbol{y}}] - \widehat{\boldsymbol{y}})^2 + 2(\boldsymbol{y} - E[\widehat{\boldsymbol{y}}])(E[\widehat{\boldsymbol{y}}] - \widehat{\boldsymbol{y}})$$

Take expected value of both sides:

$$\boxed{E[L] = E[(\boldsymbol{y} - E[\widehat{\boldsymbol{y}}])^2 + (E[\widehat{\boldsymbol{y}}] - \widehat{\boldsymbol{y}})^2] = Bias^2 + Variance}$$

using: $E[2(\boldsymbol{y} - E[\widehat{\boldsymbol{y}}])] = 2(\boldsymbol{y} - E[\widehat{\boldsymbol{y}}])$ and $E[(E[\widehat{\boldsymbol{y}}] - \widehat{\boldsymbol{y}})] = (E[\widehat{\boldsymbol{y}}] - E[\widehat{\boldsymbol{y}}]) = 0$

Bias – Variance trade-off

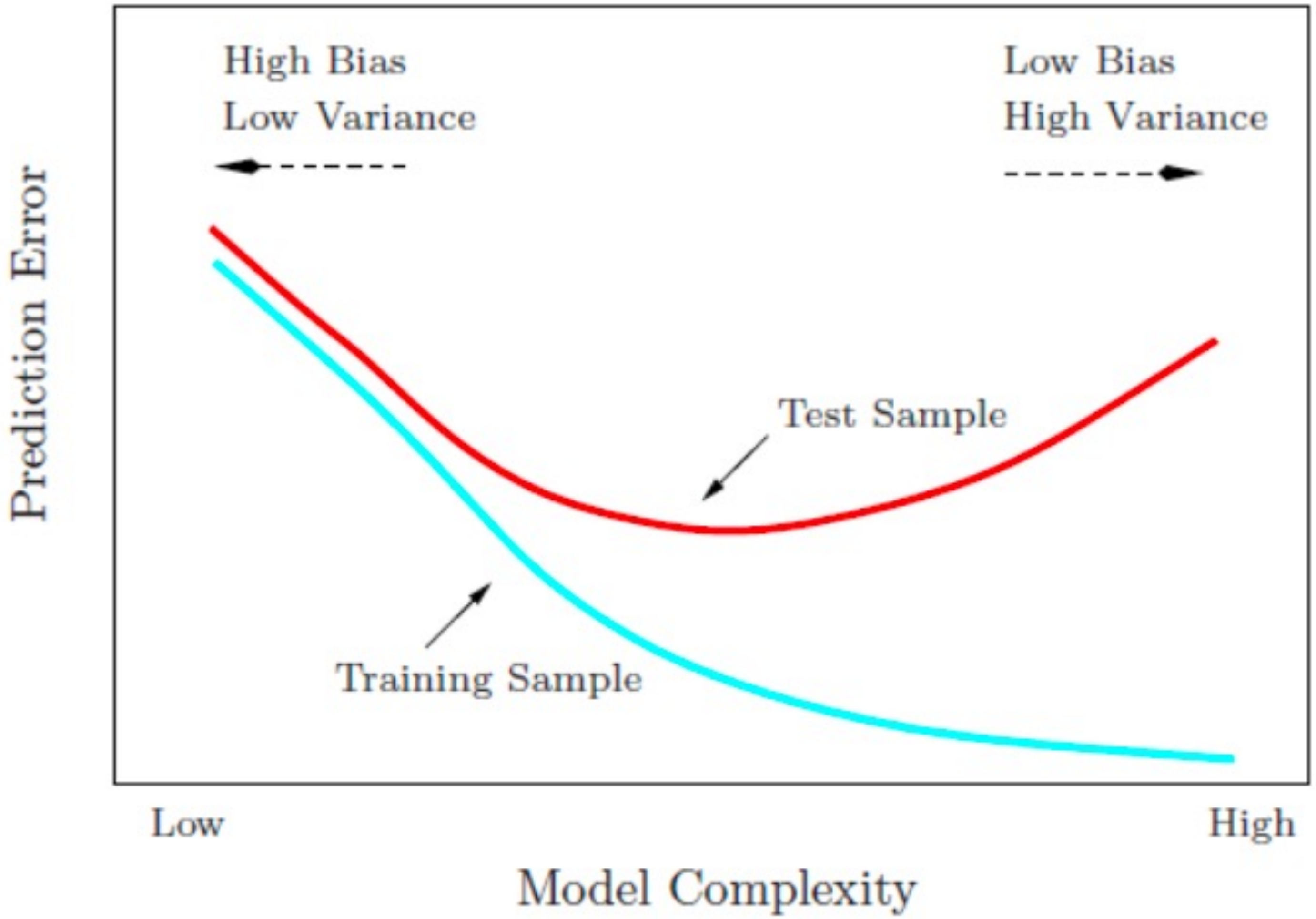


Figure from Elements of Statistical Learning (Hastie, Tibshirani, Friedman)

Bias – Variance trade-off

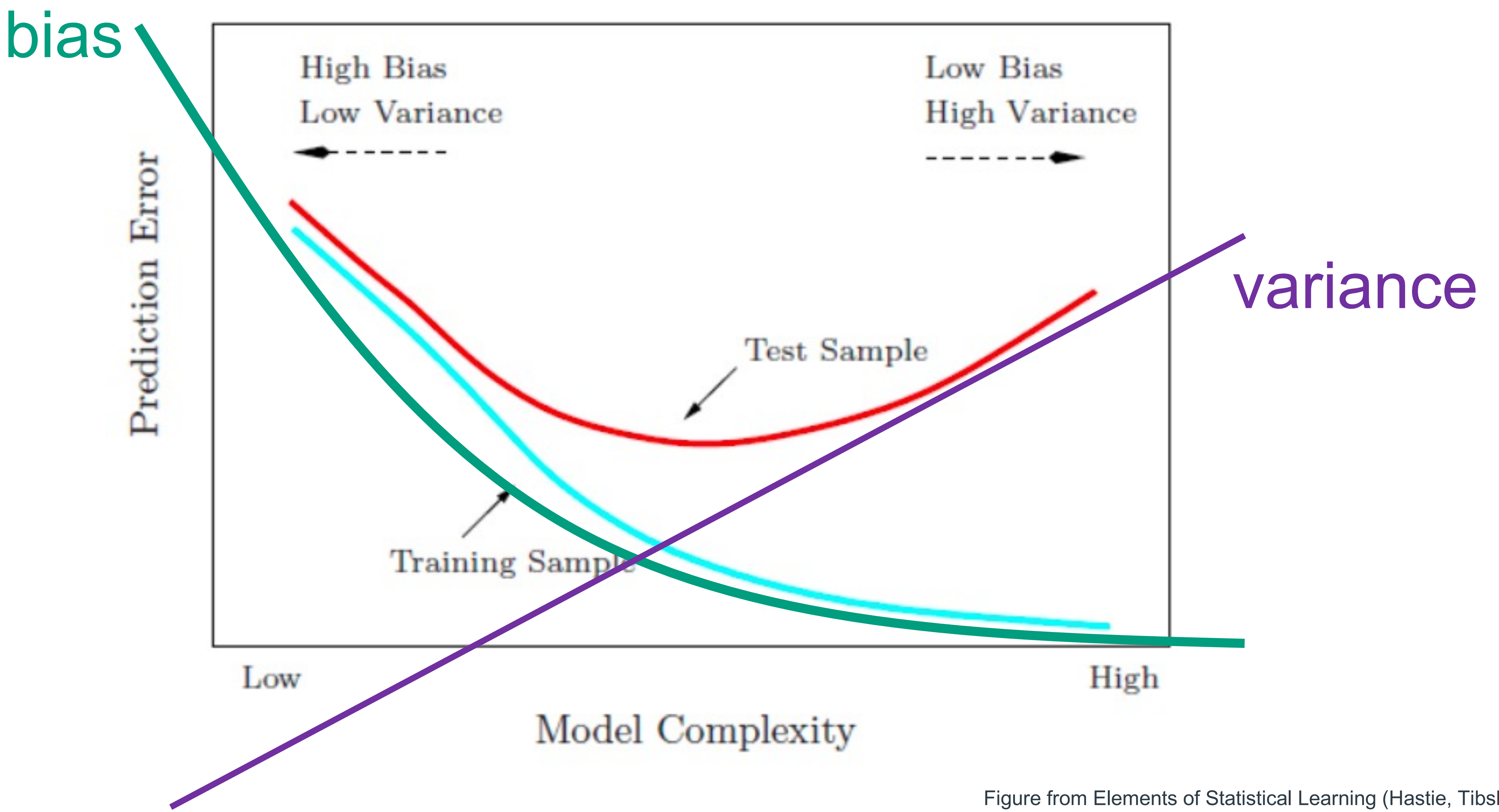


Figure from Elements of Statistical Learning (Hastie, Tibshirani, Friedman)

Bias – Variance trade-off

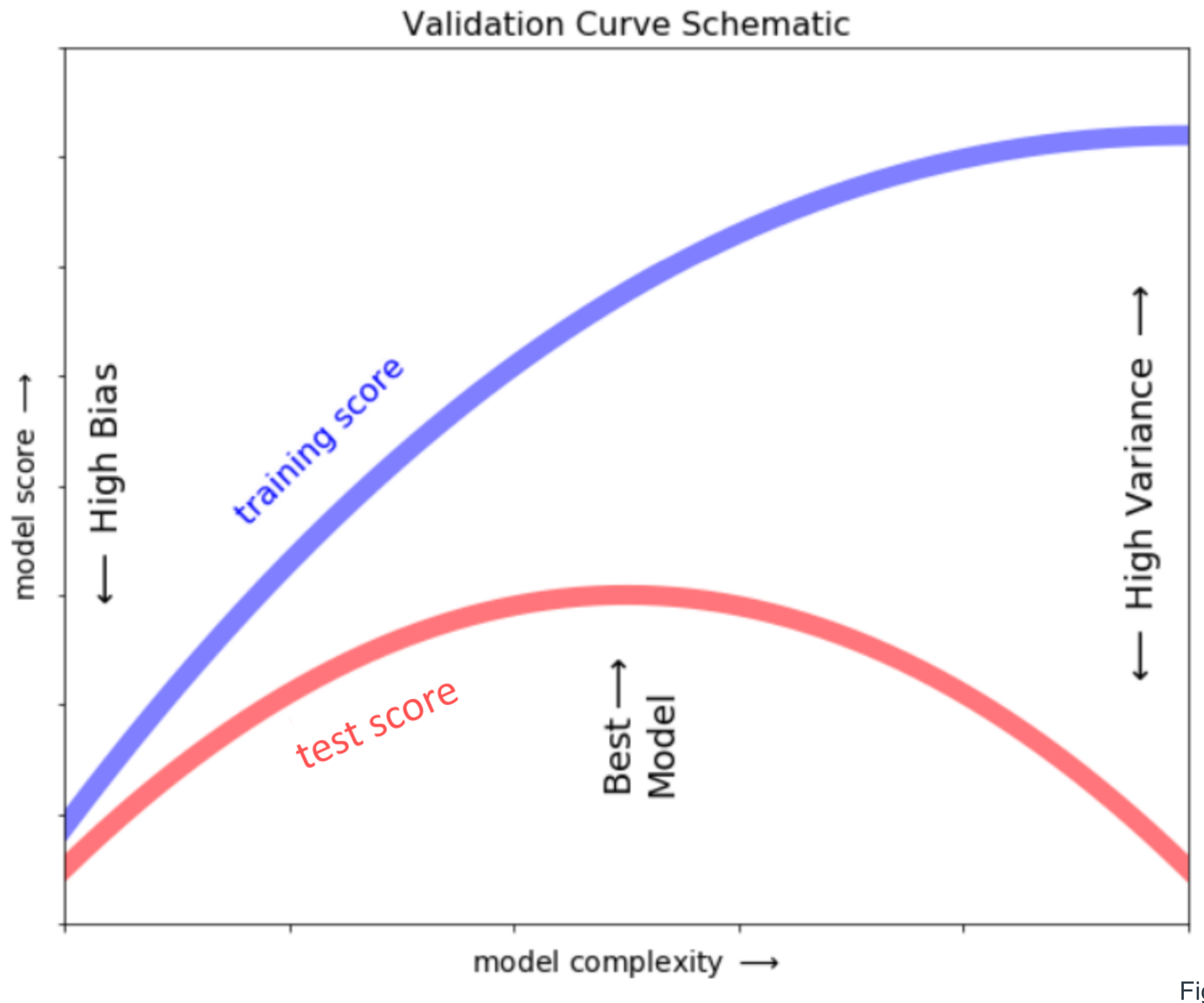


Figure from Python Machine Learning (Raschka & Mirjalili)

Tackling overfitting via regularization

Tackling overfitting

- Finding the appropriate **complexity** – **bias-variance trade-off** important for good performance

  - Collect **more training data**
  - Introduce a **penalty for complexity** via **regularization**
  - Choose a **simpler model** with fewer parameters
  - **Reduce** the **dimensionality** of the data (feature selection, dimensionality reduction, Ch.06)

Tackling overfitting via regularization

- Finding the appropriate **complexity** – **bias-variance trade-off** important for good performance
- Good option: **Tune** complexity of model using **regularisation**
- Regularisation is very useful for
  - Handling **collinearity** (high **correlation** among **features**)
  - **Filter out noise** from data
  - **Preventing overfitting**
- To make regularisation work properly, features need to be **scaled** / **standardised**

- **Concept** of regularisation
  - Introduce **additional information** (**bias**) to penalise **extreme parameter** (weight) **values**
  - **Most common** form of regularisation is so-called **L2 regularisation** (sometimes also called **L2 shrinkage** or **weight decay**) (details see Ch.04 in book)

ℓ_2- regularization (or L2 regularization / Ridge)

- ℓ_2-regularization: Add Euclidean norm (two-norm) of weights to the loss function
- **ℓ_2-regularized logistic regression:**

$$\boxed{L(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log(\sigma(z)) + (1-y^{(i)})\log(1-\sigma(z))\right] + \frac{\lambda}{2}||\mathbf{w}||_2^2}$$

recall that $||\mathbf{w}||_2^2 = \sum_{i=1}^{m} w_i^2$ and $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} := \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}$

and $z := b + \mathbf{x}^{(i)}\mathbf{w} = \mathbf{x}^{(i)}\boldsymbol{\theta}$

ℓ_2- regularization

$$L(\boldsymbol{\theta}) = \underbrace{-\sum_{i=1}^{n} \left[y^{(i)} \log(\sigma(z)) + (1 - y^{(i)}) \log(1 - \sigma(z)) \right]}_{\text{minimize the “distance” to the data}} + \underbrace{\frac{\lambda}{2} ||\mathbf{w}||_2^2}_{\text{keep the weights small}}$$

- Now **two** objectives: minimize the “distance” to the data but **also** keep the **weights small**
- Other classifiers and regression models can be regularized in the same way!

ℓ_2- regularization

$$+\frac{\lambda}{2}||\mathbf{w}||_2^2$$

Regularisation parameter λ

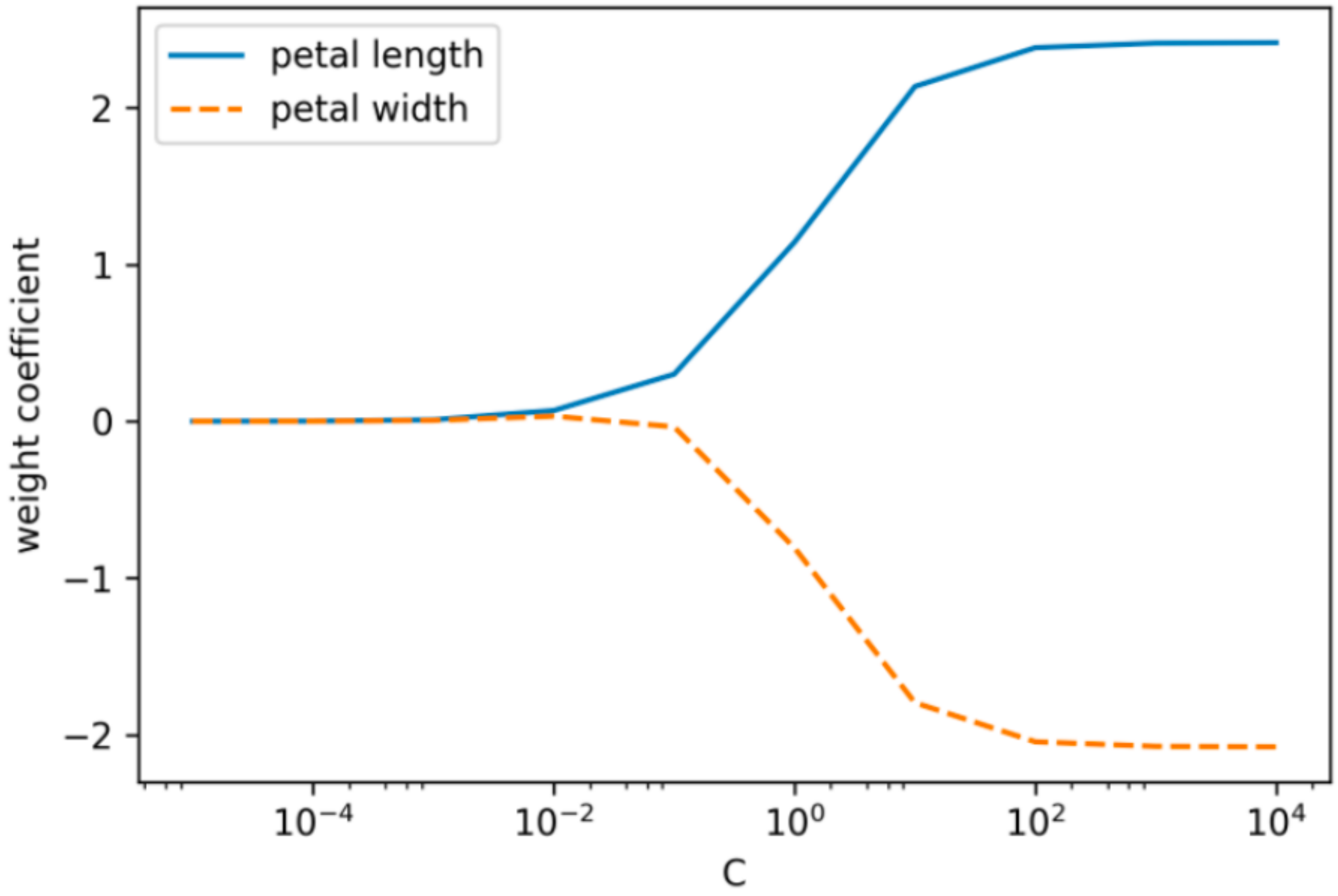
- Provides **control** for **how well** the training data are fitted while keeping the weights small
- **Increasing** the value of $\lambda \rightarrow$ **increases** regularisation **strength**
- The `LogisticRegression` class in scikit-learn implements a parameter C for regularisation
- C is the inverse of λ (C is the inverse regularisation parameter)
- Decreasing value of C $\rightarrow$ **increases** regularisation strength
- Effect of regularisation can be visualised by plotting the "ℓ_2**-regularization path**"

$$C := \frac{1}{\lambda}$$

ℓ_2- regularized logistic regression

$$+\frac{\lambda}{2}||\mathbf{w}||_2^2$$

$$C := \frac{1}{\lambda}$$



`03_logreg_iris_regularization.ipynb`

Figure from Python Machine Learning (Raschka & Mirjalili)

ℓ_2- regularized logistic regression

$$+\frac{\lambda}{2}||\mathbf{w}||_2^2 \qquad C := \frac{1}{\lambda}$$

Code example

- Use Wisconsin breast cancer data set in scikit-learn
- Use ALL features
- Split the data 100 times into different training and test sets (test_size=0.3)
- Intialise `LogisticRegression` class with `LogisticRegression(C=100.0, random_state=1)`
- Print out average and standard deviation of validation accuracy across the 100 splits

`03_logreg_cancer_regularization.ipynb`

ℓ_2- regularized logistic regression

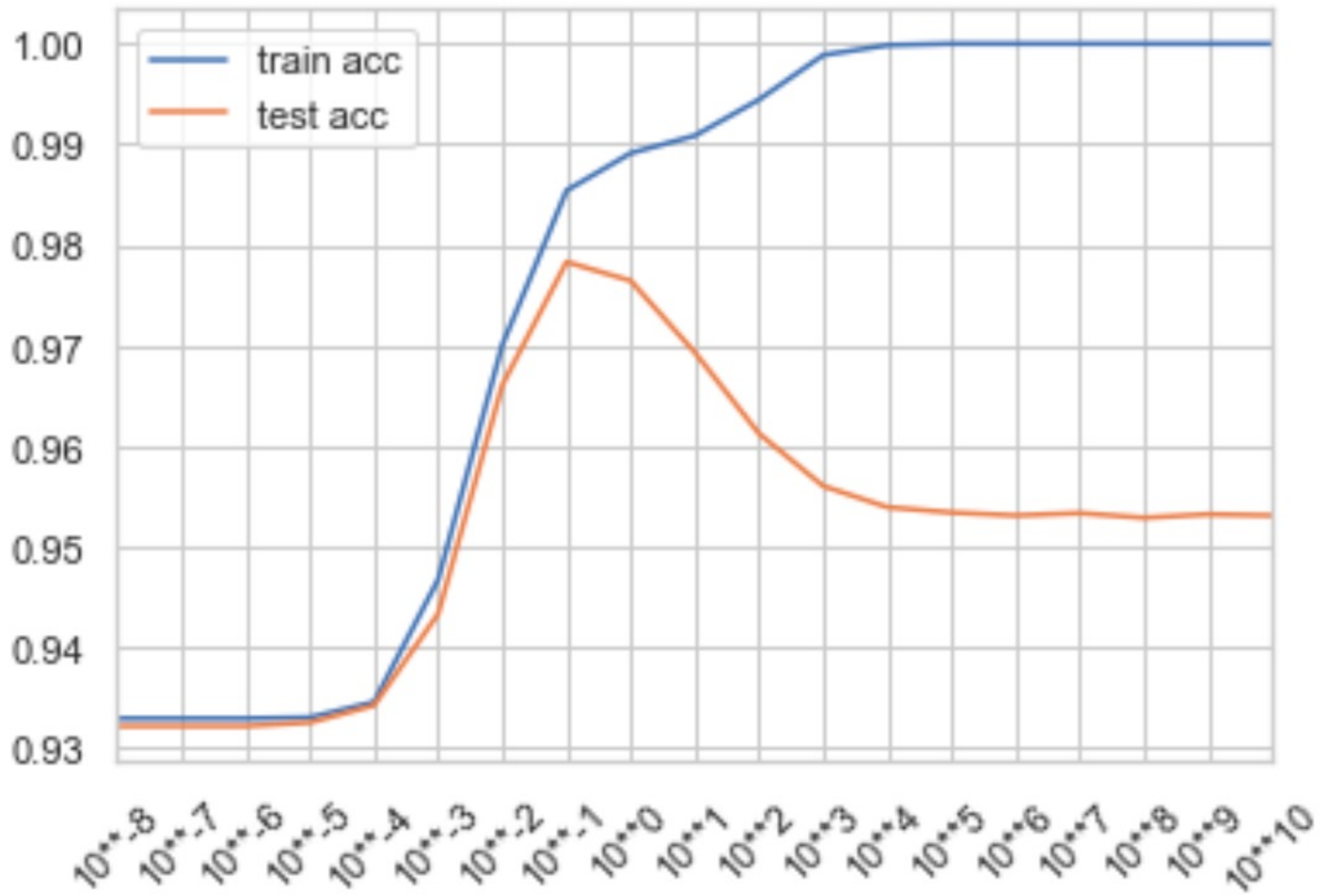
$$+ \frac{\lambda}{2}||\mathbf{w}||_2^2 \qquad C := \frac{1}{\lambda}$$

Code example

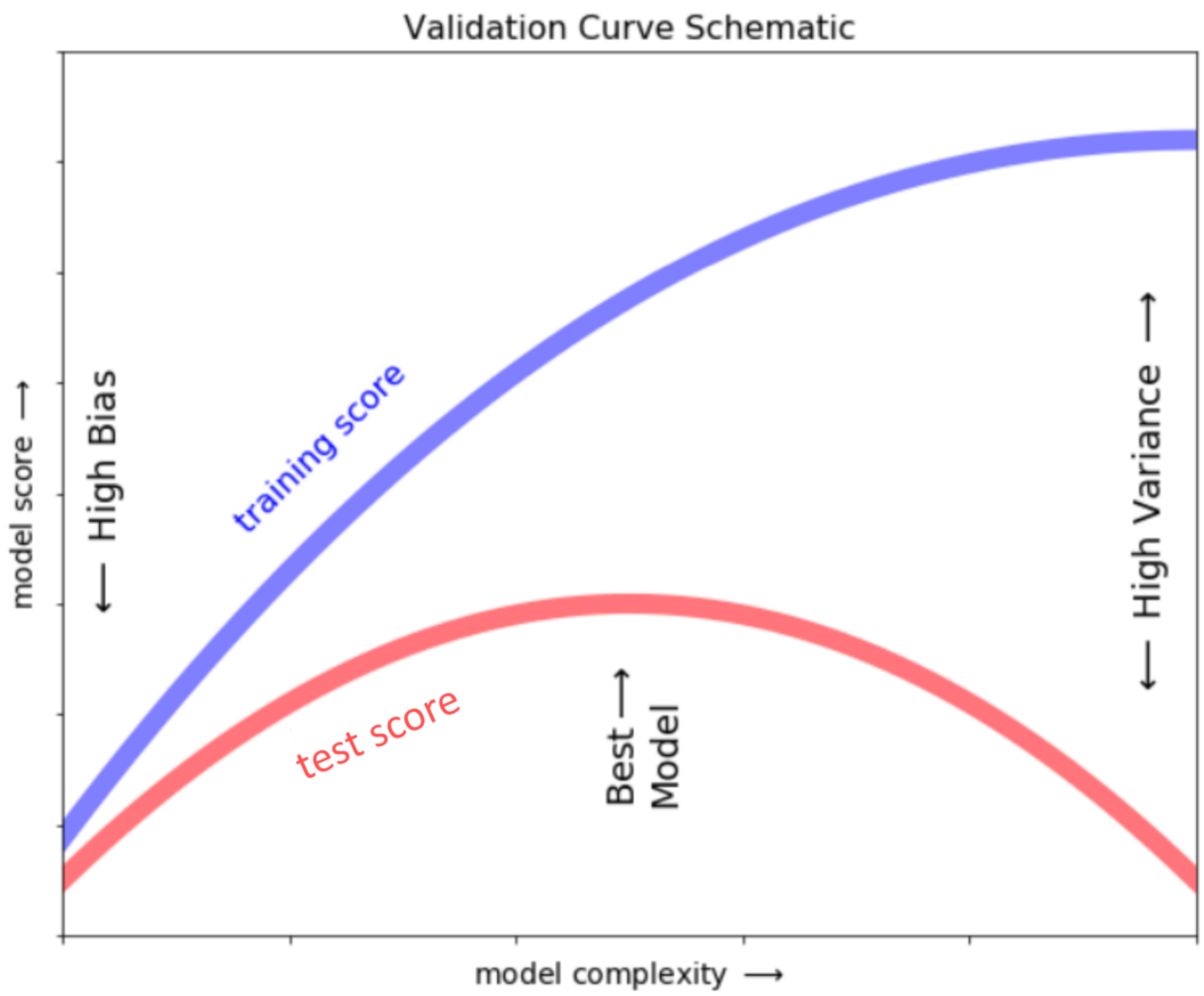
- Use Wisconsin breast cancer data set in scikit-learn
- Use ALL features
- Split the data 100 times into different training and test sets (test_size=0.3)
- Intialise `LogisticRegression` class with `LogisticRegression(C=10.0**c, random_state=1)`
  - small c varies from -8 to 10 (step by 1)
- Print out average and standard deviation of validation accuracy across the 100 splits
- Plot train and test accuracy across each C in one plot
- For which C does the model provide the best test accuracy? For which C does the model overfit? For which C do does the model underfit?

`03_logreg_cancer_regularization.ipynb`

ℓ_2- regularized logistic regression (exercise)



Bias – Variance trade-off



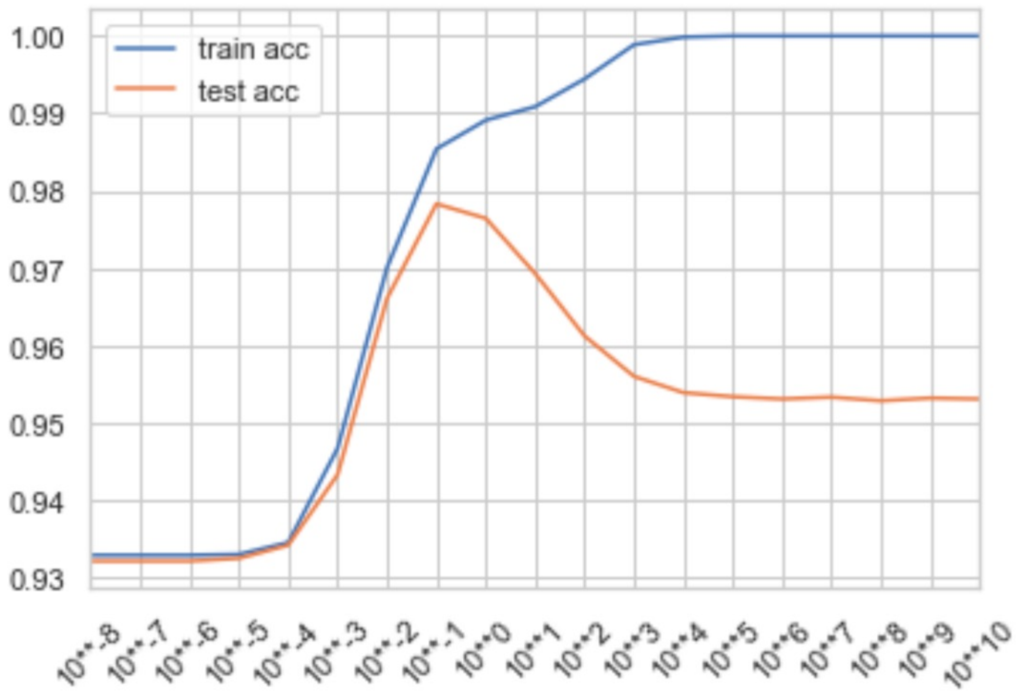
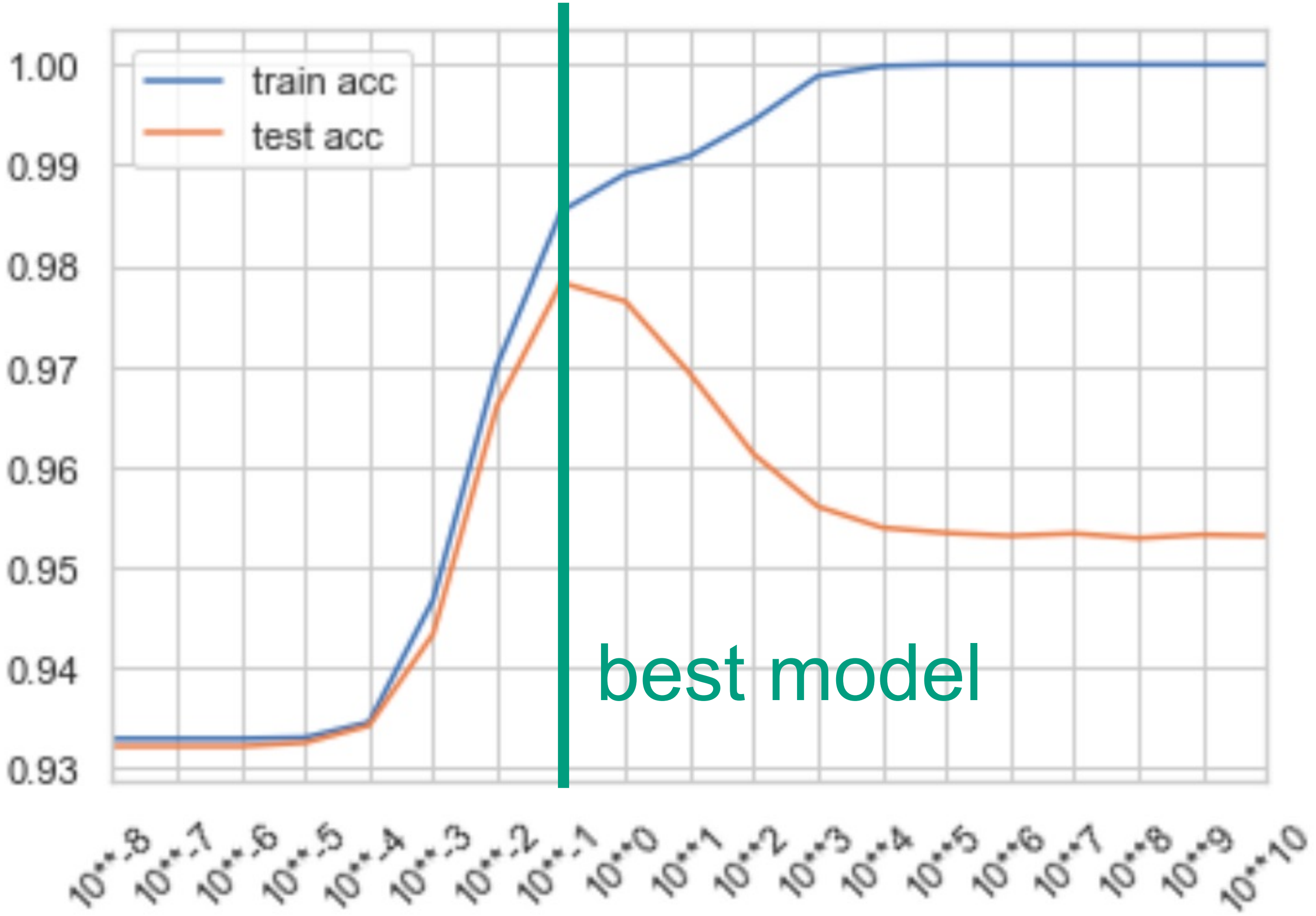


Figure from Python Machine Learning (Raschka & Mirjalili)

ℓ_2- regularized logistic regression (example)



Tackling overfitting via regularization

Sparsity-promoting regularization

ℓ_1- regularization (or L1 / Lasso)

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log(\sigma(z)) + (1-y^{(i)})\log(1-\sigma(z))\right] \;\; + \lambda||\mathbf{w}||_1$$

where $$||\mathbf{w}||_1 = \sum_{i=1}^{m}|w_i|$$

and $$\boldsymbol{\theta} = \begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_m\end{bmatrix} := \begin{bmatrix}b\\ \mathbf{w}\end{bmatrix}$$

and $$z := b + \mathbf{x}^{(i)}\mathbf{w} = \mathbf{x}^{(i)}\boldsymbol{\theta}.$$

Geometric interpretation

- Example: Mean squared error
- Two features

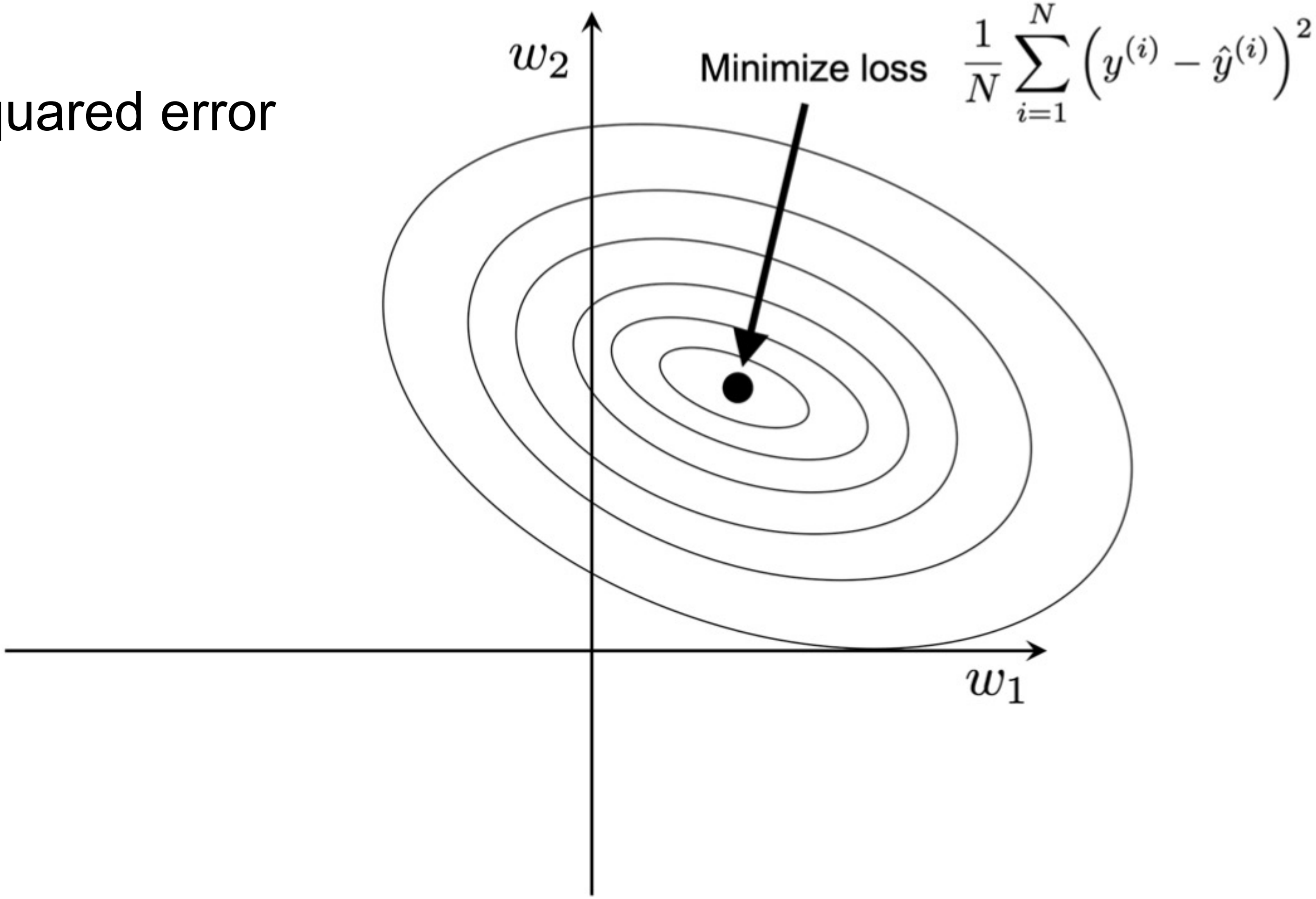


Figure from Python Machine Learning (Raschka & Mirjalili)

Geometric interpretation: ℓ_2- regularization

- Example: Mean squared error
- Two features

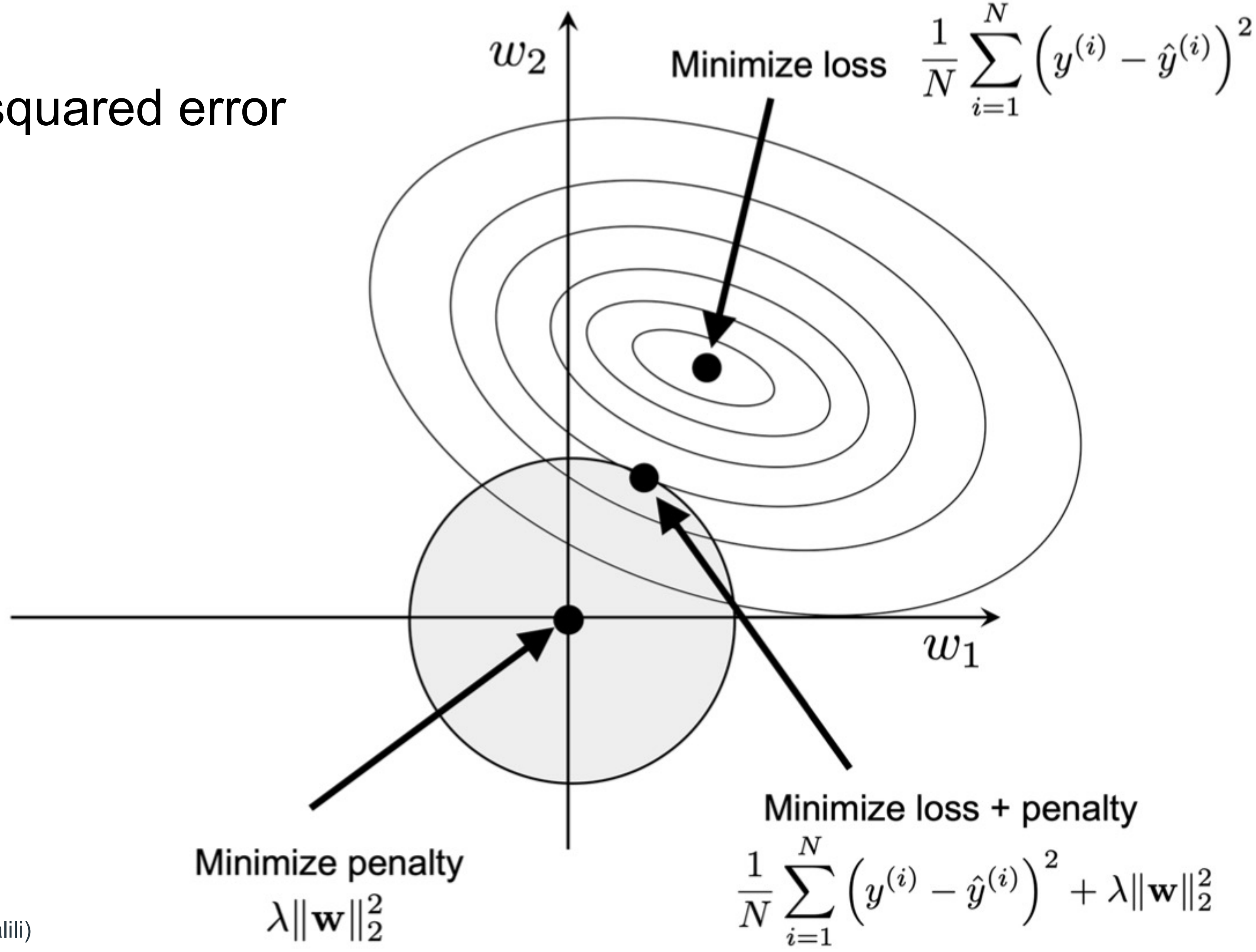


Figure from Python Machine Learning (Raschka & Mirjalili)

Geometric interpretation: ℓ_1- regularization

- Example: Mean squared error
- Two features

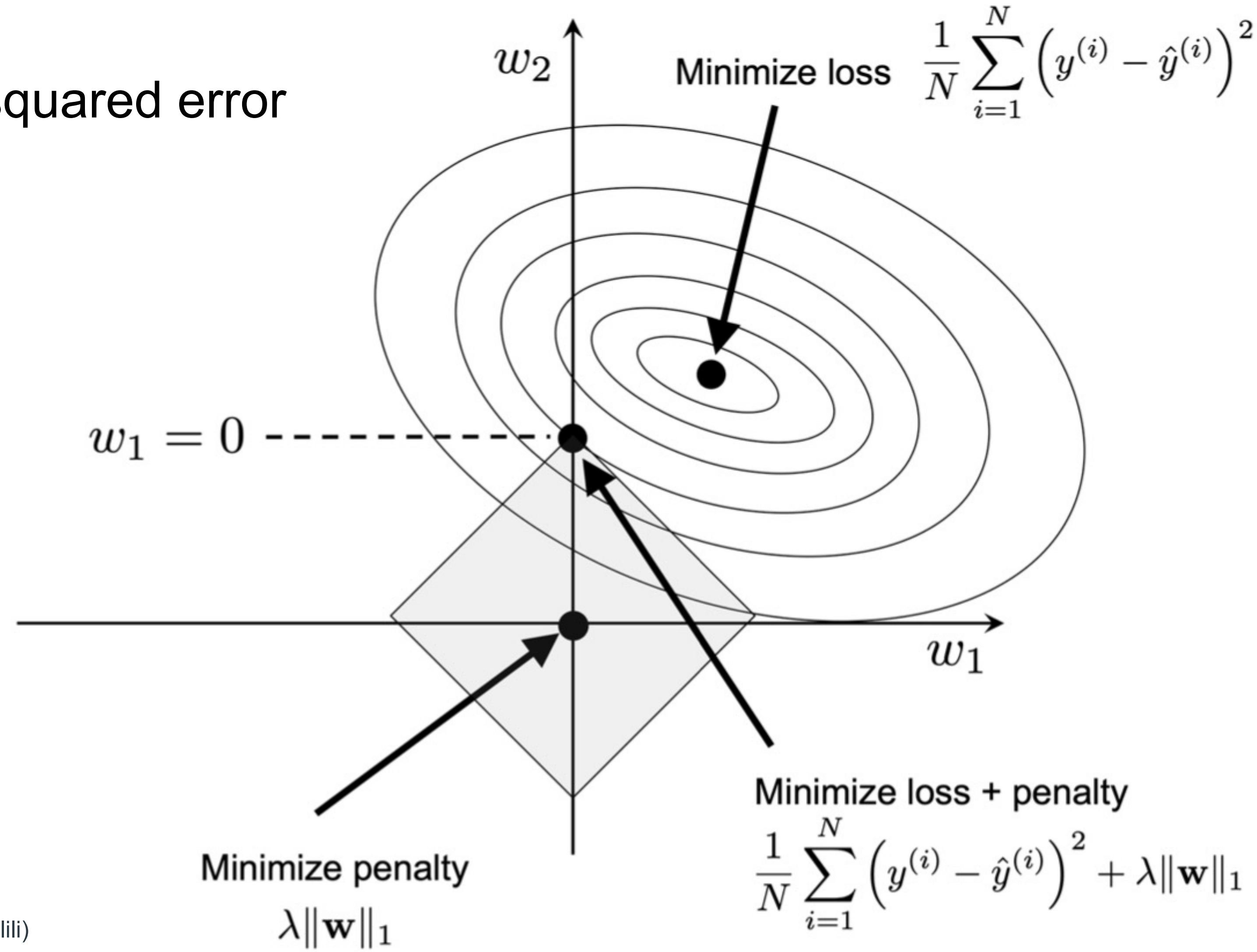


Figure from Python Machine Learning (Raschka & Mirjalili)

Geometric interpretation: ℓ_1- regularization

Due to the shape of the ℓ_1-loss this leads to a selection of weight such that many weights are zero (here: "automatic feature selection")

We say ℓ_1 regularization **promotes "sparsity"**

It leads to models with less parameters / lower complexity

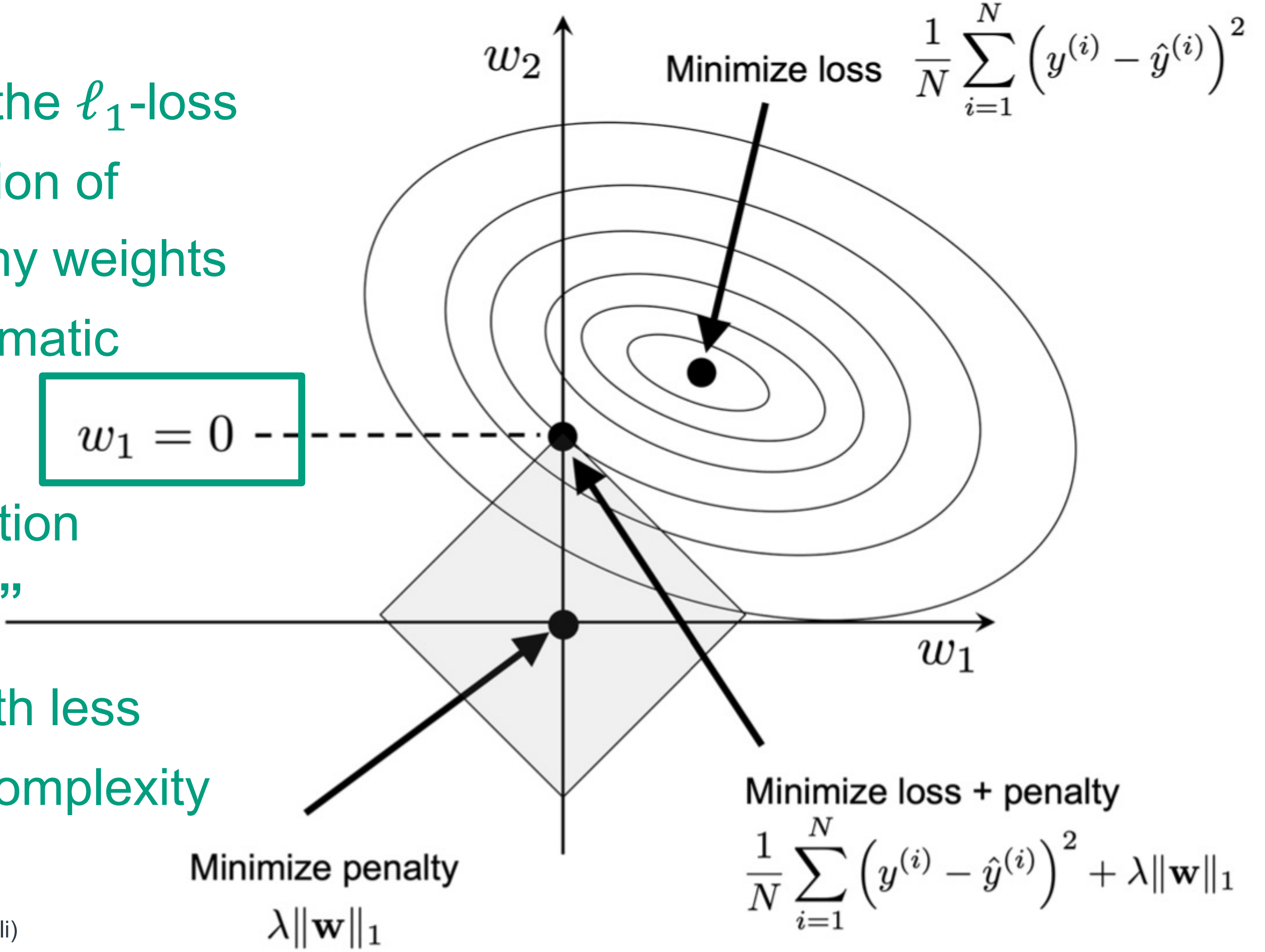
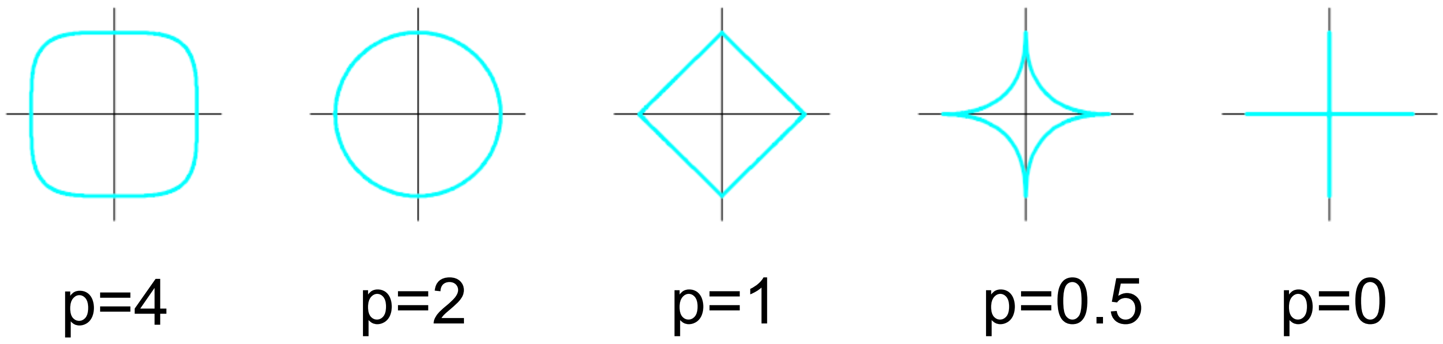


Figure from Python Machine Learning (Raschka & Mirjalili)

Geometric interpretation: ℓ_p- regularization



$$+\frac{\lambda}{p}||\mathbf{w}||_p^p$$

$$||\mathbf{w}||_p^p = \sum_{i=1}^{m} |w_i|^p$$

- Due to computational restriction in practice used are ℓ_1(Lasso), ℓ_2(Ridge), ℓ_1+ ℓ_2 (Elastic net)

Figure from Python Machine Learning (Raschka & Mirjalili)