Chapter 5: Linear regression

Volodymyr Vovk

v.vovk@rhul.ac.uk
Office Bedford 2-20

CS3920/CS5920 Machine Learning
Last edited: November 7, 2022

Version with solutions on slides 20–22

Plan

Simple linear regression

- Suppose we have only one feature ($p = 1$): each sample is a real number x.
- The linear regression model is

$$\hat{y} = wx + b,$$

  where $\hat{y}$ is the prediction for the label, and w (the slope) and b (the intercept) are parameters.
- w is the effect on $\hat{y}$ of a one unit increase in x.

Multiple linear regression

- If we have p features, the linear regression model is

$$\begin{aligned}\hat{y} &= w \cdot x + b \\ &= w[0]x[0] + w[1]x[1] + \cdots + w[p-1]x[p-1] + b,\end{aligned}$$

  where $x[j]$ is the $(j+1)$st feature.
- $w[j]$ is interpreted as the effect on $\hat{y}$ of a one unit increase in $x[j]$, holding all other features fixed.
- We will discuss three ways of estimating the coefficients $w[j]$: Least Squares, Ridge Regression, and Lasso.
- Once the parameters are estimated, we can use them for prediction.

Linear regression vs Nearest Neighbours

- Linear regression is an example of a parametric approach: it assumes a linear functional form for the prediction $\hat{y} = f(x)$. (For example, f is determined by two parameters in the case of simple linear regression.)
- Nearest Neighbours regression: does not assume a parametric form for $\hat{y} = f(x)$, and so is a non-parametric method.

Parametric methods: advantages and disadvantages

Advantages:

- They are often easy to fit, because we need to estimate only a small number of parameters.
- In many cases (including linear regression), the coefficients have simple interpretations and can be easily estimated.

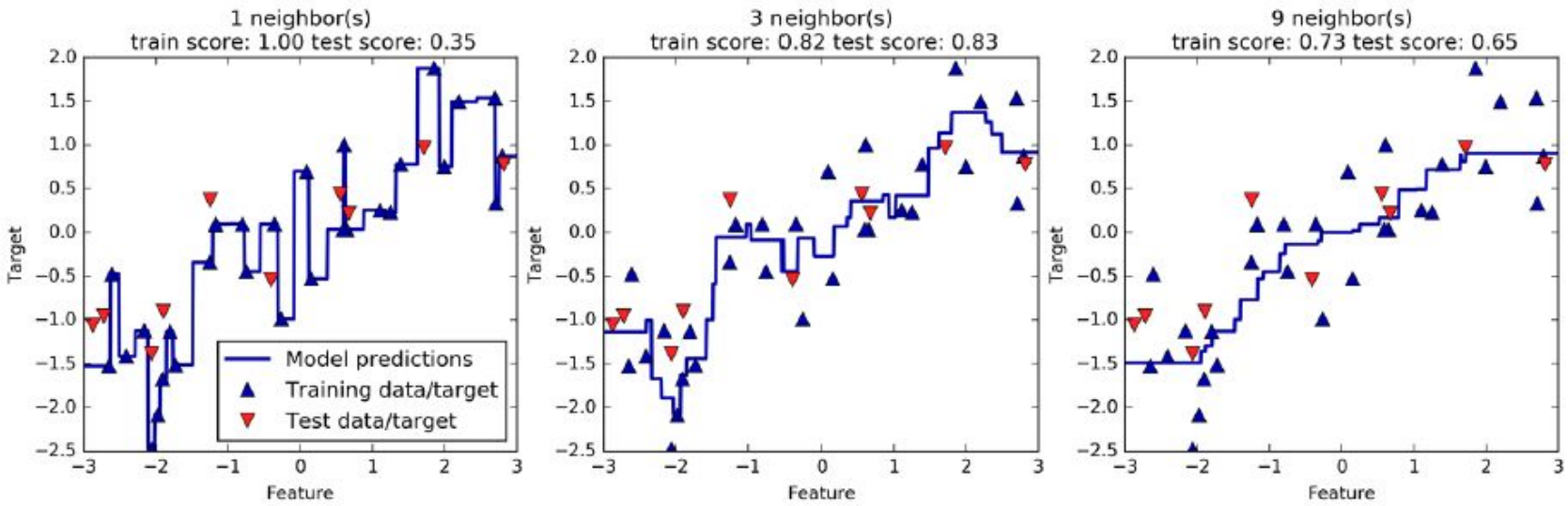
Disadvantage:

- They make strong assumptions about the form of $\hat{y} = f(x)$. If the specified functional form is far from the truth, the parametric method will perform poorly.

Non-parametric methods

- Non-parametric methods provide an alternative and more flexible approach.
- As we can see on the following slide (from Chapter 4):
  - The flexibility of K Nearest Neighbours diminishes as K grows.
  - The 1 Nearest Neighbour algorithm is extremely flexible and overfits.

Regression on `forge` (Chapter 4)



The two extremes

- The 1 Nearest Neighbour method is the most flexible method studied in machine learning.
  - Often overfits.
  - We are often interested in non-parametric methods that are less flexible.
- Linear regression is the most inflexible method studied in machine learning.
  - Often underfits.
  - We are often interested in parametric methods that are more flexible.

Estimating the regression parameters by Least Squares

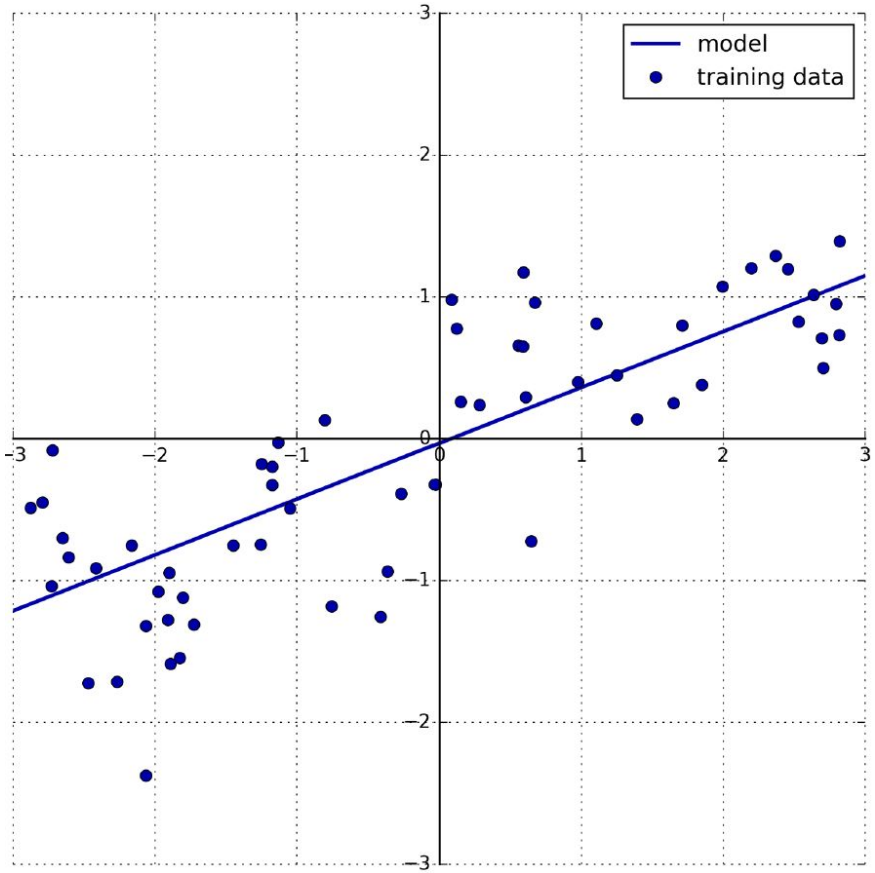
This is a classical approach to regression (going back to Gauss and Legendre, 1805!).

- The parameters are estimated using the Least Squares approach: we choose w and b to minimize

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \text{where } \hat{y}_i = w \cdot x_i + b.$$

- RSS stands for residual sum of squares (and $y_i - \hat{y}_i$ are sometimes referred to as residuals).
- The optimal w and b are usually denoted $\hat{w}$ and $\hat{b}$, respectively. The components of $\hat{w}$ are $\hat{w}[0], \hat{w}[1], \ldots, \hat{w}[p-1]$.

Predictions on wave



Feature engineering (1)

- If you compare the predictions made by the straight line with those made by the K Nearest Neighbors, using a straight line to make predictions seems very restrictive; all the fine details of the data are lost.
- But for datasets with many features, linear models can be very powerful!
- And we can add features to our training set at will. You will see (or have seen) examples in Lab 5 (for the `diabetes` and `wave` datasets).
- Including derived features is called feature engineering.

Feature engineering (2)

- Suppose that you believe that a parabola $\hat{y} = \alpha x^2 + \beta x + b$ would be a better fit for your data than a straight line (e.g., the `wave` dataset looks curving down if you go left to right).
- To fit a parabola, you add another feature to your data set, x^2.
- Fitting a linear function in the new features gives you $\hat{y} = w[0]x^2 + w[1]x + b$, a parabola!
- You can try it in Lab 5.

R^2 (1)

- Remember the residual sum of squares:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 .$$

- The no-data analogue is the total sum of squares

$$\text{TSS} := \sum_{i=1}^{n} (y_i - \bar{y})^2 ,$$

where $\bar{y}$ is the average label,

$$\bar{y} := \frac{1}{n} \sum_{i=1}^{n} y_i .$$

R^2 (2)

- R^2: the percentage of variability in the label that is explained by the data,

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} \in [0, 1].$$

- R^2 measures the performance on the training set only. A good (i.e., close to 1) R^2 is compatible with poor performance on new data (because of the possibility of overfitting).

Test R^2 (1)

- `scikit-learn` uses the notion of test R^2.
- Given a training set, find the estimates $\hat{w}[0], \hat{w}[1], \ldots, \hat{w}[p-1], \hat{b}$ of the parameters.
- For each test sample x_i^* (with true label y_i^*, not used at this step) compute the prediction

$$\begin{aligned}\hat{y}_i^* &= \hat{w} \cdot x_i^* + \hat{b} \\ &= \hat{w}[0]x_i^*[0] + \hat{w}[1]x_i^*[1] + \cdots + \hat{w}[p-1]x_i^*[p-1] + \hat{b}.\end{aligned}$$

- Compute the test RSS

$$\text{RSS} = \sum_{i=1}^{m} (y_i^* - \hat{y}_i^*)^2,$$

where m is the size of the test set.

Test R^2 (2)

- Compute the test TSS

$$\text{TSS} := \sum_{i=1}^{m} (y_i^* - \bar{y}^*)^2 ,$$

where $\bar{y}^*$ is the average label in the test set,

$$\bar{y}^* := \frac{1}{m} \sum_{i=1}^{m} y_i^* .$$

- Compute the test R^2 using the same formula,

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} \in (-\infty, 1].$$

Test R^2 (3)

- Notice that now R^2 can be negative!
- And notice that the test TSS has an element of data snooping in it; a slightly better definition would use $\bar{y}$ (the average label of the training set) instead of $\bar{y}^*$. (But the difference is usually tiny.)
- The same definition of test (and training) R^2 can be used for any linear model (such as Ridge Regression or Lasso discussed later in this chapter),
- and even for any regression model (except that $\hat{y}_i^*$ will be defined differently).

Exercise

- Consider the model $\hat{y} = 2x + 3$ (found from a training set using simple linear regression).
- What is the test R^2 of this model on the following test set?

$$(x_1^*, y_1^*) = (0, 1)$$
$$(x_2^*, y_2^*) = (2, 5).$$

(Answer: 0.)

- Give an example of a test set for which $R^2 < 0$.

Solution (1)

- *The predictions and RSS are:*

$$\hat{y}_1^* = 2 \times 0 + 3 = 3$$
$$\hat{y}_2^* = 2 \times 2 + 3 = 7$$
$$RSS = (1-3)^2 + (5-7)^2 = 8.$$

- *The average test label and TSS are:*

$$\bar{y}^* = \frac{1+5}{2} = 3$$
$$TSS = (1-3)^2 + (5-3)^2 = 8.$$

- *Finally,*

$$R^2 = \frac{TSS - RSS}{TSS} = \frac{8-8}{8} = 0.$$

Solution (2)

- *The following test set has* $R^2 < 0$*:*

$$(x_1^*, y_1^*) = (2, -2)$$
$$(x_2^*, y_2^*) = (3, 0).$$

- *The intuition behind this choice is: we want the performance of the given model to be awful!*
- *For a formal check, see the next slide.*

Solution (3)

- *The predictions and RSS are:*

$$\hat{y}_1^* = 2 \times 2 + 3 = 7$$
$$\hat{y}_2^* = 2 \times 3 + 3 = 9$$
$$RSS = (-2 - 7)^2 + (0 - 9)^2 = 162.$$

- *The average test label and TSS are:*

$$\bar{y}^* = \frac{-2 + 0}{2} = -1$$
$$TSS = (-2 + 1)^2 + (0 + 1)^2 = 2.$$

- *Finally,*

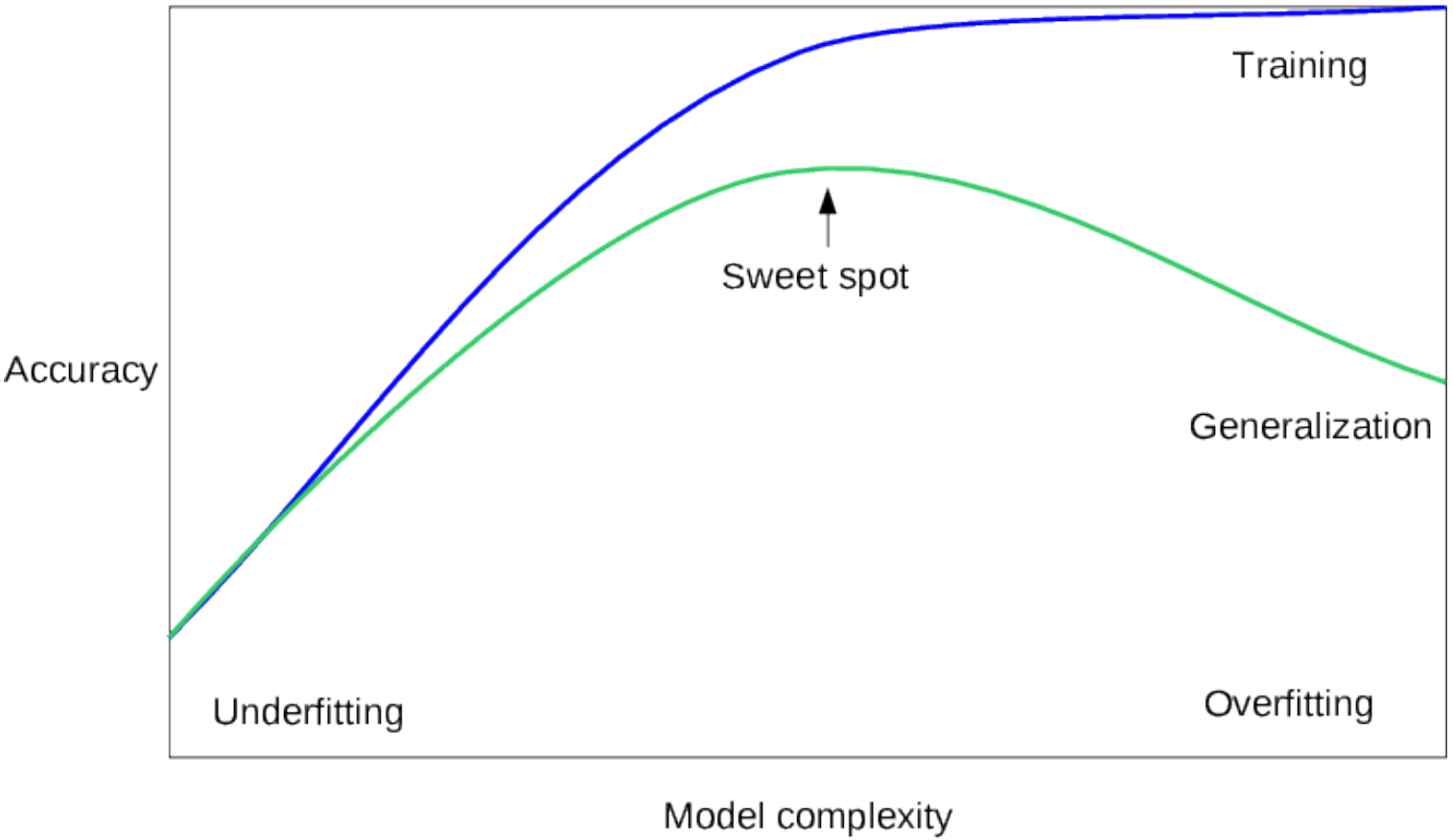
$$R^2 = \frac{TSS - RSS}{TSS} = \frac{2 - 162}{2} = -80.$$

Plan

Regularization

- In Ridge Regression the coefficients (w) are chosen not only so that they predict well on the training data, but to make their magnitude as small as possible.
- Intuitively, this means each feature should have as little effect on the outcome as possible (which translates to having a small slope), while still predicting well.
- This is an example of regularization, a standard way to avoid overfitting.
- Adding regularization means moving left in the figure of Chapter 4 (reproduced on the next slide).

Looking for a balance (Chapter 4)



Ridge Regression (1)

- The Least Squares procedure estimates w and b by minimizing

$$\text{RSS} = \sum_{i=1}^{n} (y_i - w \cdot x_i - b)^2 .$$

- Ridge Regression is similar, except that the parameters are estimated by minimizing

$$\text{RSS} + \alpha \|w\|^2 = \text{RSS} + \alpha \sum_{j=0}^{p-1} w[j]^2,$$

where $\alpha \geq 0$ is the regularization parameter (or tuning parameter).

- We are trading off two criteria:
  - fitting the data well
  - moving the parameter estimates towards zero

Ridge Regression (2)

- When $\alpha = 0$: Ridge Regression coincides with Least Squares.
- As $\alpha \to \infty$, the Ridge Regression coefficient estimates approach zero.
- Selecting a good value for α is critical.
- Notice: the shrinkage penalty $\alpha \sum_{j=0}^{p-1} w[j]^2$ is not applied to the intercept b!
  - we want to shrink the estimated association of each features with the label
  - we do not want to shrink the estimated intercept

Terminology

- A fancier name for the Euclidean norm

$$\sqrt{\sum_{j=0}^{p-1} w[j]^2}$$

  is the L_2 norm.
- In which case $\|w\|$ may be written as $\|w\|_2$, and we may be said to be using L_2 regularization.

Plan

Lasso (1)

- We can modify Ridge Regression in such a way that it achieves "model selection" as by-product.
- Ridge Regression typically includes all p features in the final model.
- The shrinkage penalty $\alpha \sum_j w[j]^2$ shrinks the coefficients towards zero, but does not set them exactly to zero (unless $\alpha = \infty$).
- May not be a problem for prediction accuracy, but may complicate model interpretation when p is large.
- Lasso will set many coefficients $w[j]$ to zero.

Lasso (2)

- The Lasso coefficients $\hat{w}$ and $\hat{b}$ minimize the quantity

$$\text{RSS} + \alpha \sum_{j=0}^{p-1} |w[j]| \,.$$

- The predictions are computed as before: the prediction for a test sample x^* is

$$\hat{w} \cdot x^* + \hat{b} = \hat{w}[0]x^*[0] + \cdots + \hat{w}[p-1]x^*[p-1] + \hat{b},$$

where $x^*[j]$ are the features.

Lasso (3)

- The only difference from Ridge Regression is that the L_2 shrinkage penalty $\alpha \sum_j w[j]^2$ has been replaced by the L_1 shrinkage penalty $\alpha \sum_j |w[j]|$.
- The L_1 norm of a coefficient vector $w = (w[0], \ldots, w[p-1])$ is defined by

$$\|w\|_1 = \sum_{j=0}^{p-1} |w[j]| .$$

Remark: Lasso = least absolute shrinkage and selection operator.

Lasso and model selection

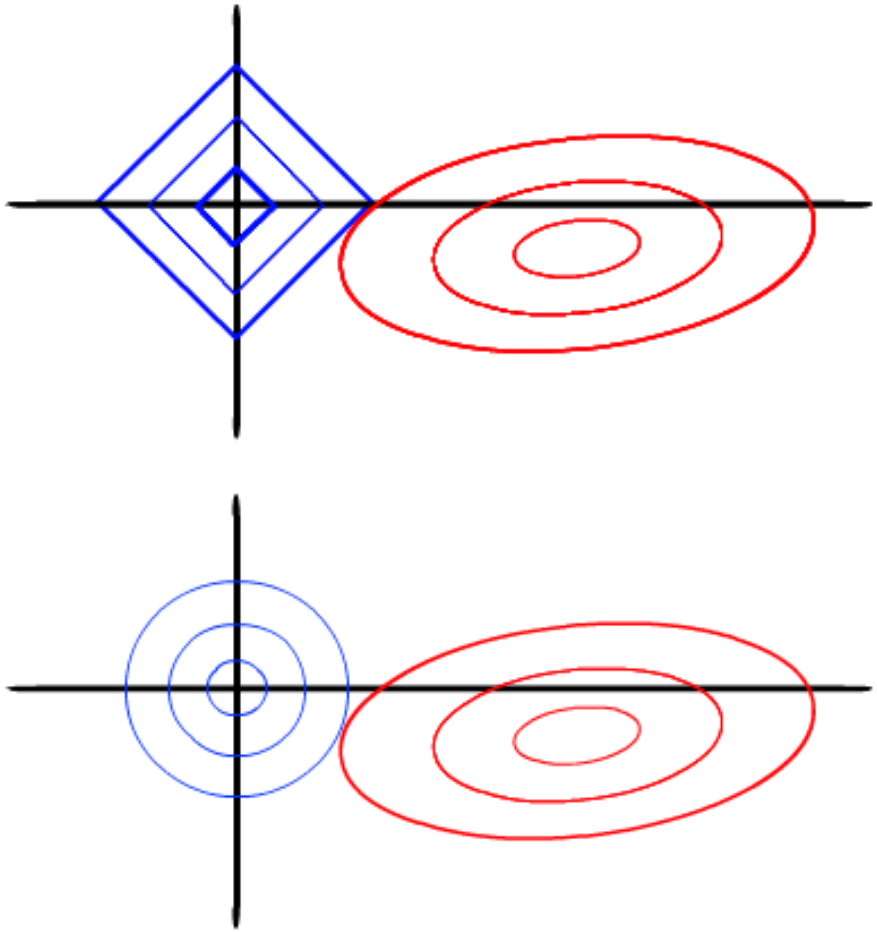
- As Ridge Regression, the Lasso shrinks the coefficient estimates towards zero.
- However, using the L_1 shrinkage penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the regularization parameter α is sufficiently large.
- So the Lasso performs model selection (yielding sparse models, i.e., models that involve only some of the features).

The case $p = 2$ (*)

It helps intuition to consider the case $p = 2$ (2 features):

- An L_2-ball $\|w\|_2 \leq r$ looks like a 2D ball ($=$ circle) around 0.
- But an L_1-ball $\|w\|_1 \leq r$ looks like a diamond around 0!
- The "RSS-ball" $\text{RSS}(w) \leq r$ is an ellipse around $\hat{w}^{\text{LS}}$.
- If we draw the L_2-ball and RSS-ball containing $\hat{w}^{\text{RR}}$ on their surfaces, these two balls will touch.
- If we draw the L_1-ball and RSS-ball containing $\hat{w}^{\text{L}}$ on their surfaces, they will also touch.
- Here $\hat{w}^{\text{LS}}$, $\hat{w}^{\text{RR}}$, and $\hat{w}^{\text{L}}$ mean the vector parameter w selected by Least Squares, Ridge Regression, and Lasso, respectively.

The geometry of model selection (*) (1)



The geometry of model selection (*) (2)

The picture on the previous slide: $p = 2$.

- There is a good chance that the diamond and ellipse will touch at the diamond's corner, especially when the diamond is small (i.e., α is large). Some coefficients will then be zero.
- In higher dimensions ($p \gg 2$), many of the coefficient estimates may equal zero simultaneously.
- It would be a remarkable coincidence for this to happen for the circle and ellipse.

Elastic net

- `scikit-learn` also provides the `ElasticNet` class, which combines the penalties of Lasso and Ridge Regression.
- In practice, this combination works best, though at the price of having two parameters to adjust: one for the L_1 regularization, and one for the L_2 regularization.

Plan

Parameters

- The regularization parameter of Ridge Regression and Lasso is $\alpha \geq 0$.
- Large values of α lead to simpler models.
- Least Squares does not have any parameters (it's the special case of both Ridge Regression and Lasso corresponding to $\alpha = 0$).
- Tuning α is very important.
- Usually α is searched for on a logarithmic scale (for example, we may try $\alpha = 2^k$ for $k = -5, -4, \ldots, 5$).

L_2 or L_1 regularization

- If you assume that only a few of your features are actually important, you should use L_1 regularization.
- Otherwise, you should default to L_2.
- L_1 can also be useful if interpretability of the model is important: when we have only a few features, it is easier to explain which features are important to the model, and what the effects of these features are.

Strengths and weaknesses

++ Linear models are very fast to train, and also fast to predict.

++ They scale to very large datasets and work well with sparse data.

\+ Linear models make it easy to understand how a prediction is made.

– It is often not entirely clear why coefficients are the way they are (especially if your dataset has highly correlated features).

\+ Linear models often perform well when the number of features is large compared to the number of samples ($p \gg n$).

– However, for smaller p other models might yield better generalization performance (such as Support Vector Machines, to be discussed in Chapter 8).

Using linear models for classification

- Later in the course we will also consider using linear models for classification.
- The idea for binary classification: instead of predicting using a linear function,

$$\hat{y} = w \cdot x^* + b,$$

predict $+1$ if $w \cdot x^* + b > 0$ and predict -1 if $w \cdot x^* + b < 0$ (and do what you want if $w \cdot x^* + b = 0$). The three main methods:

  - Logistic regression.
  - Linear discriminant analysis.
  - Maximum margin classifier.

- Linear models for classification share strengths and weaknesses of linear models for regression.

Estimator classes

- Now we continue the discussion started in Lab 2.
- All machine learning models in `scikit-learn` are implemented in their own classes, which are called `Estimator` classes.
- The `Estimator` classes implementing the algorithms we have seen so far (or are about to see):
  - `KNeighborsClassifier` class in the `neighbors` module (Lab 2).
  - `LinearRegression` in the `linear_model` module. This class implements Least Squares.
  - `Ridge` in the `linear_model` module. This class implements Ridge Regression.
  - `Lasso` in the `linear_model` module. This class implements Lasso.

Three fundamental methods (1)

In supervised learning, each `Estimator` class has three fundamental methods associated with it:

- To instantiate an `Estimator` class into an object (and perhaps set some parameters of the model), we use the `__init__` method. Example:

```
knn = KNeighborsClassifier(n_neighbors=1)
ridge = Ridge()
```

Three fundamental methods (2)

- All estimators have a `fit` method, which is used to build the model.
  - The `fit` method always requires as its first argument the data `X`, typically represented as a `NumPy` array where each row represents a single sample.
  - Supervised estimators also require a `y` argument, which is a one-dimensional `NumPy` array containing the labels for regression or classification.
- To create a prediction in the form of a new output like `y`, you use the `predict` method.

Method chaining (1)

- The `fit` method of all `scikit-learn` models returns `self`. This allows us to instantiate a model and fit it in one line:

```
linreg = LinearRegression().fit(X_train, y_train)
```

We have used the return value of `fit` (which is `self`) to assign the trained model to the variable `linreg`.

- This concatenation of method calls (here `__init__` and then `fit`) is known as method chaining.
- Another common application of method chaining in `scikit-learn` is to fit and predict in one line:

```
linreg = LinearRegression()
y_pred = linreg.fit(X_train, y_train).predict(X_test)
```

Method chaining (2)

- Finally, you can even do model instantiation, fitting, and predicting in one line:

```
y_pred = LinearRegression().fit(X_train,
  y_train).predict(X_test)
```

- The last shortest variant is not ideal. A lot is happening in a single line, which might make the code hard to read. Additionally, the fitted linear regression model isn't stored in any variable, so we can't inspect it or use it to predict on any other data.

Further reading

M17, Chapter 2.

Feature engineering: M17, Chapter 4.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright (2016). Statistical learning with sparsity: the lasso and generalizations.

An advanced book on Lasso and related methods.

`scikit-learn` user guide, Section 3.3.4.6: test R^2.

`http://scikit-learn.org/stable/user_guide.html`.

Robert Tibshirani (1996).

Regression shrinkage and selection via the lasso.

Journal of the Royal Statistical Society B 58:267–288.

The paper that introduced Lasso in machine learning.

Further reading: conformal prediction

V05, Section 2.3: computationally efficient conformal predictor over Ridge Regression.

Jing Lei (2019).
Fast exact conformalization of lasso using piecewise linear homotopy.
Biometrika 106:749–764.
Computationally efficient conformal predictor over Lasso and elastic net.

Permissions and changes

Permissions:

- The pictures from M17 are used with permission from Andreas Müller.
- The picture on slide 35 is used with permission from Dmitry Laptev.

Changes since first posted:

- on 27 October 2022, replaced `Boston Housing` by `diabetes` on slide 12.