End-to-End Machine Learning Project

Chapter 2: pp 39-41, 69, 76-78

End-to-End Machine Learning Project

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Real Data

- Popular open data repositories:
  - UC Irvine Machine Learning Repository
  - Kaggle datasets
  - Amazon's AWS datasets
- Meta portals (they list open data repositories):
  - *http://dataportals.org/*
  - *http://opendatamonitor.eu/*
  - *http://quandl.com/*
- Other pages listing many popular open data repositories:
  - Wikipedia's list of Machine Learning datasets
  - Quora.com question
  - Datasets subreddit

Notations

- m is the number of instances in the dataset you are measuring the RMSE on.
  - For example, if you are evaluating the RMSE on a validation set of 2,000 districts, then $m = 2{,}000$.
- $\mathbf{x}^{(i)}$ is a vector of all the feature values (excluding the label) of the i^{th} instance in the dataset, and $y^{(i)}$ is its label (the desired output value for that instance).
  - For example, if the first district in the dataset is located at longitude –118.29°, latitude 33.91°, and it has 1,416 inhabitants with a median income of \$38,372, and the median house value is \$156,400 (ignoring the other features for now), then:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix}$$

and:

$$y^{(1)} = 156,400$$

- $\mathbf{X}$ is a matrix containing all the feature values (excluding labels) of all instances in the dataset. There is one row per instance and the i^{th} row is equal to the transpose of $\mathbf{x}^{(i)}$, noted $(\mathbf{x}^{(i)})^T$.[4]
  - For example, if the first district is as just described, then the matrix $\mathbf{X}$ looks like this:

$$\mathbf{X} = \begin{pmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \left(\mathbf{x}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(1999)}\right)^T \\ \left(\mathbf{x}^{(2000)}\right)^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,416 & 38,372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Summary

- $\mathbf{x}^{(i)}$: *i*-th vector (*i*-th training example)
- x_j : *j*-th element of a vector (scalar)
- $x_j^{(i)}$: *j*-th element of the *i*-th vector (scalar)

Fisher Iris (sepal length, sepal width, petal length, petal width, class)

5.3000	3.7000	1.5000	0.2000	setosa
5.0000	3.3000	1.4000	0.2000	setosa
7.0000	3.2000	4.7000	1.4000	versicolor
6.9000	3.1000	5.4000	2.1000	virginica

- Find $\mathbf{X}$, $\mathbf{y}$
- What is $\mathbf{x}^{(2)}, x_4^{(3)}, y_1$?

$\mathbf{X}$ =

5.3000	3.7000	1.5000	0.2000
5.0000	3.3000	1.4000	0.2000
7.0000	3.2000	4.7000	1.4000
6.9000	3.1000	5.4000	2.1000

$\mathbf{y}$ =

setosa
setosa
versicolor
virginica

$\mathbf{x}^{(2)}$ =
$[5.0000 \quad 3.3000 \quad 1.4000 \quad 0.2000]^T$

$x_4^{(3)}$ = 1.4000

y_1 = setosa

- h is your system's prediction function, also called a *hypothesis*. When your system is given an instance's feature vector $\mathbf{x}^{(i)}$, it outputs a predicted value $\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$ for that instance ($\hat{y}$ is pronounced "y-hat").
  - For example, if your system predicts that the median housing price in the first district is \$158,400, then $\hat{y}^{(1)} = h(\mathbf{x}^{(1)}) = 158{,}400$. The prediction error for this district is $\hat{y}^{(1)} - y^{(1)} = 2{,}000$.
- RMSE($\mathbf{X}$,h) is the cost function measured on the set of examples using your hypothesis h.

We use lowercase italic font for scalar values (such as m or $y^{(i)}$) and function names (such as h), lowercase bold font for vectors (such as $\mathbf{x}^{(i)}$), and uppercase bold font for matrices (such as $\mathbf{X}$).

Distance Measures

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Equation 2-2. Mean absolute error (MAE)

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

- Computing the root of a sum of squares (RMSE) corresponds to the *Euclidean norm*: it is the notion of distance you are familiar with. It is also called the ℓ_2 *norm*, noted $\|\cdot\|_2$ (or just $\|\cdot\|$).
- Computing the sum of absolutes (MAE) corresponds to the ℓ_1 *norm*, noted $\|\cdot\|_1$. It is sometimes called the *Manhattan norm* because it measures the distance between two points in a city if you can only travel along orthogonal city blocks.
- More generally, the ℓ_k *norm* of a vector $\mathbf{v}$ containing n elements is defined as $\|\mathbf{v}\|_k = \left(|v_0|^k + |v_1|^k + \cdots + |v_n|^k\right)^{\frac{1}{k}}$. ℓ_0 just gives the number of non-zero elements in the vector, and ℓ_∞ gives the maximum absolute value in the vector.

Take a Look at the Data

- https://github.com/ageron/handson-ml2/blob/master/02_end_to_end_machine_learning_project.ipynb
- 10 features → 10 dimensional vector

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

Figure 2-5. Top five rows in the dataset

- 20,640 instances (examples) → m = 20,640

```
In [6]: housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Figure 2-6. Housing info

Feature Scaling

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

Figure 2-5. Top five rows in the dataset

- Min-max Scaling (normalization)
  - Subtract min value then divide by (max – min)
  - Range = [0,1]
- Standardization
  - Subtract mean value then divide by standard deviation

Grid Search

- Iterate over hyperparameters for all possible combinations to find the best model
- Example: hyperparameters C, γ
  - Specify $C = [10, 100, 1000], \gamma = [0.1, 0.5, 0.9]$
  - Determine the cartesian product and train with $\{C, \gamma\}$ → 9 models
  - Evaluate performance using cross-validation to find the best selection of $\{C, \gamma\}$

Random Search

- Randomize the values of C, γ for each iteration
- Stop after N iterations