Computer Science

CSC380: Principles of Data Science

Basic machine learning 1

Xinchen Yu

- Probability
- Statistics

- Data Visualization
- Predictive modeling

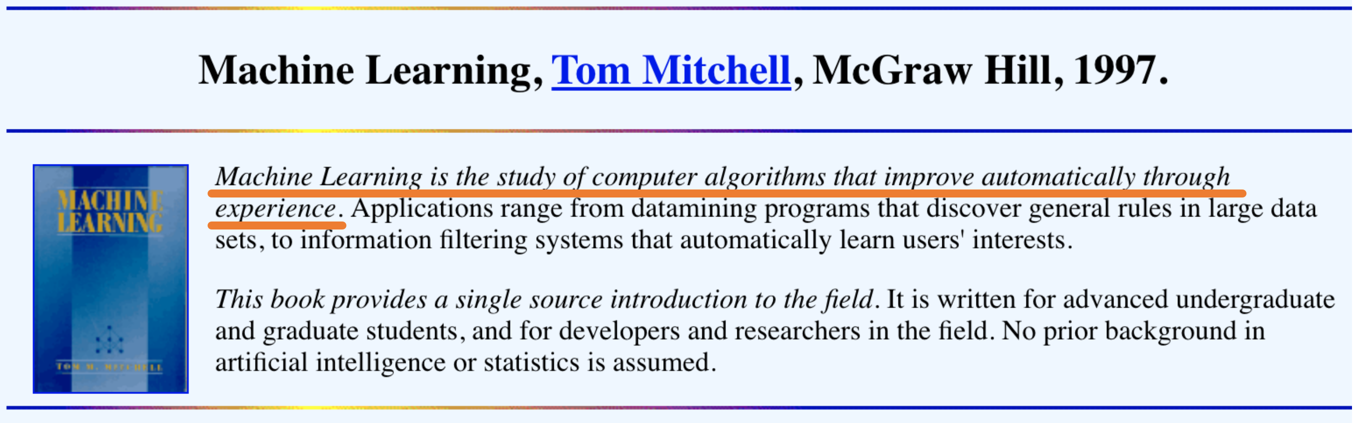
Outline

- Introduction to Machine Learning
- Supervised Learning: Linear Regression
- Overfitting and underfitting
- Regularization in regression
- Feature Selection

Introduction to Machine Learning

What is machine learning (ML)?

- **Tom Mitchell** established Machine Learning Department at CMU (2006).



Machine Learning, Tom Mitchell, McGraw Hill, 1997.

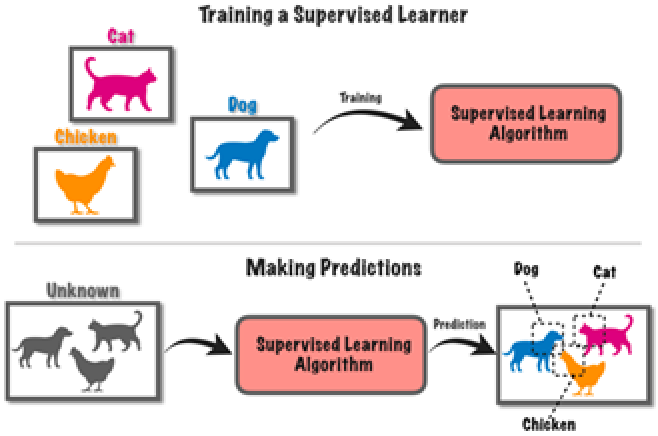
Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests.

This book provides a single source introduction to the field. It is written for advanced undergraduate and graduate students, and for developers and researchers in the field. No prior background in artificial intelligence or statistics is assumed.

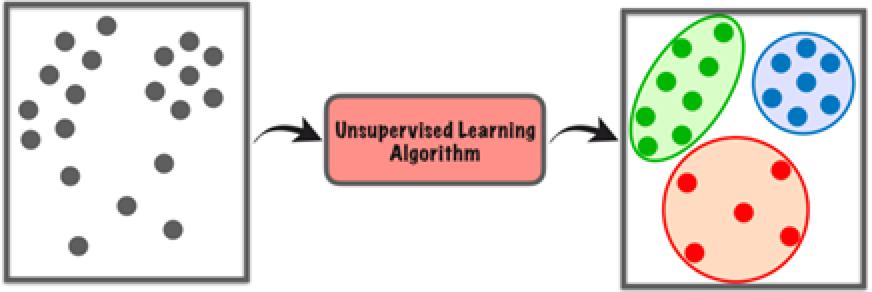
- In short: algorithms adapt to data
- A subfield of **Artificial Intelligence (AI)** – computers perform “intelligent” tasks.
- Classical AI vs ML: rule-driven approaches vs. data-driven approaches

Supervised vs Unsupervised Learning

- **Supervised Learning** - Training data consist of inputs and outputs
  - Classification, regression, translation, …

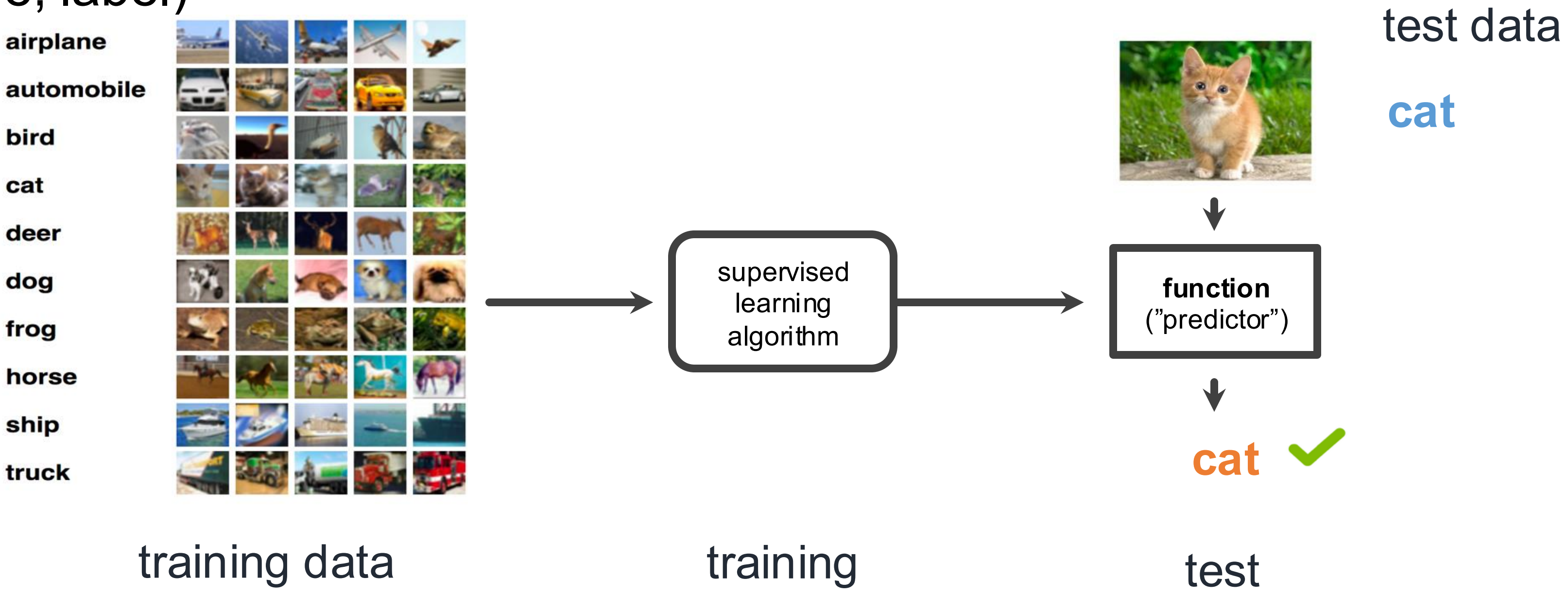

Training a Supervised Learner
Cat
Dog
Chicken
Training
Supervised Learning Algorithm
Making Predictions
Unknown
Supervised Learning Algorithm
Prediction
Dog
Cat
Chicken

- **Unsupervised Learning** – Training data only contain inputs
  - Clustering, dimensionality reduction, segmentation, …



Supervised learning

- Training / test data: datasets comprised of ***labeled examples***: pairs of (feature, label)



airplane

automobile

bird

cat

deer

dog

frog

horse

ship

truck

training data

supervised learning algorithm

training

test data

cat

function ("predictor")

cat ✓

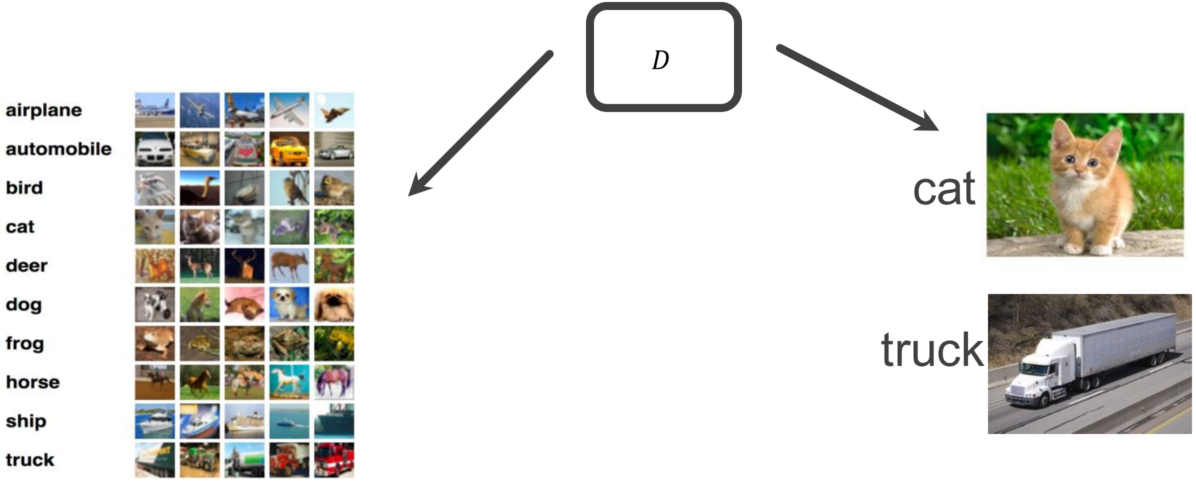
test

How should test data be chosen?

- Test data cannot be identical to training data.
- Test data cannot be just ONE data point.

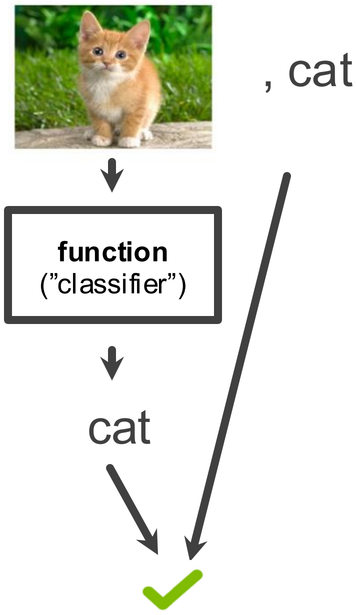
Supervised learning setup

- Key assumption: training and test data are drawn from the same *population*, or *data generating distribution* D
  - They are assumed to be IID samples: independent and identically distributed
- Training and test data are independent



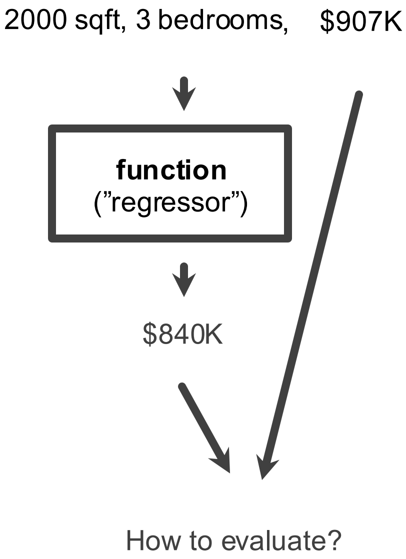
Supervised learning setup

- Scenario 1: classification



- Scenario 2: regression

(e.g. house price prediction)



- Loss function ℓ: measures the quality of prediction $\hat{y}$ respect to true label y

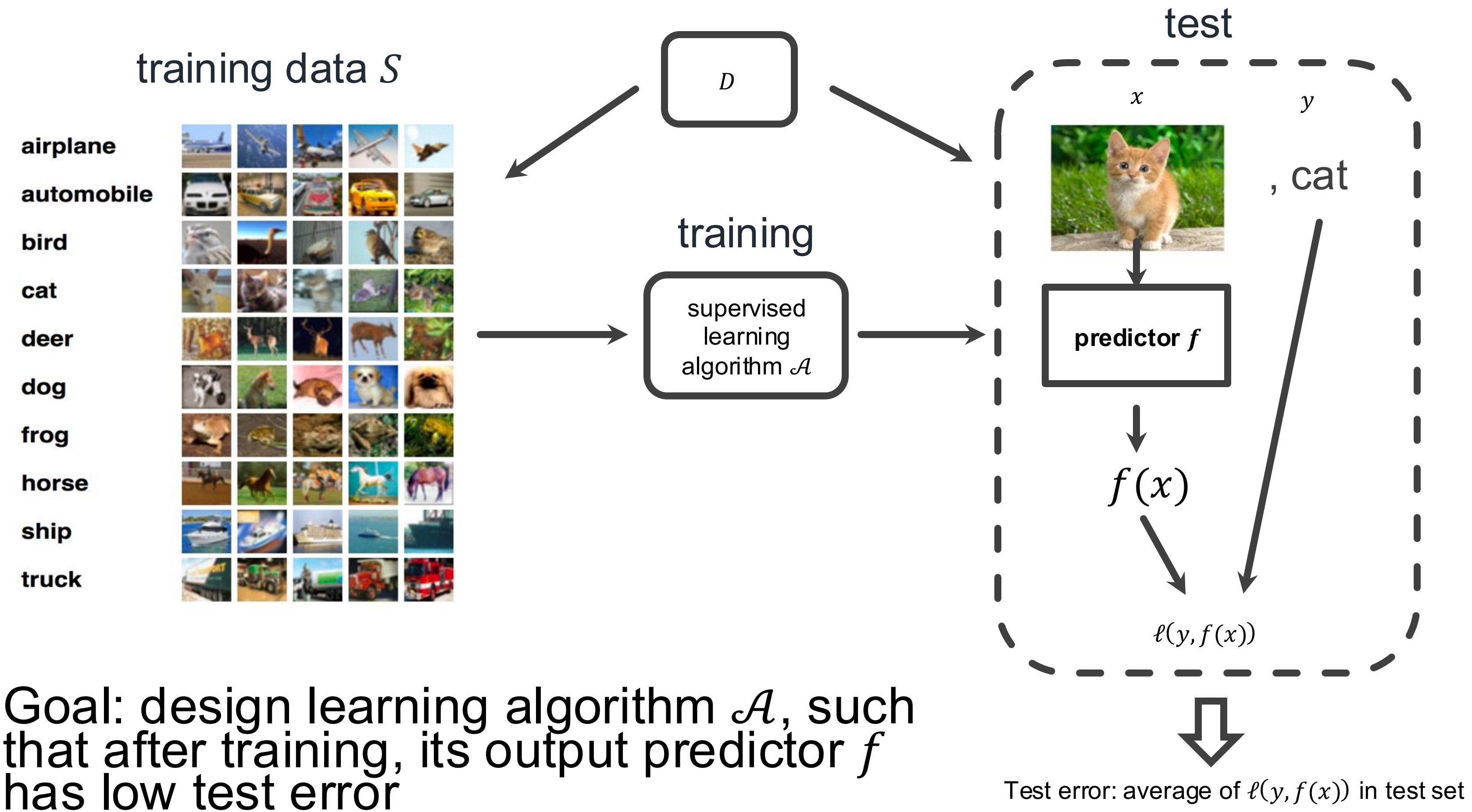
- Examples:
  - Classification error

    $\ell(y, \hat{y}) = 1$ if $y \neq \hat{y}$, and zero otherwise

  - Square loss

    $\ell(y, \hat{y}) = (y - \hat{y})^2$ - regression

Supervised learning setup in one figure

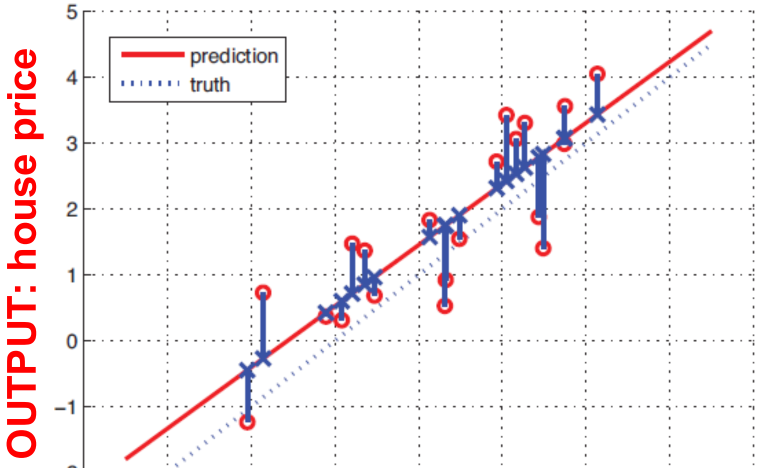


Test error: average of $\ell(y, f(x))$ in test set

- Goal: design learning algorithm $\mathcal{A}$, such that after training, its output predictor f has low test error

Supervised Learning: Linear Regression

Linear Regression



INPUT: square footage

Regression Learn a function that predicts outputs from inputs,

$$y = f(x)$$

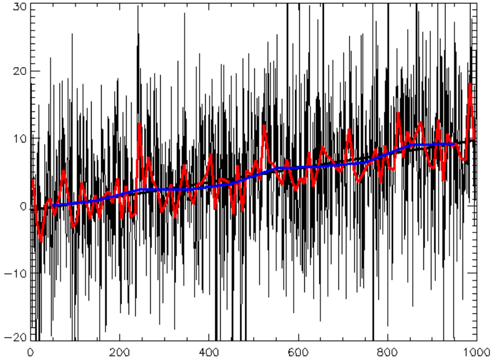
Outputs y are real-valued

Linear Regression As the name suggests, uses a *linear function*:

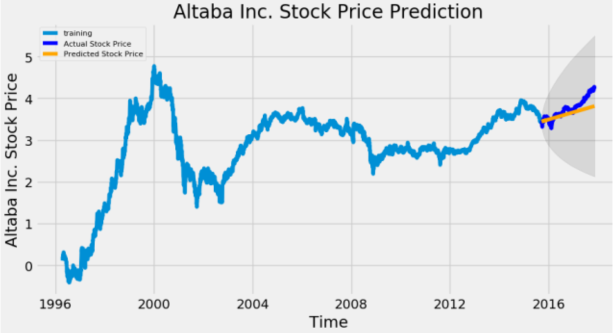
$$y = w^T x + b$$

Linear Regression

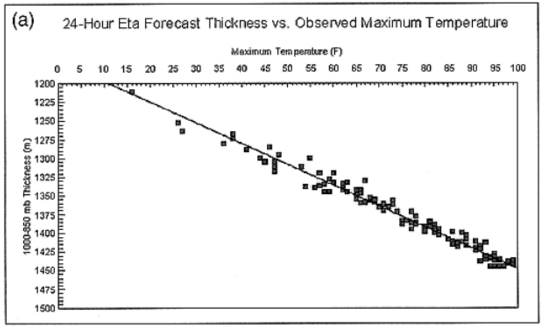
Where is linear regression useful?



Trendlines



Stock Prediction

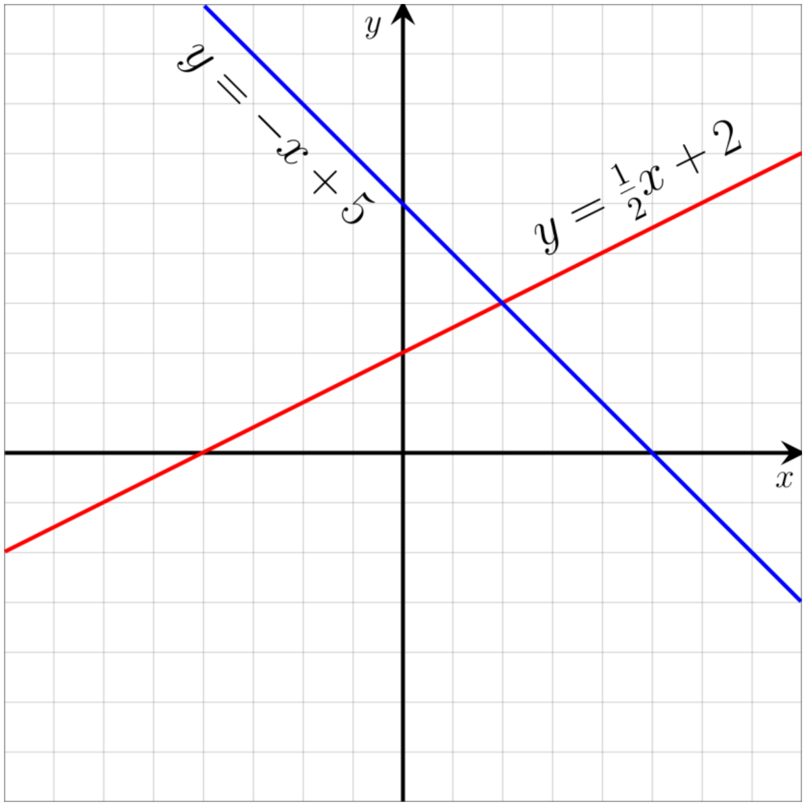


Climate Models

Massie and Rose (1997)

Used anywhere a linear relationship is assumed between continuous inputs / outputs

Line Equation



Recall the equation for a line has a *slope* and an *intercept*,

$$y = \underset{\text{Slope}}{\underset{\uparrow}{w}} \cdot x + \underset{\text{Intercept}}{\underset{\uparrow}{b}}$$

- Intercept (b) indicates where line crosses y-axis
- Slope controls angle of line
- Positive slope (w) → Line goes up left-to-right
- Negative slope → Line goes down left-to-right

Math Interlude: inner product

Two vectors:

$$\vec{x} = \langle 2, -3 \rangle \qquad \mathbf{x} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

$$\vec{y} = \langle 5, 1 \rangle \qquad \mathbf{y} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

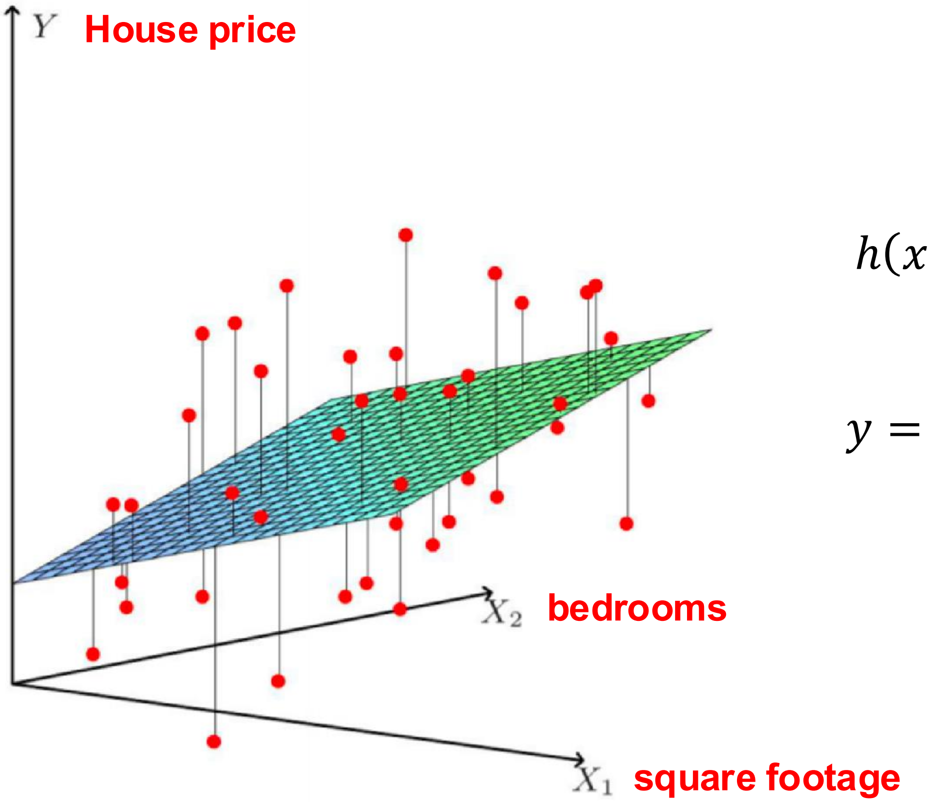
How to compute $\vec{x} \cdot \vec{y}$?

Multiply corresponding entries and add:

$$\vec{x} \cdot \vec{y} = \langle 2, -3 \rangle \cdot \langle 5, 1 \rangle = (2)(5) + (-3)(1) = 7$$

$$\mathbf{x}^T \mathbf{y} = \begin{bmatrix} 2 & -3 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \end{bmatrix} \quad \text{(or just 7)} \quad (\text{so } \vec{x} \cdot \vec{y} \text{ becomes } \mathbf{x}^T \mathbf{y})$$

Linear regression in dimension more than 2



$$h(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + b = w \cdot x + b$$

$y = w \cdot x + b$ is a hyperplane

Linear Regression

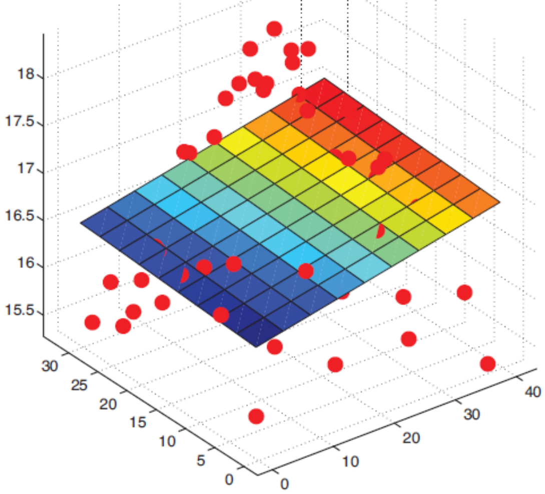
For D-dimensional input vector $x \in \mathbb{R}^D$ the plane equation,

$$y = w^T x + b$$

Sometimes we simplify this by including the intercept into the weight vector,

$$\widetilde{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_D \\ b \end{pmatrix} \qquad \widetilde{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{pmatrix} \qquad y = \widetilde{w}^T \widetilde{x}$$

[Image: Murphy, K. (2012)]

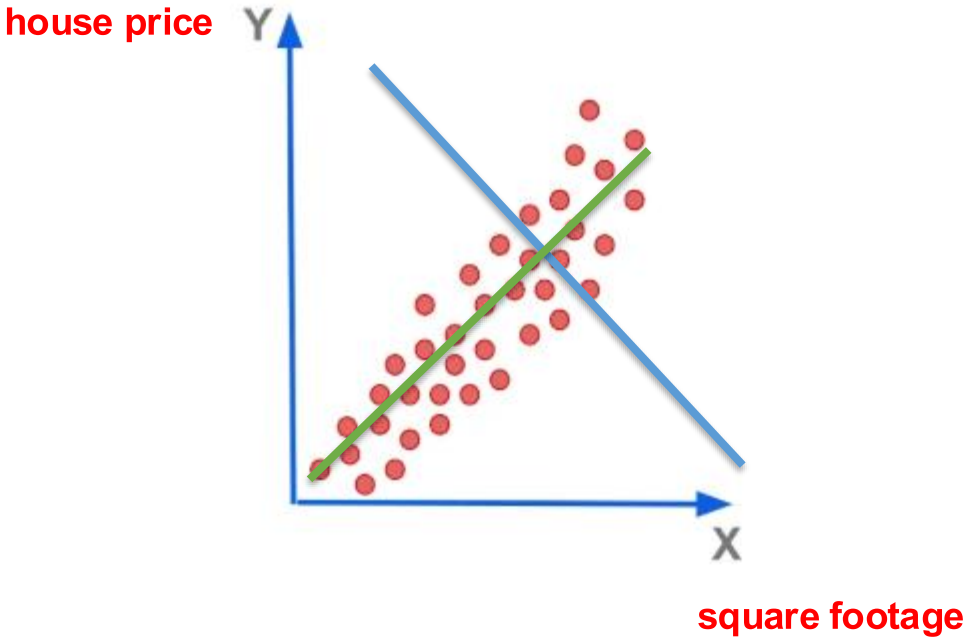


Since:

$$\widetilde{w}^T \widetilde{x} = \sum_{d=1}^{D} w_d x_d + b \cdot 1$$
$$= w^T x + b$$

Learning Linear Regression Models

- Which line is a better predictor, blue or green?



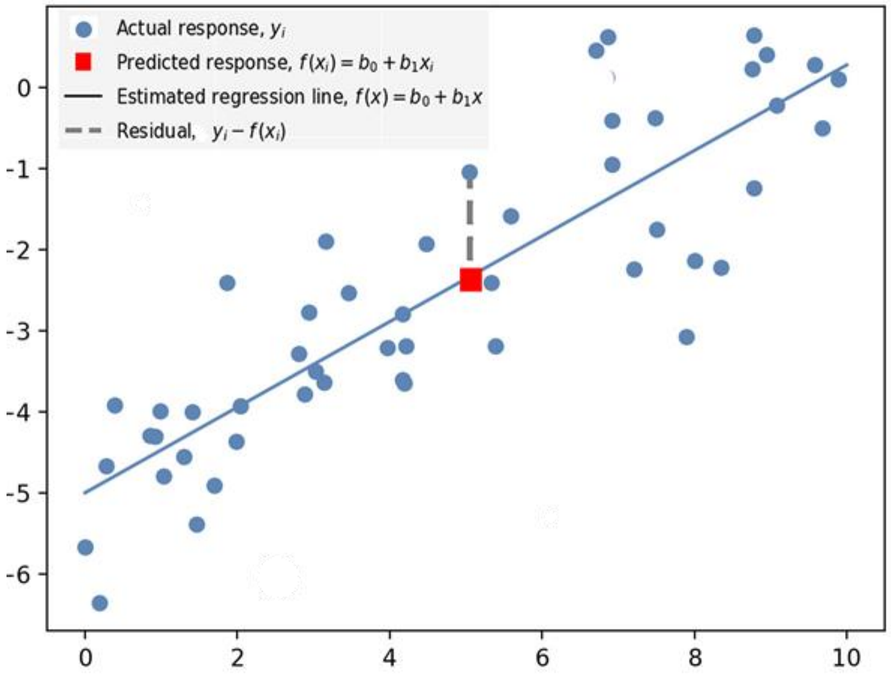
- green

Learning Linear Regression Models

There are at least two ways to think about fitting regression:

- **Intuitive** Find a plane/line that is close to data
- **Functional** Find a line that minimizes the *square* loss

Fitting Linear Regression



Intuition Find a line that is as *close as possible* to every training data point

The distance from each point to the line is the **residual**

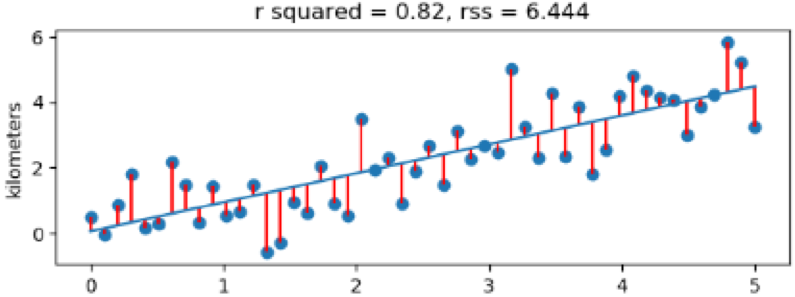
$$y - w^T x$$

Label
Prediction

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

Fitting linear regression

- Each point i induces a separate residual value $y_i - w \cdot x_i$



- We'd like to find w such that all $y_i - w \cdot x_i$ are small
- We can convert this to an optimization problem: find w that minimizes

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

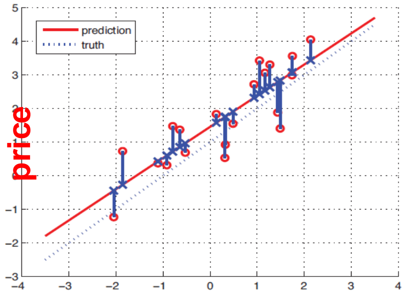
- This is called the ***least squares solution***

Math Interlude: optimization problems

$$\operatorname{argmin}_{w \in \mathrm{R}^d} \sum_{i=1}^{n} (y_i - w \cdot x_i)^2$$

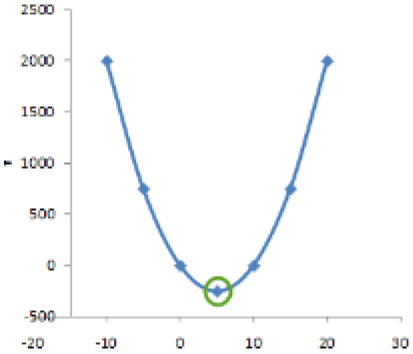
w: Optimization variable

Objective function



OUTPUT: house price

INPUT: square footage



- **Example** Find $\operatorname{argmin}_x ax^2 + bx + c \; (a > 0)$
  - $x = -\frac{b}{2a}$

Math Interlude: optimization problems

Example Suppose we have 2 data points (x=1, y=0) and (x=-1, y=1), we fit $y = w\,x$ without intercept, find the least squares solution $\hat{w}$

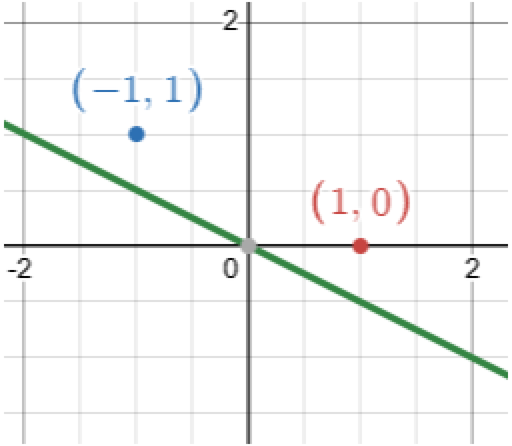
Solution the objective function of least squares is

$$(y_1 - w\,x_1)^2 + (y_2 - w\,x_2)^2$$

which is

$$w^2 + (1 + w)^2 = 2w^2 + 2w + 1$$

the minimizer is $\hat{w} = -\frac{b}{2a} = -\frac{1}{2}$



Why cannot the line fit perfectly?

- Here, we only consider $y = w\,x$ without intercept

In-class exercise: training and test loss

- We have the following training data

Study hours (x)	Exam score (y)
1	2
3	6

- Q1: We fit a linear regression model $y = w \cdot x$ that minimizes mean square error. What is this model $\hat{w}$?

- Q2: What is the average loss of model $\hat{w}$ on training and test data?

Study hours (x)	Exam score (y)
4	7
5	10

In-class exercise: training and test loss

Solution

$\widehat{w} = \mathrm{argmin}_w \, (1w-2)^2 + (3w-6)^2$

Minimizer: $\widehat{w} = -\frac{b}{2a} = 2$

$10w^2 - 40w + 40$

Study hours (x)	Exam score (y)
1	2
3	6

Training loss of $\widehat{w}$:

$$\frac{1}{2}\left((2-2)^2 + (6-6)^2\right) = 0$$

size of training set

Study hours (x)	Exam score (y)	Predicted score
1	2	2
3	6	6

Test loss of $\widehat{w}$:

$$\frac{1}{2}\left((8-7)^2 + (10-10)^2\right) = 0.5$$

size of test set

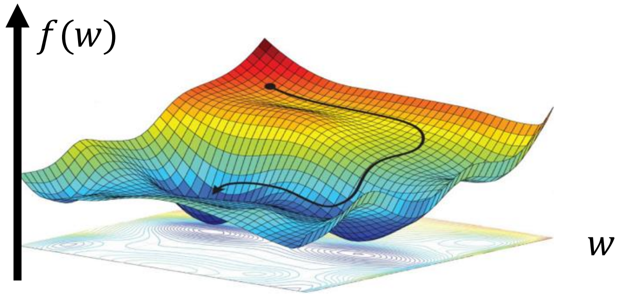
Study hours (x)	Exam score (y)	Predicted score
4	7	8
5	10	10

Usually, a trained model has smaller training loss than test loss

Math Interlude: optimization problems

- Unconstrained optimization problem: find

$$\text{argmin}_{w \in \mathrm{R}^d} f(w)$$

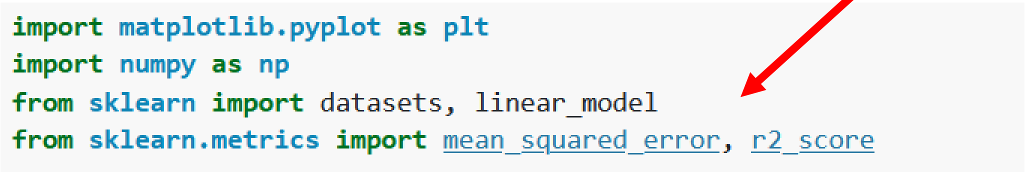


- Solutions can oftentimes be found in one of two ways:
  1. Closed form solutions
  2. Open-source or commercial optimization libraries (e.g. `cvxpy`, `scipy.optimize.minimize`)

Linear Regression in Scikit-Learn

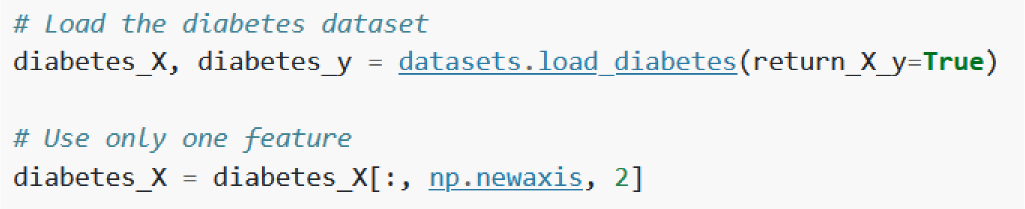
Load your libraries,

For Evaluation

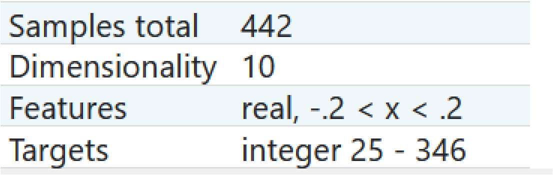


```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

Load data,

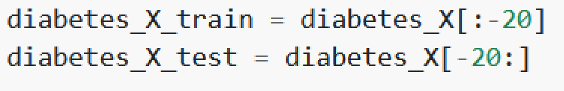


```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]
```

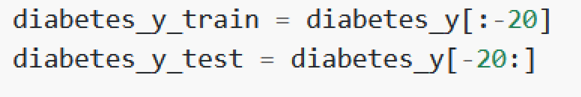


Samples total	442
Dimensionality	10
Features	real, -.2 < x < .2
Targets	integer 25 - 346

Train / Test Split:



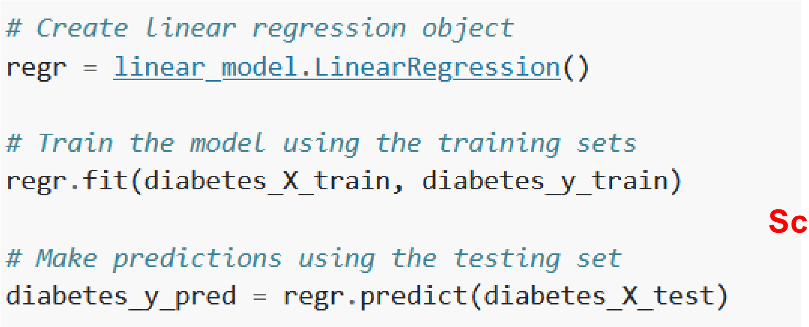
```
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
```



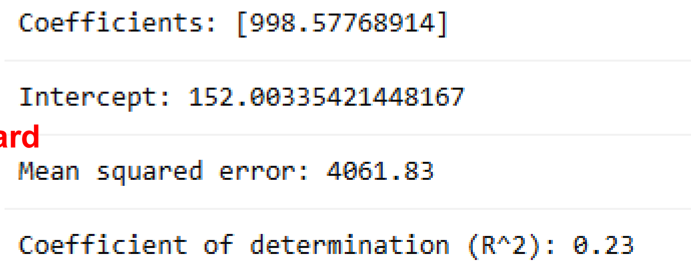
```
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

Linear Regression in Scikit-Learn

Train (fit) and predict,



```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

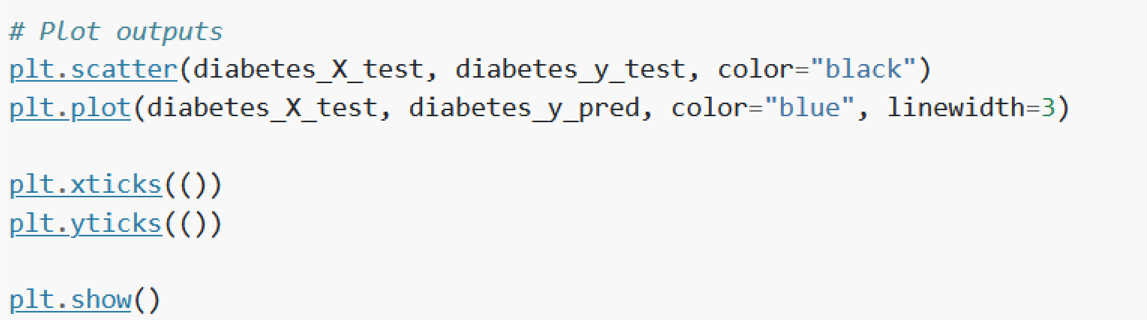


```
Coefficients: [998.57768914]

Intercept: 152.00335421448167

Mean squared error: 4061.83

Coefficient of determination (R^2): 0.23
```

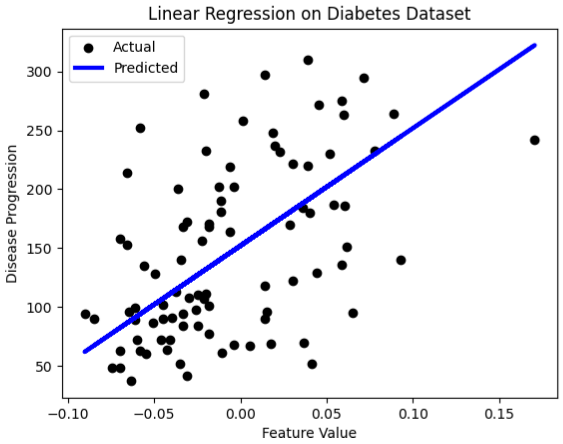
Scale sensitive, a bit hard to interpret ->

More interpretable->

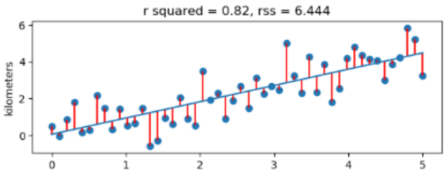
Plot regression line with the test set,



```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



Coefficient of Determination R^2



Variance unexplained by Regression model

Residual Sum-of-Squares

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{SS}} = 1 - \frac{\sum_{i=1}^{N}(y_i - w^T x_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

Total variance in dataset

Variance using avg. prediction

Where: $\bar{y} = \frac{1}{N}\sum_i y_i$ is the average output

R^2 represents the proportion of the variance in y that is predictable from a x_i.

Coefficient of Determination R^2

$$R^2 = 1 - \frac{\text{RSS}}{\text{SS}}$$

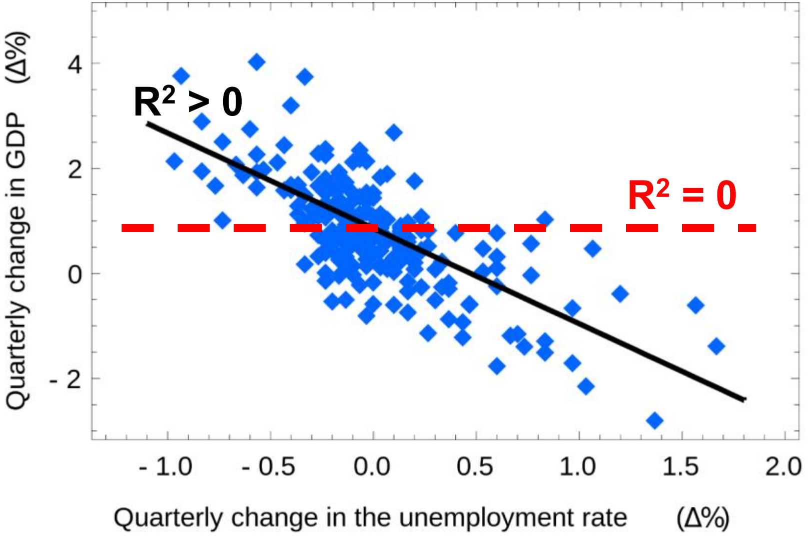
Variance unexplained by Regression model

Variance using avg. prediction

Maximum value R^2=1.0 means model explains *all variation* in the data

R^2=0 means model is as good as predicting average response

R^2<0 means model is worse than predicting average output (rare)



Overfitting and underfitting

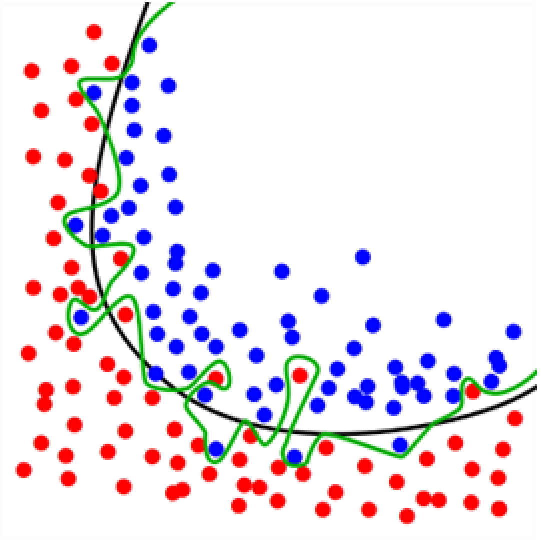
Challenge in machine learning: generalization

Why not learn the most complex predictor that can work flawlessly for the training data and be done with it? (i.e., predicts every training data point correctly)

Problem: may not generalize to unseen data – called *overfitting* the training data.

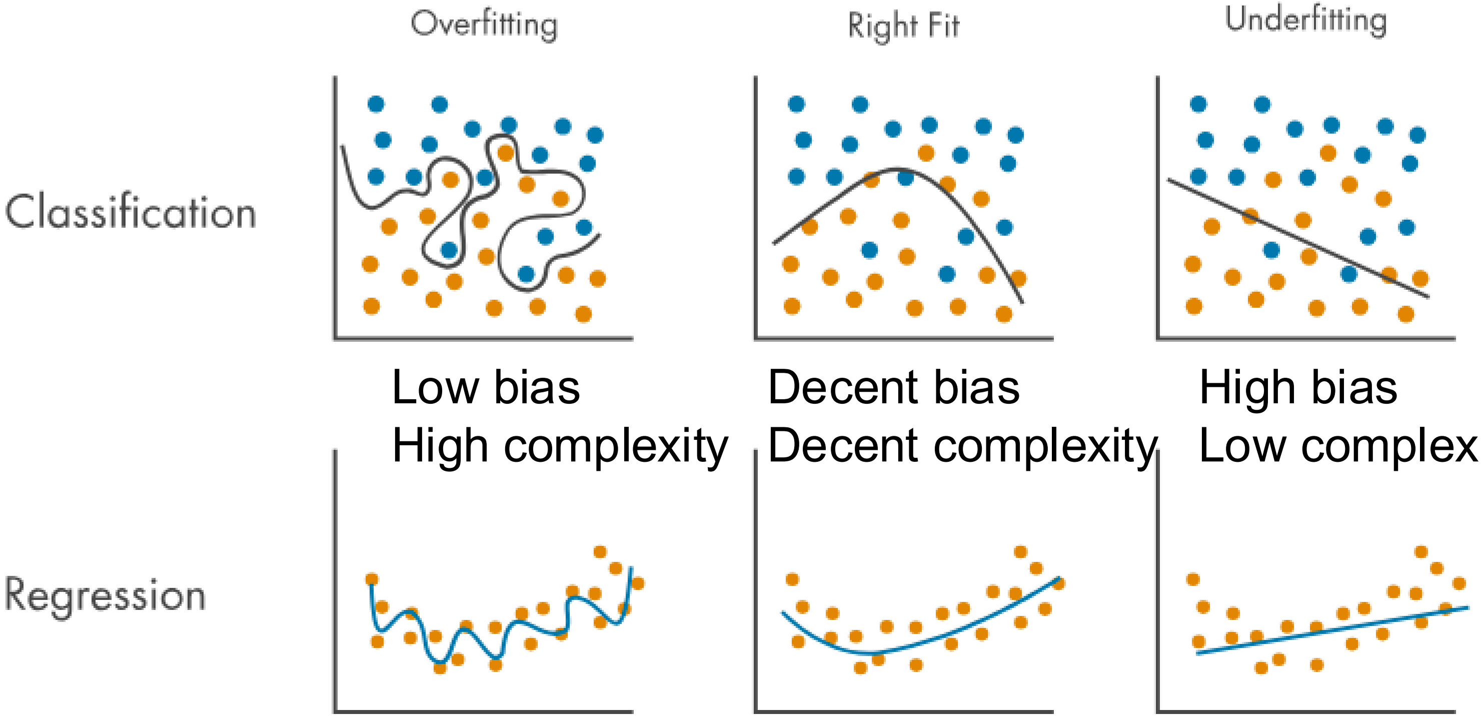
In other words, memorization is not generalization

Mitigation: Fit the training set but don't "over-do" it -- **regularization**.



green: may be sensitive to noise in training data
black: more robust and can generalize better

Overfitting and Underfitting



Low bias
High complexity

Decent bias
Decent complexity

High bias
Low complexity

Ideal: select a model that trades off bias & complexity, i.e.,

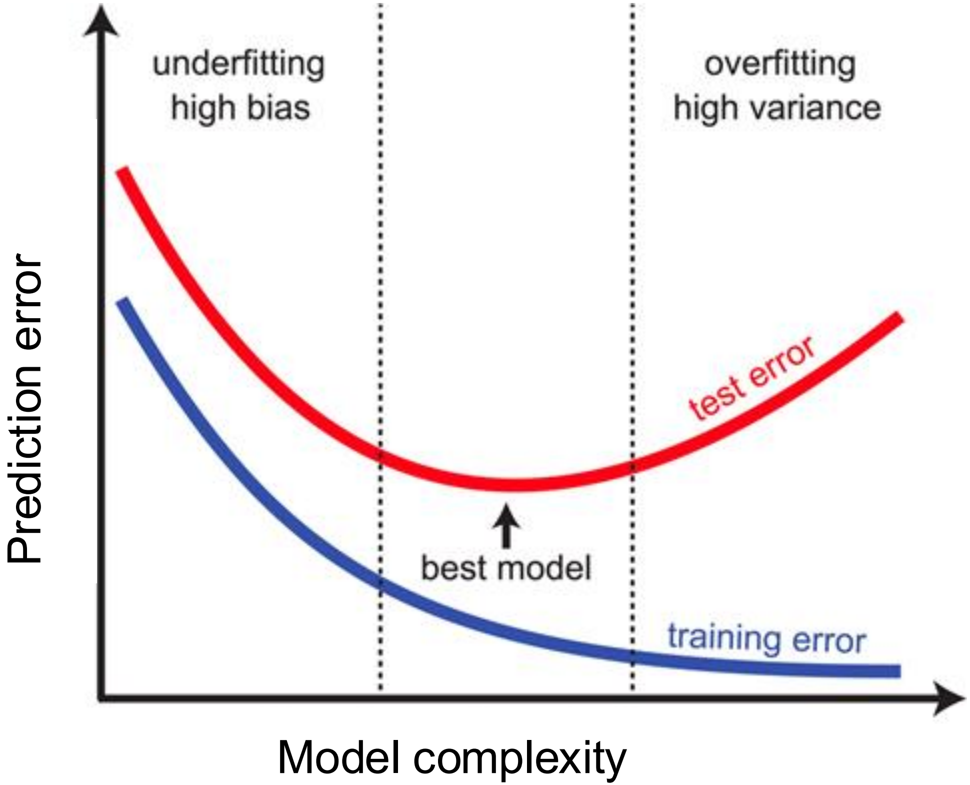
- sophisticated enough to capture meaningful patterns for accurate predictions,
- yet not so intricate that it overfits the data. **Low complexity** **Low bias**

Model selection

Examples of model complexity:

- The number of features used for prediction (more features => more complex)
- The weight of the predictors used for prediction (higher weight => more complex)

Model selection: choosing model with “just right” complexity for data


underfitting
high bias
overfitting
high variance
Prediction error
test error
best model
training error
Model complexity