Big Data and Machine Learning [1]

Jasmine(Yu) Hao

Faculty of Business and Economics
Hong Kong University

October 8, 2021

[1] This section is based on Stock and Watson (2020), Chapter 14. For the reference of machine learning programming, check Géron (2019).

Outline

- Outline
- Big data and inference at scale
- Basic terminology of machine learning
- Fundamental issues in statistical predictions
- Regression in the prediction framework
- Model selection and regularization

Statistics vs. data science

- Statistics means the practice or science of collecting and analyzing numerical data in large quantities.
- Data Science means the practice of liberating and creating meaning from data using scientific methods.[2]
- Is there really a difference? Isn't data science just applied statistics?

- When the data is big, the traditional methods of statistical inference are neither sufficient nor necessary for many practical purposes.

[2] D. Donoho "50 Years of Data Science"

The Big Data Revolution I

The Big Data Revolution II

What is big data? I

- Origin of "big data"
  - The Big Data name comes from computer scientists working to do aggregation on data that is too big to fit on a single machine
- Big Data is focused on actionable knowledge extraction from very large datasets (integral in business and industrial applications).
  - Pattern discovery: infer patterns from complex high dimensional data
  - Data mining: simplicity and scalability of algorithms is essential
  - Decision making: the end product is an decision
  - Emphasis on usefulness: infer useful signal at massive scale for decision making
- What does it mean to be "big"?
  - Big in both the number of observations (size 'n') and in the number of variables (dimension 'p')
  - Big data is not just about the volume of data, but more about the new methods that are developed to handle big data.

How much statistics do we need for data science?

"Statistical thinking not only helps make scientific discoveries, but it quantifies the reliability, reproducibility and general uncertainty associated with these discoveries. Because one can easily be fooled by complicated biases and patterns arising by chance, and because statistics has matured around making discoveries from data, statistical thinking will be integral to Big Data challenges."

Discovery with Data: Leveraging Statistics with Computer Science to Transform Science and Society (ASA White Paper 2014)

- With the help of computer science, we aim to summarize really high dimensional data in such a way that we can relate it to structural models of interest. Ultimately, we are doing statistics!

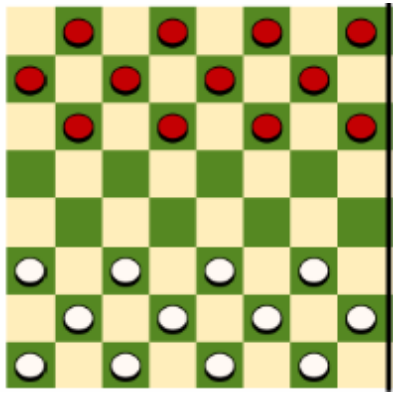
What is machine learning? I

Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.

Tom Mitchell (1998): a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Experience (data): games played by the program (with itself)

Performance measure: winning rate



What is machine learning? II

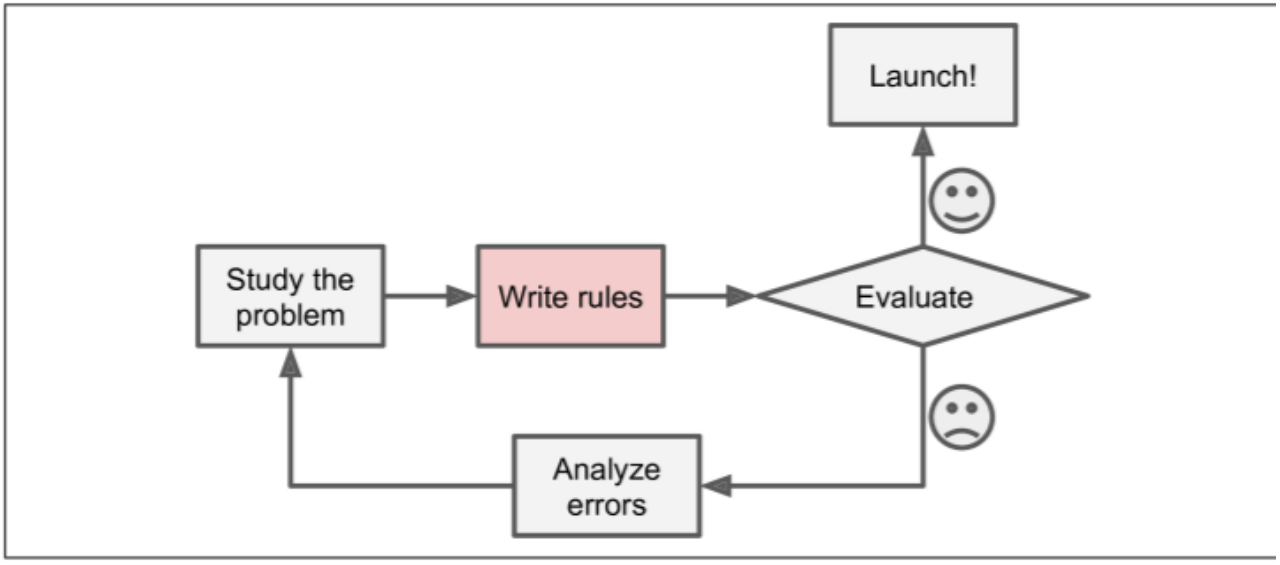


Figure 1-1. The traditional approach

What is machine learning? III

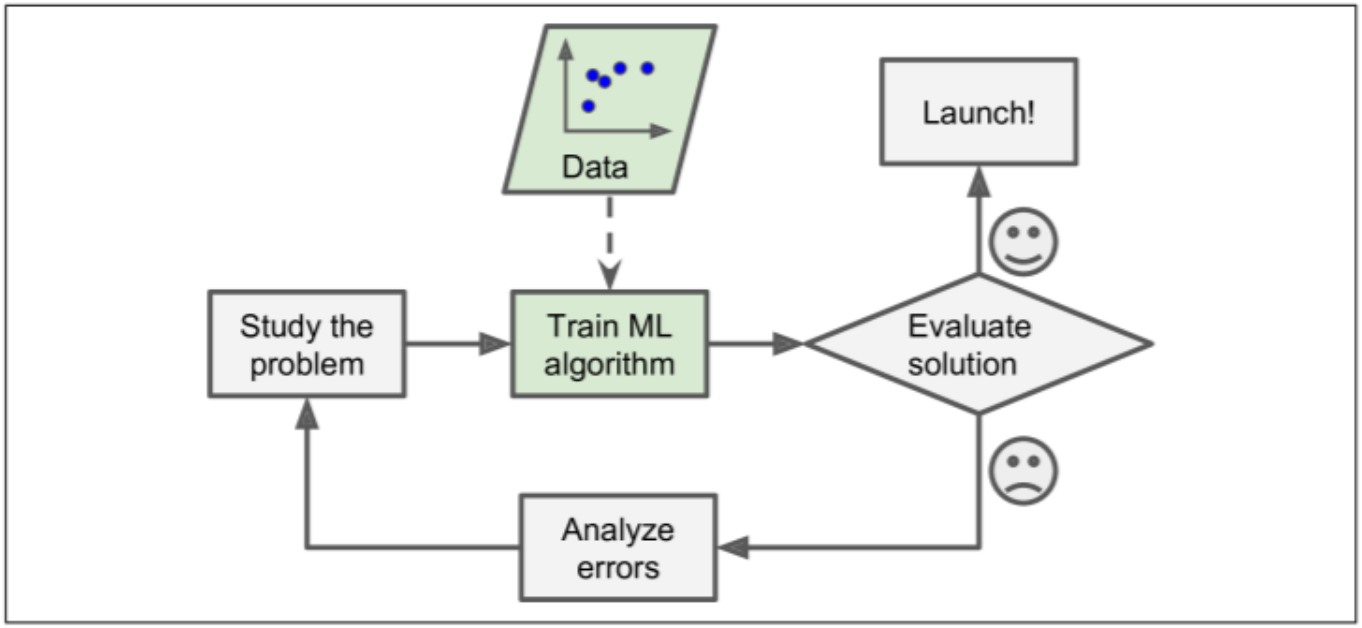


Figure 1-2. Machine Learning approach

What is machine learning? IV

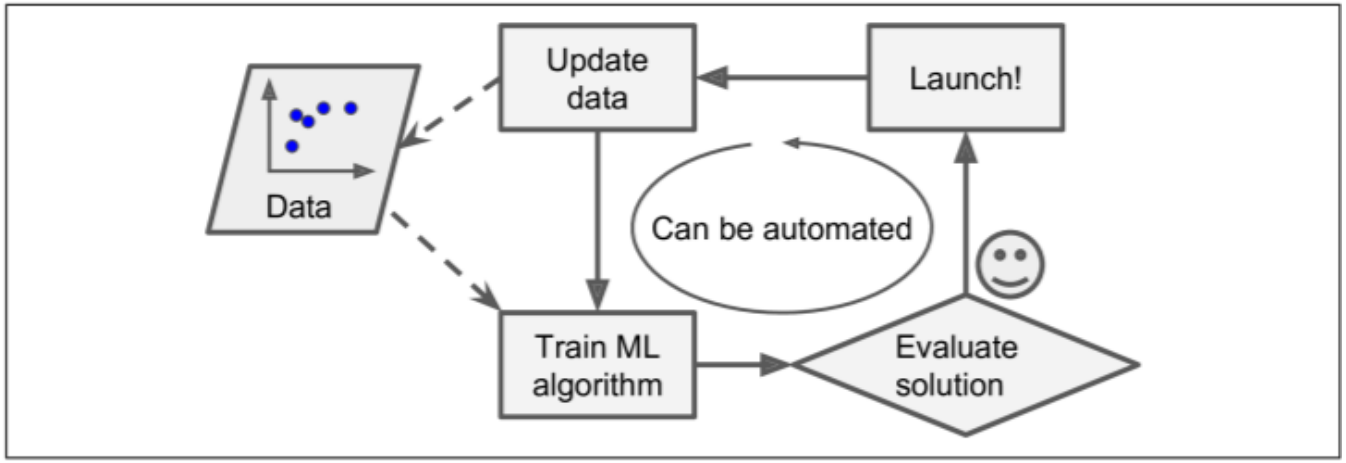


Figure 1-3. Automatically adapting to change

What is machine learning? V

To summarize, Machine Learning is great for:

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better.
- Complex problems for which there is no good solution at all using a traditional approach: the best Machine Learning techniques can find a solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

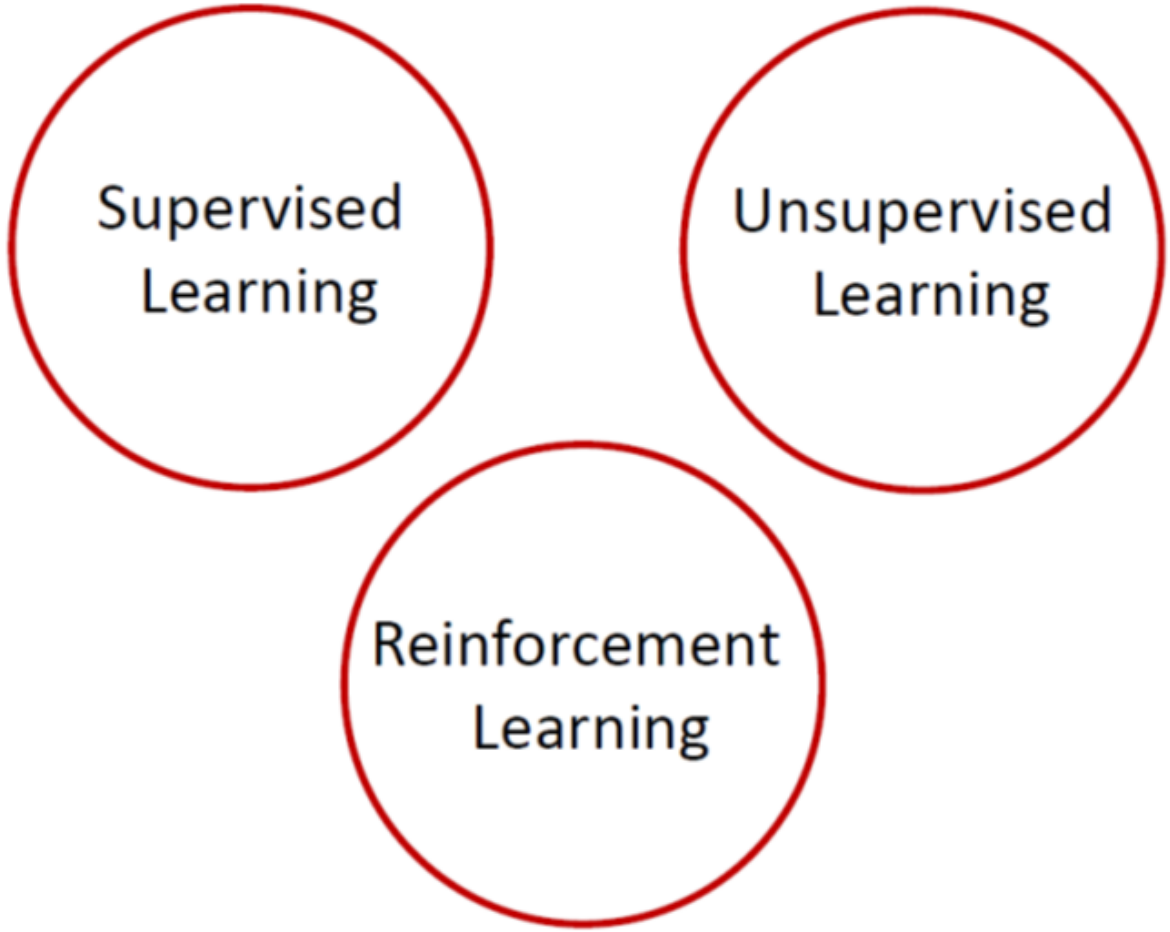
Machine learning vs. statistical learning

- Machine learning constructs **algorithms** that can learn from data , especially for **prediction**
- Statistical learning is a branch of Statistics that was developed in response to Machine learning, emphasizing building statistical models, drawing **inferences** and assessing uncertainty

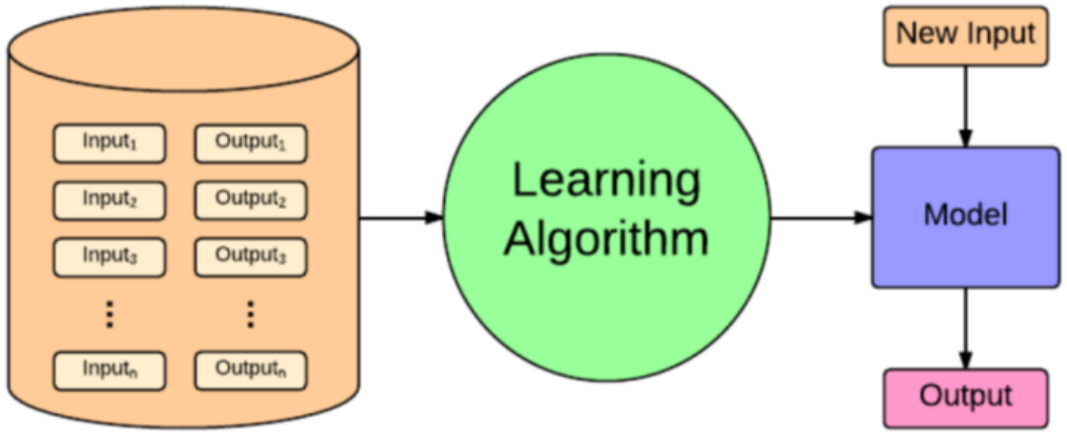
Econometrics → Statistics
→ Data Mining / Big Data / Data Science
→ Machine Learning (ML) and Artificial Intelligence (AI)

▷ Along this spectrum, you move from heavy focus on measuring and inferring the truth to a more pragmatic 'useful is true' pattern discovery approach.

Taxonomy of Machine Learning



Supervised learning



- Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output: $Y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (Y) for that data.

Important Supervised Learning Algorithms

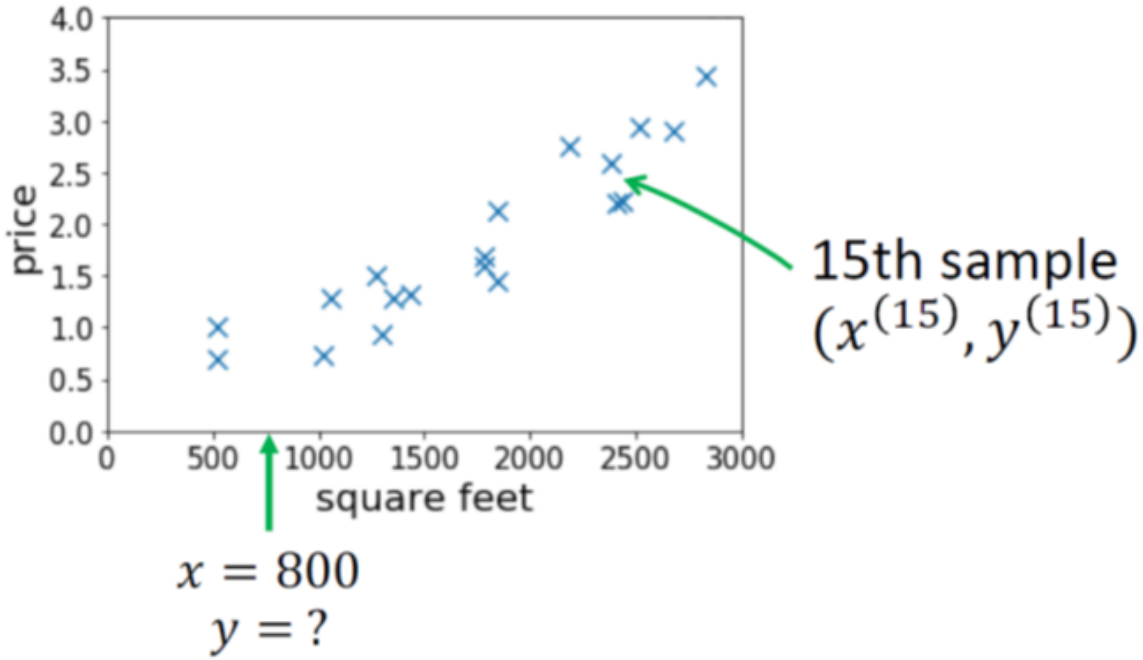
- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

Example of supervised learning: housing price prediction

Given: a dataset that contains n samples

$$(x^{(1)}, y^{(1)}), \ldots (x^{(n)}, y^{(n)})$$

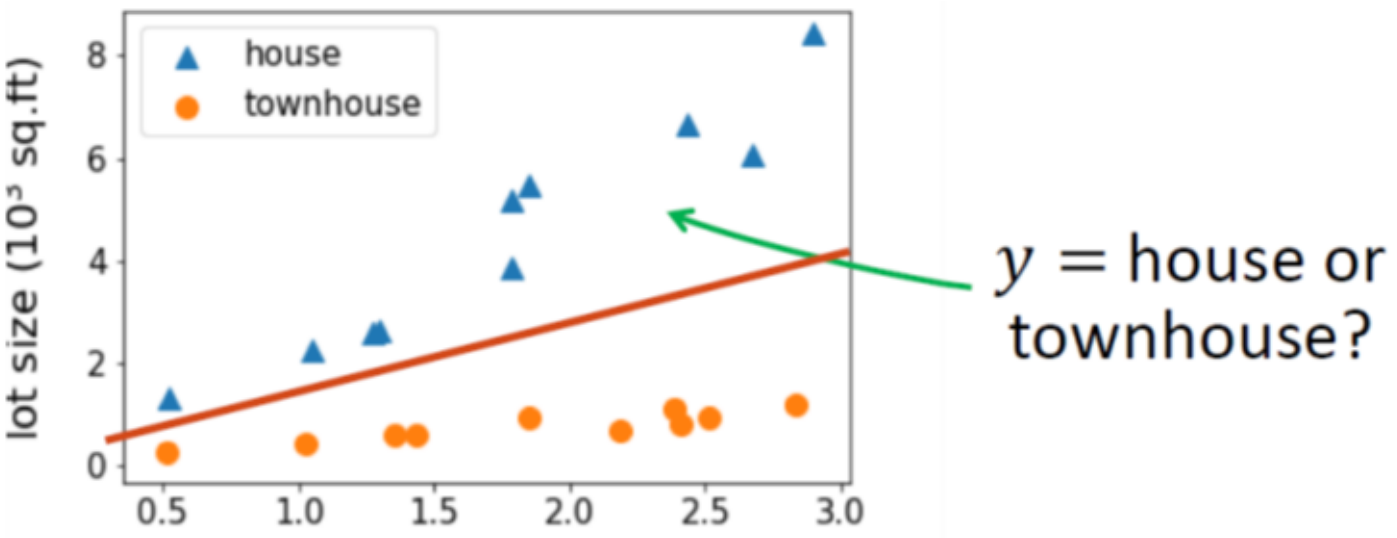
- Task: if a residence has x square feet, predict its price?



Regression vs. classification

- regression: if $y \in \mathbb{R}$ is a continuous variable
  - e.g., price prediction
- classification: the label is a discrete variable
  - e.g., the task of predicting the types of residence

(size, lot size) → house or townhouse?

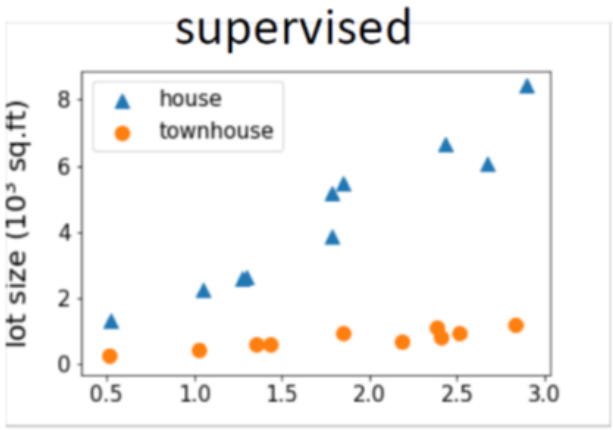


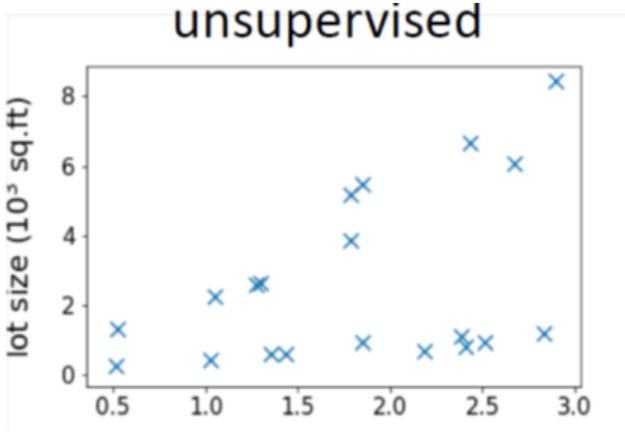
Supervised learning in computer vision

- Image Classification
  - x = raw pixels of the image, y = the main object

Unsupervised learning

- Dataset contains no labels: $x^{(1)}, \ldots x^{(n)}$
- Goal (vaguely-posed): to find interesting structures in the data



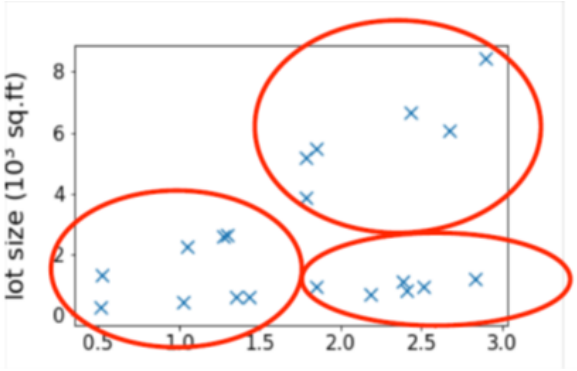


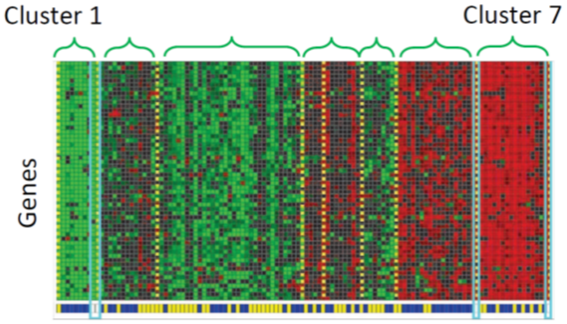
Important Supervised Learning Algorithms

- Clustering
  - K-Means
  - Hierarchical Cluster Analysis (HCA)
- Anomaly detection and novelty detection
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)

Clustering

- Clustering genes

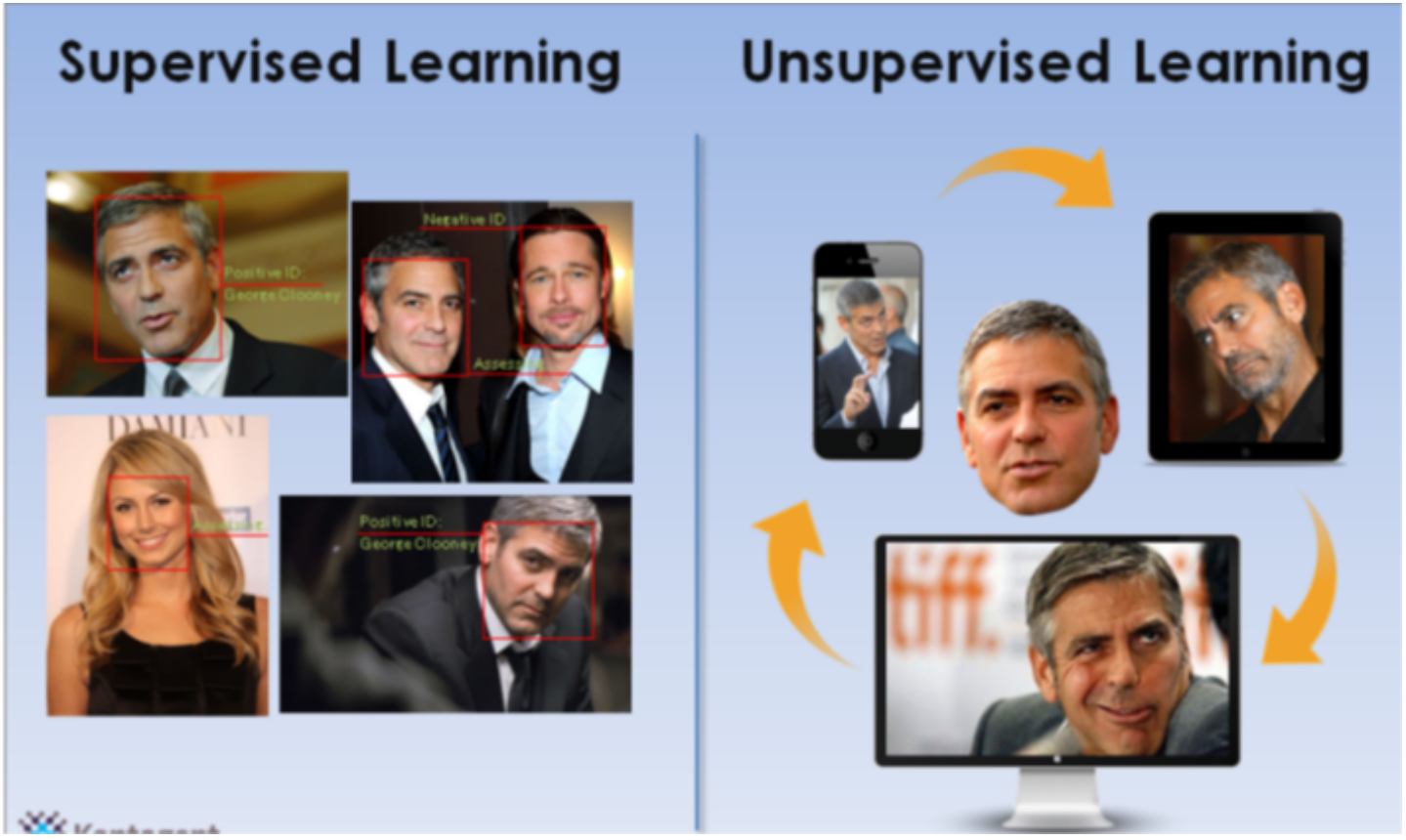




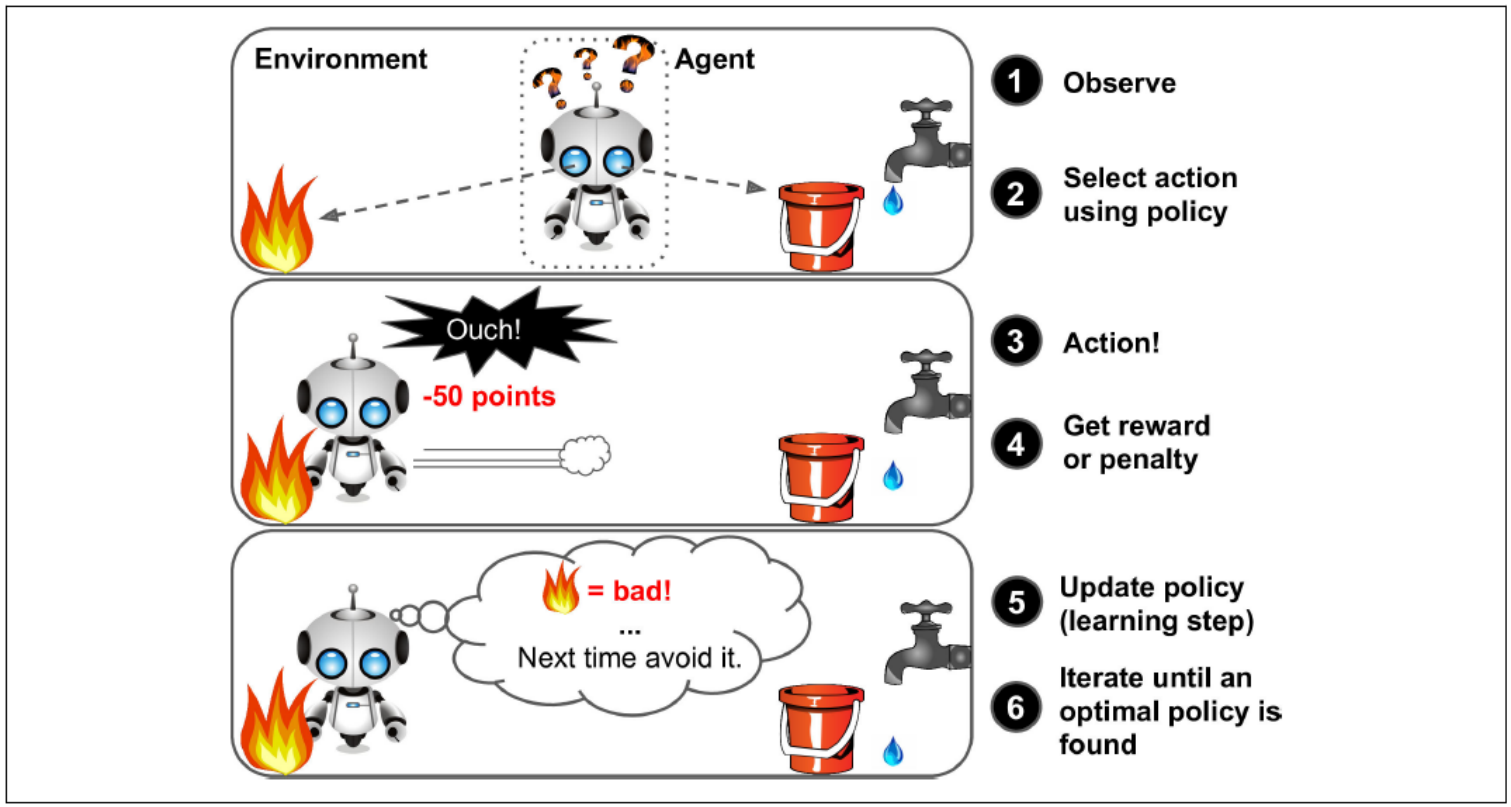
- Clustering houses

Supervised vs. unsupervised learning

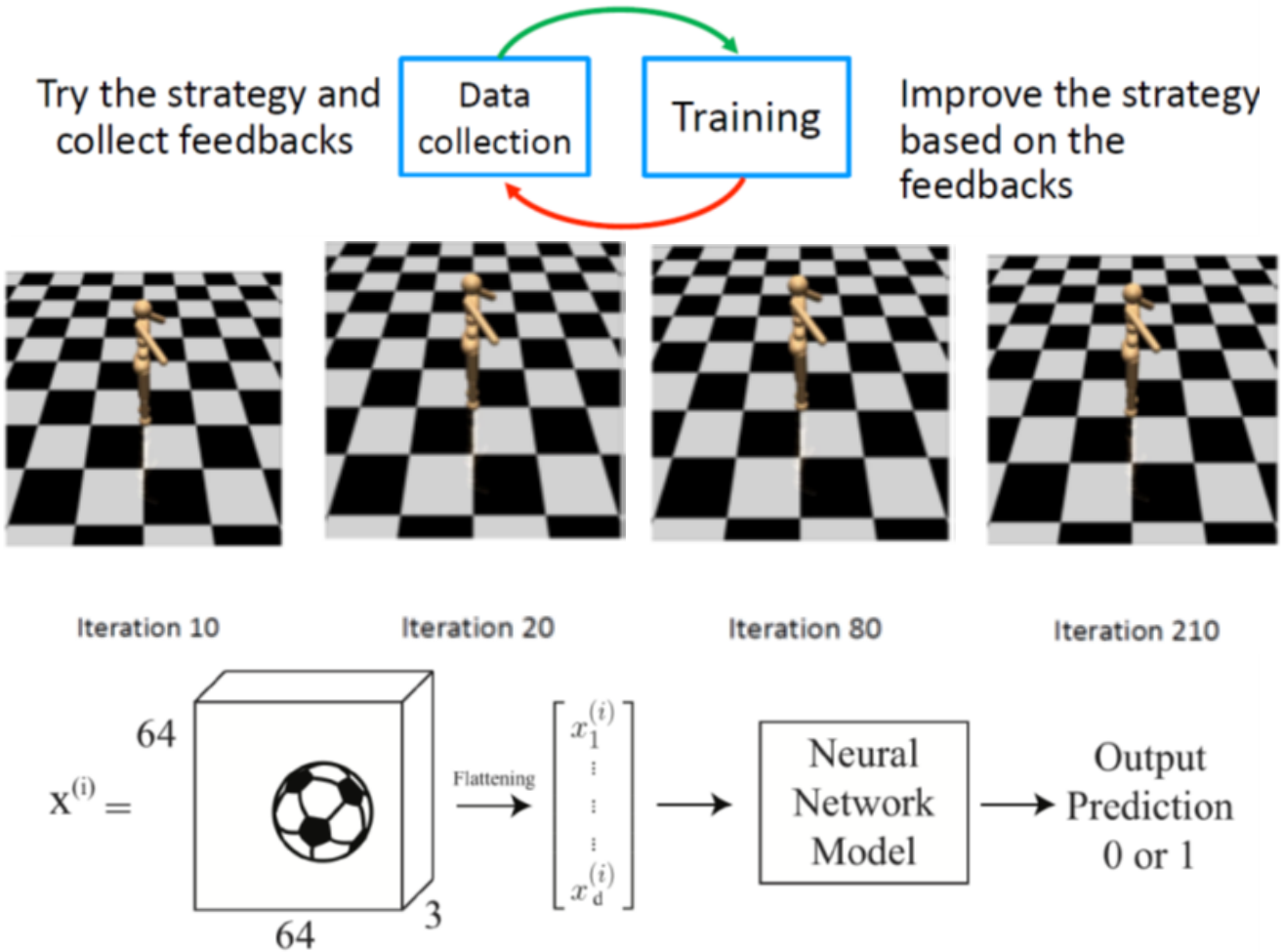
- **Supervised**: Both inputs (features, a.k.a. covariates) and outputs (labels, a.k.a. response) in training set.
- **Unsupervised**: No output values available, just inputs.



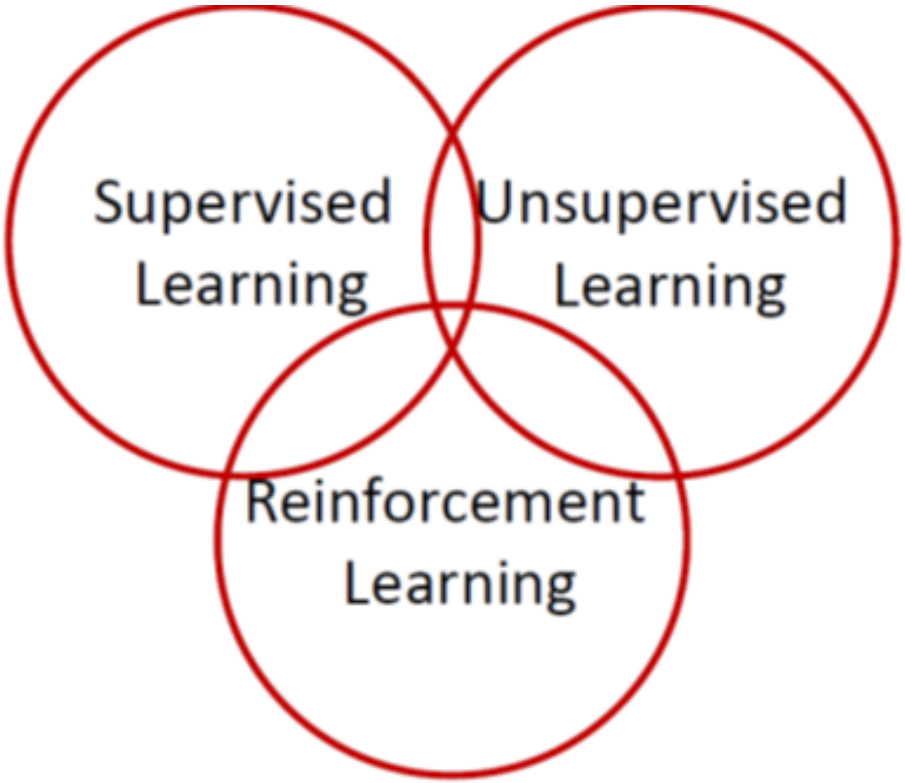
Reinforcement learning I



Reinforcement learning II



Task-based machine learning



Traditional statistics: domain experts work for many years to learn good **features**; they bring statisticians a small clean dataset
Today's scenario: Domain knolwedge is limited in new fields and large data sets are readily available. we (are forced to) start with a large dataset with many features

Data structure

- Structured vs. unstructured data
  - Structured data: a flat file with a fixed number of measurement, e.g., a patient response to a drug under consideration of certain conditions(such as age, weight, size, nutrition intake)
  - Unstructured data: doctor's notes, Twitter feeds, broker reports
- Training data vs. test data
  - Training data: the data used to fit a model or train an algorithm so as to make the model/algorithm able to predict outcomes.
  - Test data: the data independent of the training data but following the same probability distribution as the training dataset. Test data are not available unless the probability distribution is known.
  - Holdout data: part of the original dataset that is set aside and used to test the performance of the model/algorithm obtained from the training data

Statistical prediction revisited

- We assume a functional relationship between X and Y.
- Accuracy of the estimate depends on two types of errors:
- Reducible error which can be potentially reduced by improving the estimation of f(X)
- Irreducible error which is caused by the random errors that cannot be used to predict Y.
- $f(\cdot)$ represents the systematic information that X provides about Y.

$$Y = f(x) + \epsilon$$

- We aim to estimate the function to make predictions:

$$\hat{Y} = \hat{f}(x)$$

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible}}, \end{aligned}$$

Assess estimation accuracy

- General idea: goodness of fit (GOF)
- Define a way to measure errors: $e_i = y_i - \hat{y}_i$
- Define a loss function: $Loss(y, \hat{y})$
- Set a criterion to judge the accuracy of an estimator

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

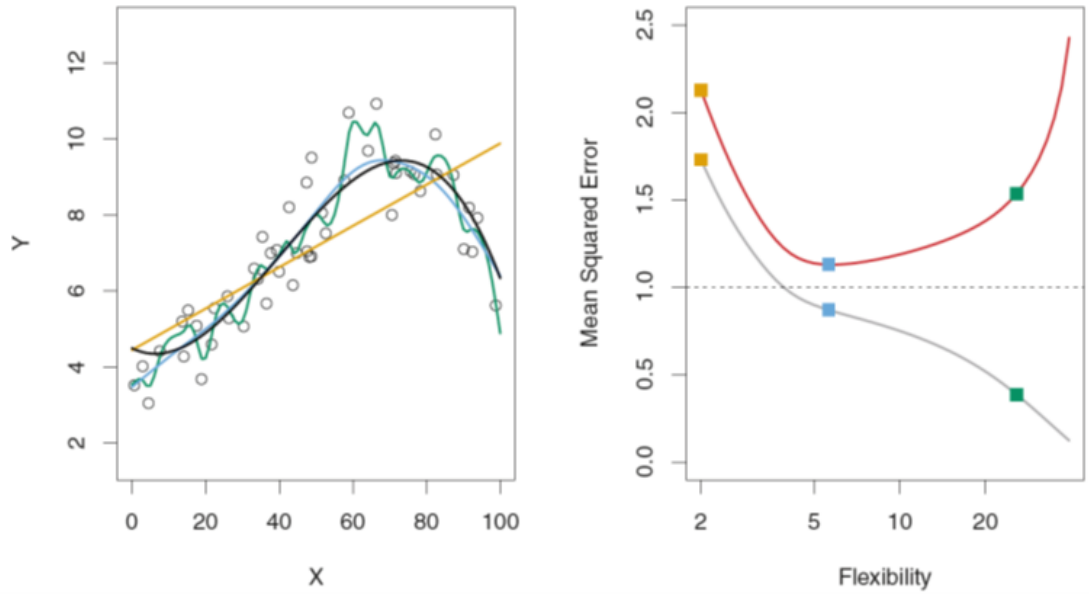
$$MSE = \frac{1}{n} \sum i = 1^n (y_i - \hat{f}(x_i))^2$$

- MSE-standard (mean squared error)
- But for prediction purposes, we are not really interested in the MSE in the training (observed) data. We want to know the error in the test (unseen) data that are not used to train the estimator.

$$AVE(y_0, -\hat{f}(x_0))^2$$

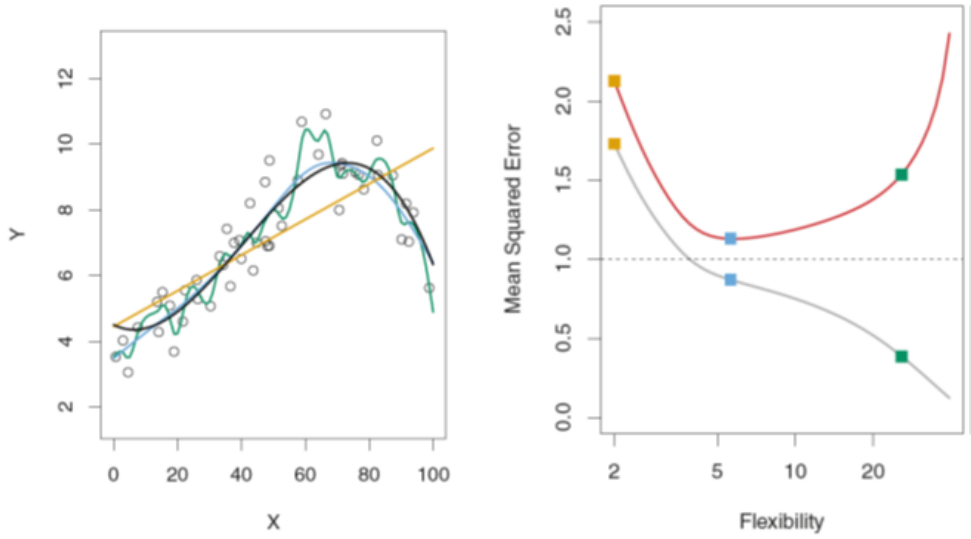
MSE in training and test data

FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand*



Fundamental problem in statistical learning: overfitting

- Overfitting the data
- Occurs when a method yields a small training MSE but a large test MSE
- Caused by the learning procedure trying too hard to follow the patterns in the training data and pick up too much random chance rather than the true properties of the unknown function.
- The cost of being too flexible: a flexible function with many degrees of freedom is used to better fit the data



The bias-variance tradeoff

- The expected test MSE can be decomposed into three fundamental quantities:

$$E(y_0 - \hat{f}(x_0)) = \text{var}(\hat{f}(x_0) + [Bias(\hat{f}(x_0))] + \text{var}(\epsilon)$$

- Variance: the amount by which f(.) would change if we estimate it using a different training data set. If a method has high variance, small changes in the training data can result in large changes in the estimate of f(.).
- Bias: the error that is introduced by approximating a (complicated) real-life problem by a much simpler model. For example, using a linear model to approximate a non-linear relationship can lead to a large bias.
- Tradeoff: when a more flexible method is used, the bias will decrease and the variance will increase. This tradeoff governs the amount of test MSE.
- At some point, increasing flexibility has little impact on the bias but starts to significantly increase the variance.

How to assess test MSE?

- The expected test MSE can be decomposed into three fundamental quantities:

$$E(y_0 - \hat{f}(x_0)) = \text{var}(\hat{f}(x_0) + [Bias(\hat{f}(x_0))] + \text{var}(\epsilon)$$

- In theoretical study, we generate both training and test data from a known distribution. It is easy to calculate the test MSE, bias, or variance for a statistical learning method.
- However, in a real-life situation, the underlying distribution function is unobserved, it is not possible to explicitly compute the test MSE, bias, or variance for a method.
- We have to estimate these quantities by cross-validation, with which we assess a method using holdout samples in the training data.

Regression

- Many problems in BD involves a response of interest (y) and a set of covariates (x).
- A general tactic is to deal in averages and lines.
- We will model the conditional mean for y given x

$$E[y|x] = f(x^\top \beta)$$

- $x = [1, x_1, \ldots, x_p]$ is your vector of covariates
- $\beta = [\beta_0, \ldots, \beta_p]$ are corresponding coefficients.
- The product is $x^\top \beta = \beta_0 + x_1\beta_1 + \ldots + x_p\beta_p$.

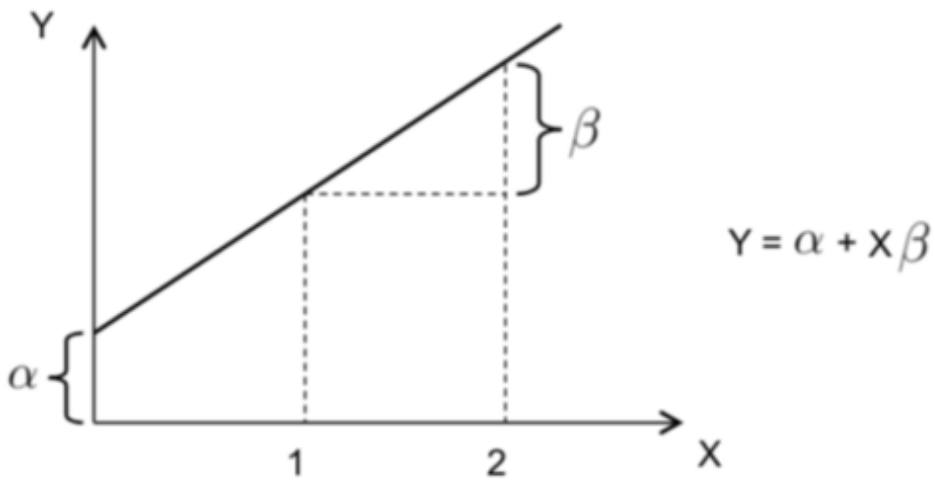
Linear Regression

In a **Gaussian** linear regression,

$$y \mid \mathbf{x} \sim \mathrm{N}(\mathbf{x}'\boldsymbol{\beta}, \sigma^2)$$

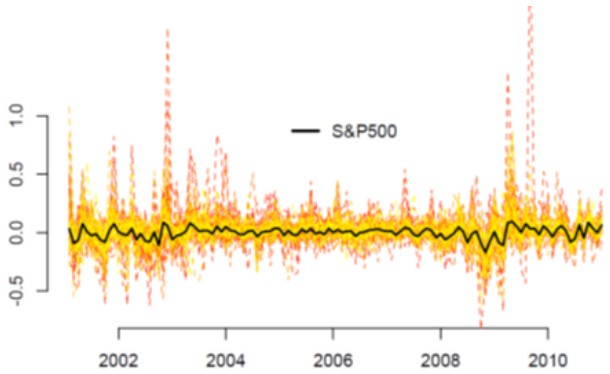
Conditional mean is $\mathbb{E}[y|\mathbf{x}] = \mathbf{x}'\beta$.

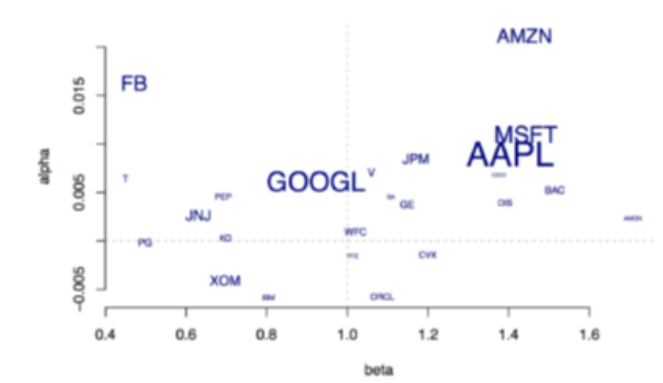
With just one x, we have simple linear regression.



$\mathbb{E}[y]$ increases by β for every unit increase in x.

Example: CAPM in finance





Fit a line between stock returns R_t and market returns $M_t(SP)$,

$$R_t \approx \alpha + \beta M_t$$

- α is money you make regardless of what the market does
- β is the asset's sensitivity to broad market movement

Important questions with using regressions for predictions

1. How well does the model fit the data?
2. What if the linear model fails?
3. Given a set of predictor values, what response value should we predict, and how accurate is our prediction?
4. Is at least one of the predictors $X_1, X_2, \ldots, X_p$ useful in predicting the response?
5. Do all the predictors help to explain Y, or is only a subset of the predictors useful? How do we select predictors?

Fitting the regression model

- **Data**: Have a lot of pairs (x_i, y_i)
- **Predictions**: $\hat{y}_i = \beta_1 + \beta_1 x_i$ is model prediction

$$\min_{(\beta_0,\beta_1)} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

- **Loss of data**: Estimate parameter β_0 and β_1 by solving least squares problem
- $RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$
- residual standard error(RSE)

$$RSE = \sqrt{\frac{1}{n-k-1} RSS}$$

R-squared Statistic

Quantity	Value
Residual standard error	3.26
R^2	0.612
F-statistic	312.1

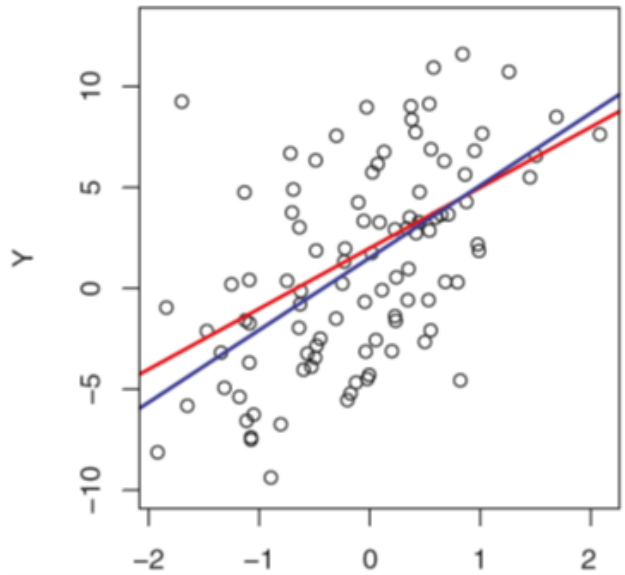
- RSE provides an absolute measure of lack of fit of a model to the data.
- Problem with RSE: measured in the units of Y and thus unclear what constitutes a good RSE. Need a measure invariant in units, i.e., in terms of ratio

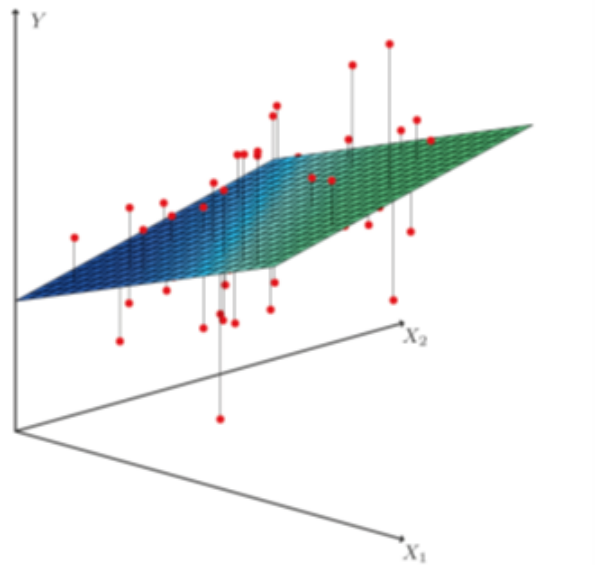
$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

- Total sum of squares: the total variance in Y
- Residual sum of squares: measuring deviation from the fitted value

What if linear models fail?

- Linear models are an extremely simplified approximation for real-world problems. There is no guarantee that a linear model will fit the data well.





- Remedies:
- 1. non-linear regression
- 2. Nonparametric estimation (if you have enough data)

Non-linear regression I

- Common practices:
- Adding quadratic terms (recall the Mincer regression)
- Adding interaction terms

An interaction term is the product of two covariates,

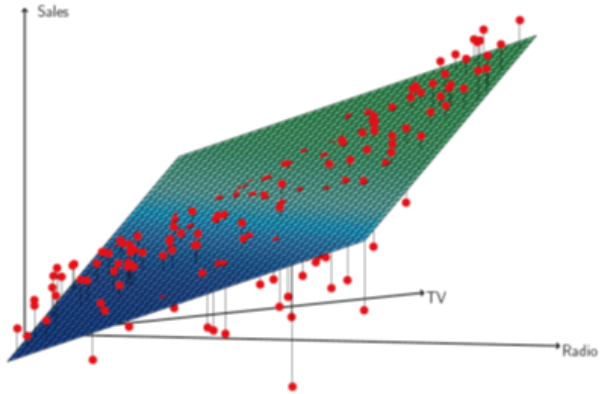
$$\mathbb{E}[\, y \mid \mathbf{x} \,] = \ldots + \beta_j x_j + x_j x_k \beta_{jk}$$

so that the effect on $\mathbb{E}[y]$ of a unit increase in x_j is $\beta_j + x_k \beta_{jk}$.
It depends upon x_k!

$$sales = \beta_0 + \beta_1 \cdot TV + \beta_2 \cdot radio + \beta_3 \cdot (radio \cdot TV) + \varepsilon$$

Non-linear regression II

- The positive residuals tend to lie along the 45-degree line (where TV and Radio budgets are split evenly), suggesting that it is difficult to decouple the effect of these two.



	Coefficient	Std. error	t-statistic	p-value
Intercept	6.7502	0.248	27.23	< 0.0001
TV	0.0191	0.002	12.70	< 0.0001
radio	0.0289	0.009	3.24	0.0014
TV×radio	0.0011	0.000	20.73	< 0.0001

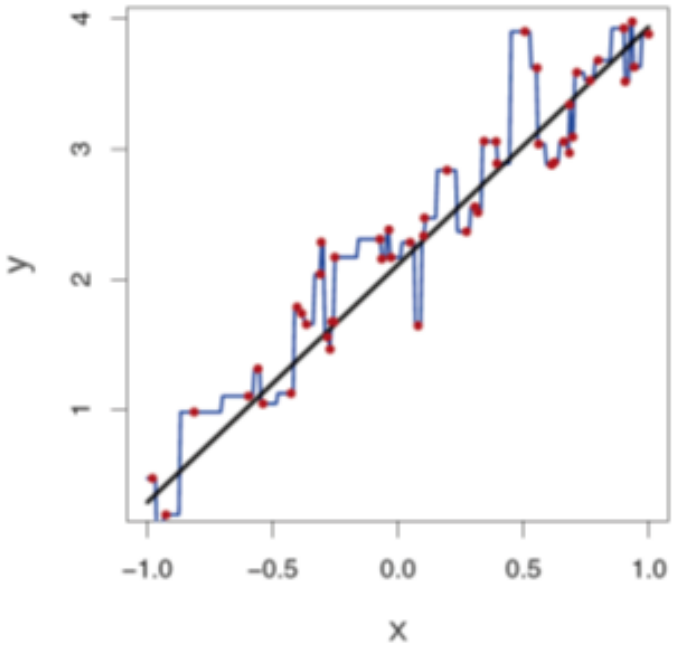
Nonparametric estimation: KNN regression I

- Parametric models rely on strong assumptions on the functional form of the model. Nonparametric methods are more flexible.
- One of the simplest and best-known method: K-nearest neighbors regression (KNN) regression
- Choose a value of K
- For a prediction point x0, identify the K training observations that are closest to x0
- Use the average of all the training responses to estimate $f(x_0)$

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i$$

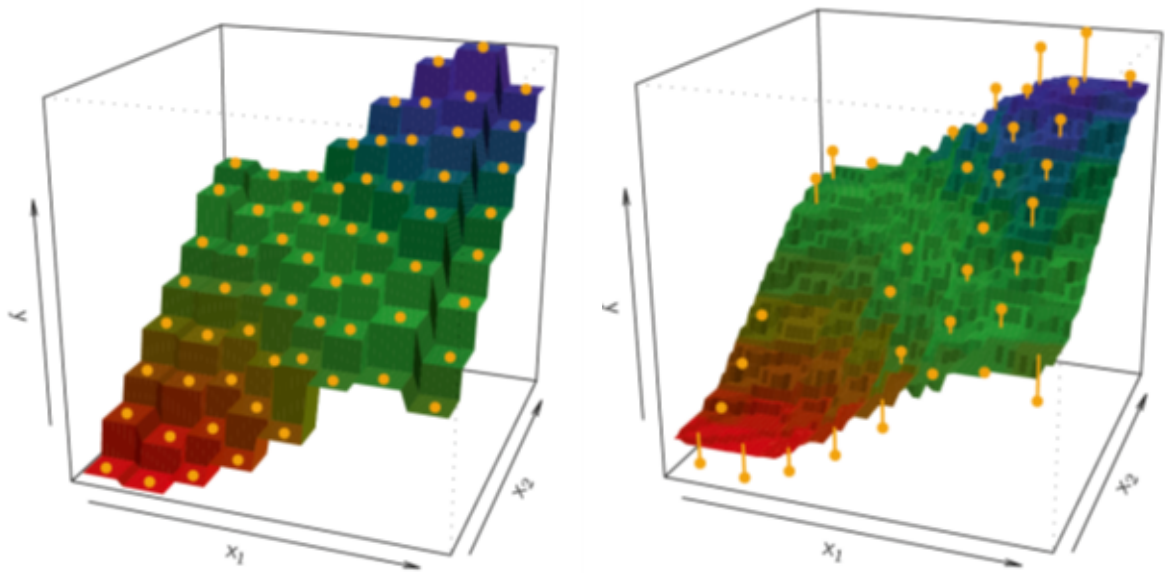
Nonparametric estimation: KNN regression II

- The non-parametric approach can cause the overfitting problem and thus a large test MSE. Thus, the parametric approach will outperform the nonparametric approach if the parametric form that has been selected is close to the true form of $f(\cdot)$.



Nonparametric estimation: KNN regression III

- How do we choose K?



- K=1
- K=9
- The optimal value for K depends on the bias-variance tradeoff

Nonparametric estimation: KNN regression IV

- A smaller K provides the more flexible fit, which will have low bias but high variance. The high variance is caused by the fact that the prediction in a given region is entirely dependent on a small number of observations.
- A larger K provides a smoother and less variable fit; the prediction in a region is an average of several observations, and so changing one observation has a smaller effect. But the smoothing fit may cause bias by missing some of the structure of $f(X)$.

Accuracy of estimates and prediction

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \ldots + \hat{\beta}_p X_p$$

is only an estimate for the **true population regression plane**

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p.$$

- The inaccuracy in the coefficient estimates is related to the reducible error. We compute a confidence interval to determine this inaccuracy.
- However, how much will Y vary with the estimate at a particular point depends also on the irreducible error. We compute a prediction intervals to account for both the errors.

```
confint(reg.both, level = 0.95) # 95% CI for the coefficients
```

```
##                 2.5 %    97.5 %
## (Intercept) -10.791000  5.668670
## rm            6.901448  8.526692
## ptratio      -1.530892 -1.003431
```

```
predict(reg.both, newdata = data.frame(rm=1, ptratio=2),
        level = 0.95, interval = "confidence")
```

```
##        fit       lwr      upr
## 1 2.618583 -4.509618 9.746783
```

Is there a relationship between the response and predictors?

- $H_0 : \beta_1 = \cdots = \beta_p = 0$
- H_a: at least one of β_j is non-zero
- Model the data as

$$y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \varepsilon$$

- This hypothesis test is performed by computing the F-statistic,

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

- The larger the F-statistic, the strong evidence against null hypotheiss

Model (feature) selection

- Recall our basic problem of using some regressors (features) to predict outcome y. In a typical high-dimensional big-data problem, the number of features is huge, making a regression impossible to interpret and difficult to run.

- $x \in \mathbb{R}^d$ for large d
- E.g.,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_d \end{bmatrix} \begin{array}{l} \text{--- living size} \\ \text{--- lot size} \\ \text{--- \# floors} \\ \text{--- condition} \\ \text{--- zip code} \\ \vdots \end{array} \longrightarrow y \text{ --- price}$$

- The number of features increases drastically when interaction terms are included.
- Can we find the best subset of x's for predicting y without including many redundant or irrelevant variables?

Model (feature) selection

- Recall our basic problem of using some regressors (features) to predict outcome y. In a typical high-dimensional big-data problem, the number of features is huge, making a regression impossible to interpret and difficult to run.

- $x \in \mathbb{R}^d$ for large d
- E.g.,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_d \end{bmatrix} \begin{array}{l} \text{--- living size} \\ \text{--- lot size} \\ \text{--- \# floors} \\ \text{--- condition} \\ \text{--- zip code} \\ \vdots \end{array} \longrightarrow y \text{ --- price}$$

- The number of features increases drastically when interaction terms are included.
- Can we find the best subset of x's for predicting y without including many redundant or irrelevant variables?

A "naïve" model selection approach

- How about using R-squared?
- R-squared is a measure of the goodness of fit and always increases with more covariates. Add one new variable to a regression and see how the R2 changes. A small increase suggests that the new variable is marginally useful.
- In-Sample(IS) R^2
  - But how well does each model perdict new(unseen) data?
  - ☹We can't use In-Sample R^2 for model comparisons.
- An Out-of-Sample(OOS) R^2
  - Split the data into 10 random subsets('folds')
  - repeat 10 times: fit $\hat{\beta}$ using only 9/10 data, and record R^2 on the left-out subset.
- ☺The OOS R^2 give us a sense of how well each model can predict data it has not seen yet.
- This process is called cross validation. But it is extremely slow when the number of features is large.

Regularization

- General idea: Impose restrictions on the estimates such that they are suitably disciplined. Therefore, variable selection is achieved with estimation rather than model comparison/testing.

  What do we mean by 'suitable restrictions'?

  - *We would like our estimates* $\widehat{\beta}$ *to be less variable,*
  - *We would like our estimates* $\widehat{\beta}$ *to yield better prediction,*
  - *We would like our estimates* $\widehat{\beta}$ *to have exact zeros.*

  We will *penalize* solutions $\widehat{\beta}$ that are not desirable. Ultimately, we are departing from optimality to stabilize the system (bias-variance tradeoff).

- We need another framework to talk about the issue of regularization.

Maximum likelihood estimation (MLE)

Likelihood is a function of the unknown parameters β of a *statistical model*, given data:

$$\mathcal{L}(\beta \,|\, \mathtt{data}) = \mathbb{P}(\mathtt{data} \,|\, \beta).$$

Maximum Likelihood (ML) Estimation:

$$\widehat{\beta} = \max_{\beta} \mathcal{L}(\beta \,|\, \mathtt{data})$$

ML estimates $\widehat{\beta}$ are those parameter values β that are most likely to have generated our data.

$$\mathbb{P}(data \,|\, \beta) = \mathbb{P}(y_1|\boldsymbol{x}_1, \beta) \times \mathbb{P}(y_2|\boldsymbol{x}_2, \beta) \cdots \times \mathbb{P}(y_n|\boldsymbol{x}_n, \beta).$$

Deviance in MLE

Deviance refers to the distance between our fit and "data" (saturated model). You want to make it as small as possible.

Minimize deviance $\Leftrightarrow$ *maximize likelihood.*

$$Dev(\boldsymbol{\beta}) = -2\log \mathcal{L}(\boldsymbol{\beta} \,|\, \texttt{data}) + C$$

C is a constant you can mostly ignore.

Deviance is a measure of GOF that plays the role of residual sums of squares for a broader class of models (logistic regression etc.)

$$Dev(\boldsymbol{\beta}) \propto \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 \qquad R^2 = 1 - \frac{dev(\widehat{\boldsymbol{\beta}})}{dev(\boldsymbol{\beta} = \mathbf{0})}.$$

Penalized MLE for regularization

Using all predictors $x_1, \ldots, x_p$, the *maximum likelihood estimator* $\widehat{\beta}_{MLE}$ minimizes deviance:

$$\widehat{\beta}_{MLE} = \arg\min_{\beta} \left\{ -\frac{2}{n} \log LHD(\beta) \right\}.$$

The *penalized maximum likelihood estimator* $\widehat{\beta}_{PMLE}$ minimizes deviance **plus a penalty**:

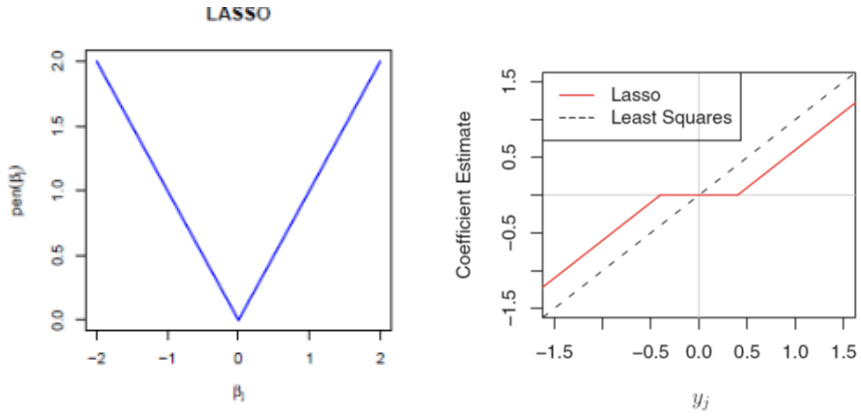
$$\widehat{\beta}_{PMLE} = \arg\min_{\beta} \left\{ -\frac{2}{n} \log LHD(\beta) + pen(\beta) \right\}.$$

- The penalty term is created such that it is small when beta is close to zero. Thus, it has the effect of shrinking the estimates towards zero if they are not important to fit the model.

The PMLE estimator is **shrunk to zero** $\rightsquigarrow$ *automatic variable selection* ☺.

LASSO

- The most famous method for regularization is LASSO (least absolute shrinkage and selection operator).
- The LASSO fits $\hat{\beta}_{PMLE}$ to minimize $-\frac{2}{n}\log LHD(\beta) + \lambda \sum_j |\beta_j|$.



- With appropriate choice of lamda, LASSO performs variable selection by forcing some estimated coefficients to zero. Or we say, LASSO yields sparse models by reducing dimensions of a model.s
- LASSO is the modern version of OLS.

LASSO in practice

The LASSO fits $\hat{\beta}_{PMLE}$ to minimize $-\frac{2}{n}\log LHD(\beta) + \lambda \sum_j |\beta_j|$.

LASSO path estimation:

Start with big λ_1 so big that $\widehat{\beta}_{\lambda_1} = \mathbf{0}$ (null model).

For $t = 2 \ldots T$: update $\widehat{\beta}_{\lambda_{t-1}}$ to be optimal under $\lambda_t < \lambda_{t-1}$.

Since estimated $\widehat{\beta}_{\lambda_t}$ changes smoothly along this path:

- ☺ It's fast! Each update is easy.
- ☺ It's stable: optimal λ_t may change a bit from sample to sample, but that won't affect the model much.

There are many packages for fitting lasso regressions in R.

`glmnet` is most common. `gamlr` is a useful alternative.

These two are very similar, and they share syntax.

Both use the `Matrix` library representation for sparse matrices.

References I

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Stock, J. H. and Watson, M. W. (2020). *Introduction to econometrics*, volume 4. Pearson New York.