MACHINE LEARNING

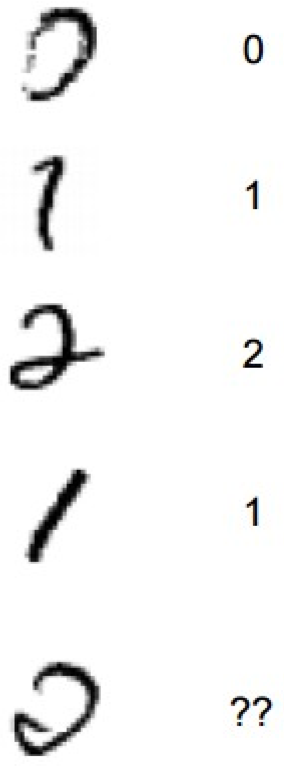
Slide adapted from *learning from data* book and course, and Berkeley cs188 by Dan Klein, and Pieter Abbeel

Machine Learning ??

- Learning from data
- Tasks:
  - Prediction
  - Classification
  - Recognition
- Focus on Supervised Learning only
- Classification: Naïve Bayes
- Regression: Linear Regression

Example: Digit Recognition

- Input: images/ pixel grids
- Output: a digit 0-9
- Setup:
  - Get a large collection of example images, each label with a digit
  - Note: someone has to hand label all this data
  - Want to learn to predict labels of new, future digit images



Other classification Tasks

- Classification: given inputs x, predict labels (classes) y
- Examples:
  - Spam detection (input: document/email, classes: spam or not)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grading (input: document, classes: grades)
  - Movie rating (input: a movie, classes: rating)
  - Credit Approval (input: user profile, classes: accept/reject)
  - … many more

The essence of machine learning

- The essence of machine learning:
  - A pattern exists
  - We cannot pin it down mathematically
  - We have data on it
- A pattern exists. We don’t know it. We have data to learn it.
- Learning from data to get an information that can make prediction

Credit Approval Classification

- Applicant information:

Age	23 years
Gender	male
Annual salary	$30,000
Years in residence	1 year
Years in job	1 year
Current debt	$15,000
…	…

- Approve credit?

Credit Approval Classification

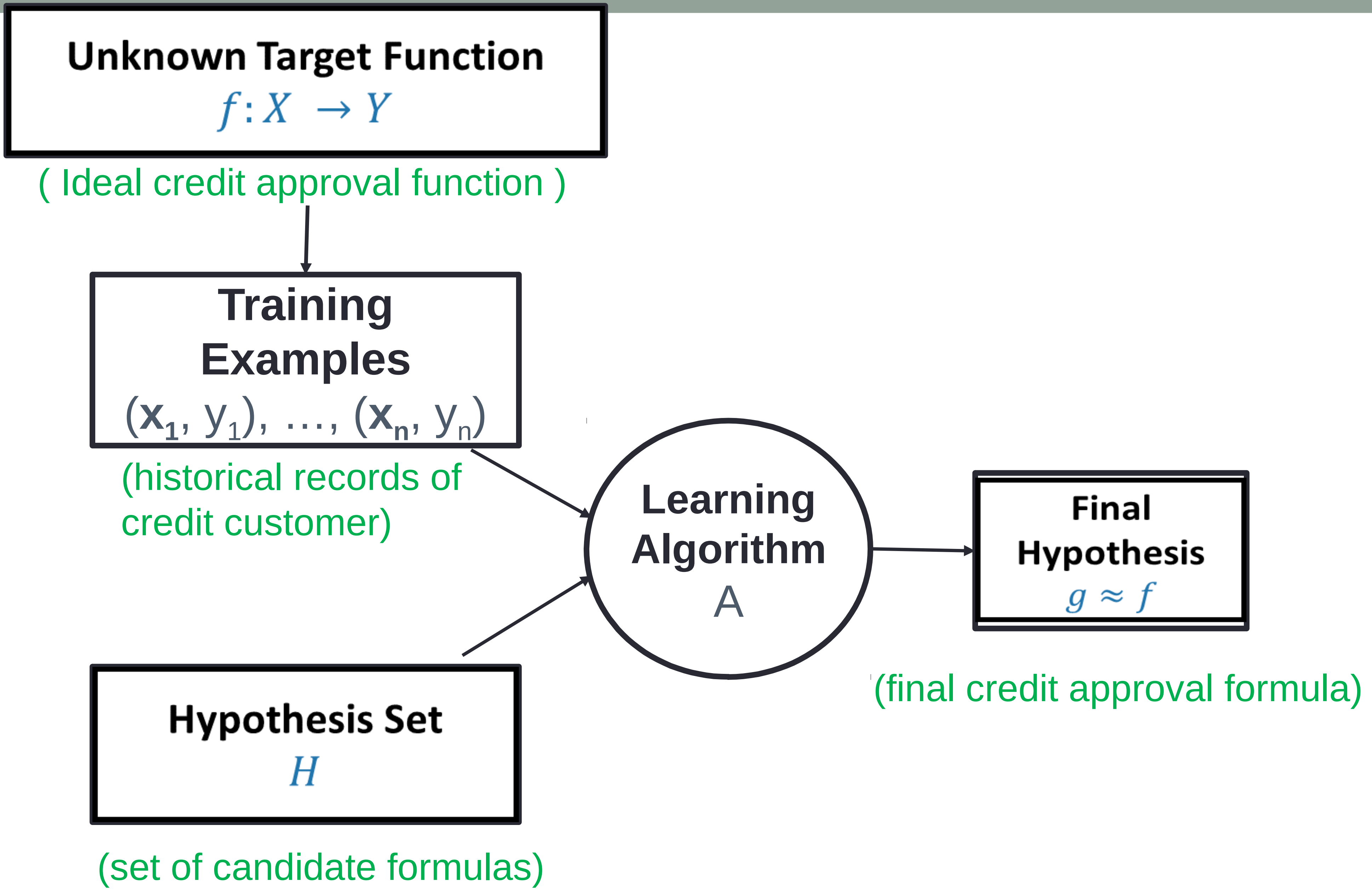
- There is no credit approval formula
- Banks have a lots of data
  - Customer information: checking status, employment, etc.
  - Whether or not they defaulted on their credit (good or bad).

Relation: german_credit

No.	checking_status Nominal	duration Numeric	credit_history Nominal	purpose Nominal	credit_amount Numeric	savings_status Nominal	employment Nominal	installment_commitment Numeric	personal_status Nominal
1	(0	6.0	critical/other exi...	radio/tv	1169.0	no known savi...	)=7	4.0	male single
2	0(=X(200	48.0	existing paid	radio/tv	5951.0	(100	1(=X(4	2.0	female div/dep...
3	no checking	12.0	critical/other exi...	education	2096.0	(100	4(=X(7	2.0	male single
4	(0	42.0	existing paid	furnitu...	7882.0	(100	4(=X(7	2.0	male single
5	(0	24.0	delayed previously	new car	4870.0	(100	1(=X(4	3.0	male single
6	no checking	36.0	existing paid	education	9055.0	no known savi...	1(=X(4	2.0	male single

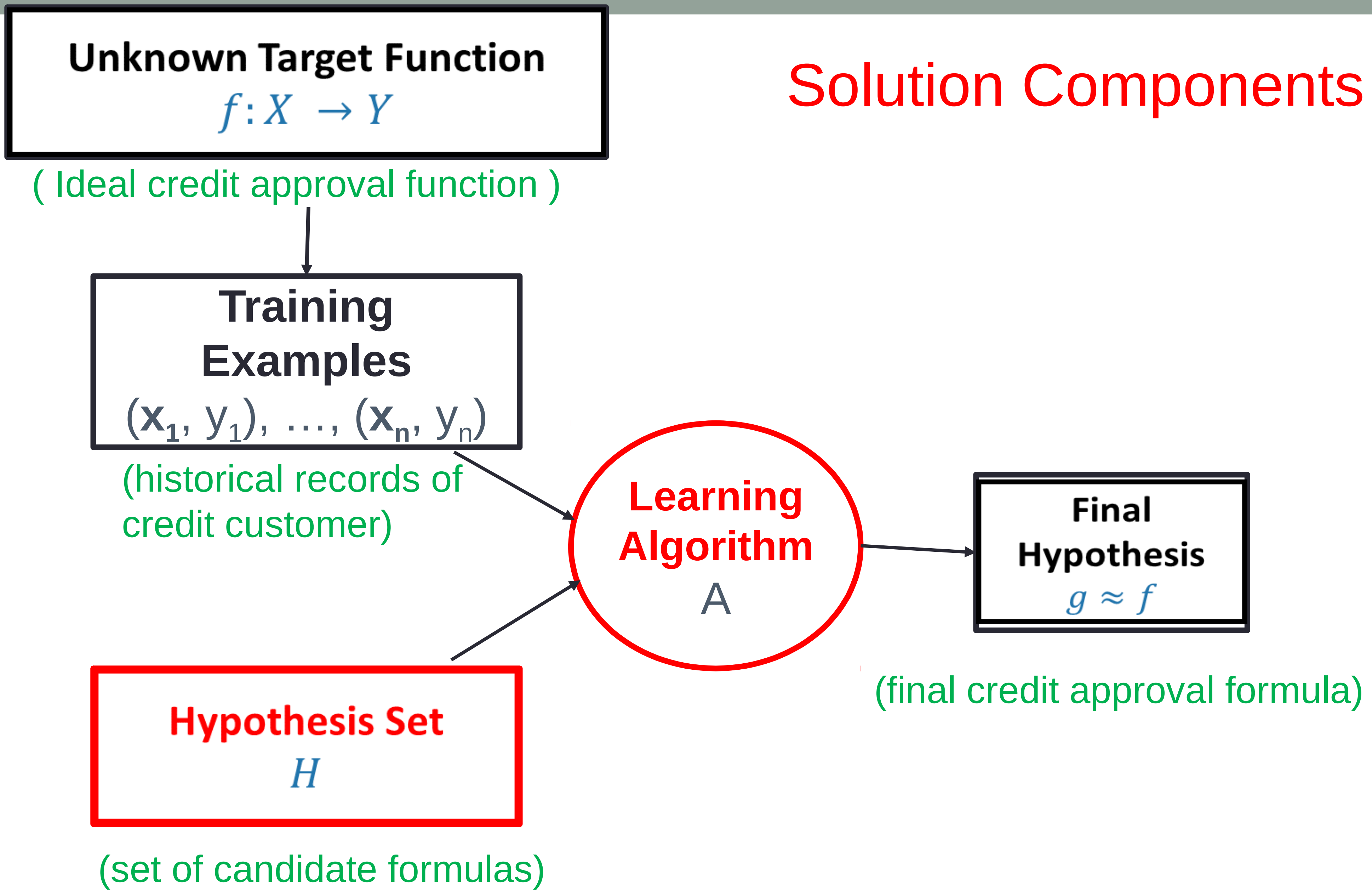
Components of learning

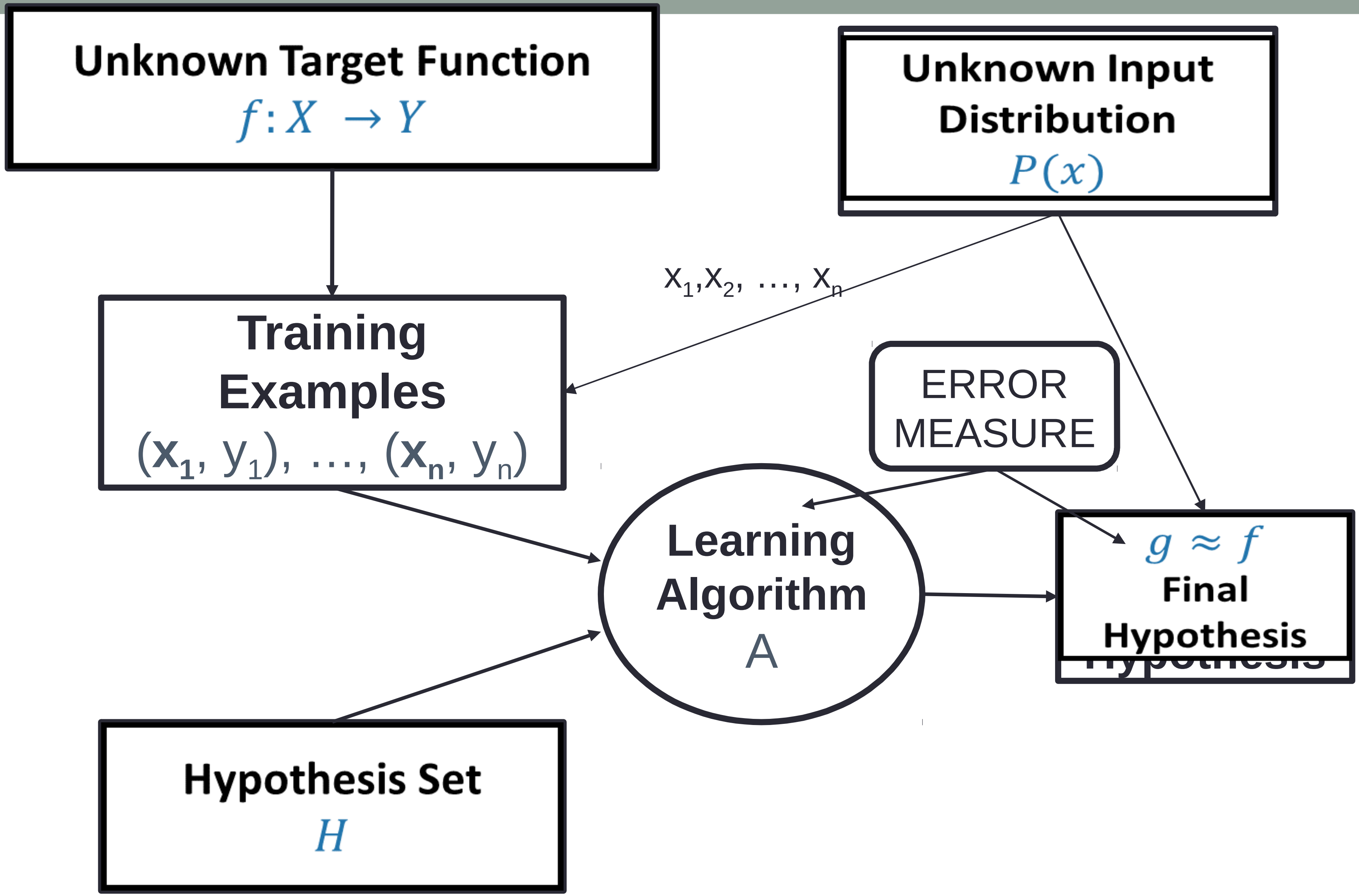
- Formalization:
  - Input: **x** (customer application)
  - Output: y (good/bad customer?)
  - Target function: (ideal credit approval formula)
  - Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ (historical records)
  - Hypothesis: $g : X \to Y$ (formula/classifier to be used)



Unknown Target Function
$f: X \to Y$
(Ideal credit approval function)
Training Examples
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
(historical records of credit customer)
Learning Algorithm
A
Final Hypothesis
$g \approx f$
(final credit approval formula)
Hypothesis Set
H
(set of candidate formulas)

Solution Components





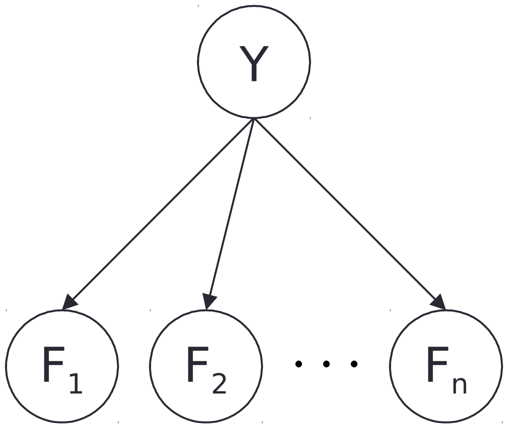
The general supervised learning problem

Model-Based Classification

- Model-Based approach
  - Build a model (e.g. Bayes' net) where both the label and features are random variables
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features
- Challenges (solution components)
  - How to answer the query
  - How should we learn its parameters?
  - What structure should the BN have?

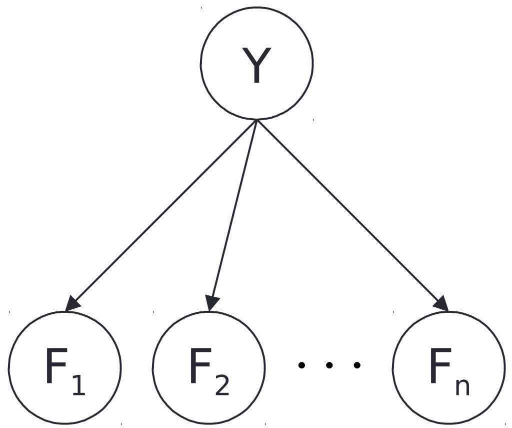
Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independent effects of the label
- In other word: features are conditional independent given the class/label
- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature vales are on/off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to feature vector, e.g.
  -  $\to < F_{0,0} = 0,\ F_{0,1} = 0,\ \ldots,\ F_{15,15} = 0 >$
- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$



General Naïve Bayes

- A general Naïve Bayes Model:



- |Y| parameters

$$P(\mathsf{Y}, \mathsf{F}_1 \ldots \mathsf{F}_n) = \quad P(\mathsf{Y}) \prod_i P(\mathsf{F}_i | \mathsf{Y})$$

$|Y| \times |F|^n$ values

$|Y| \times |F|^n$ values

- We only have to specify how each feature depends on the class
- Total number of parameters is linear in n
- Model is very simplistic, but often work anyway.

Inference for Naïve Bayes

- Goal: compute posterior distribution over label variable Y
  - Step 1: get joint probability of label and evidence for each label

$$P(Y, f_1 \ldots f_n) = \begin{bmatrix} P(y_1, f_1 \ldots f_n) \\ P(y_2, f_1 \ldots f_n) \\ \vdots \\ P(y_k, f_1 \ldots f_n) \end{bmatrix} \Rightarrow \frac{\begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}}{P(f_1 \ldots f_n)}$$

$+$

$$\Downarrow$$

$$P(Y|f_1 \ldots f_n)$$

  - Step 2: sum to get probability of evidence
  - Step 3: normalize by dividing Step 1 by Step 2

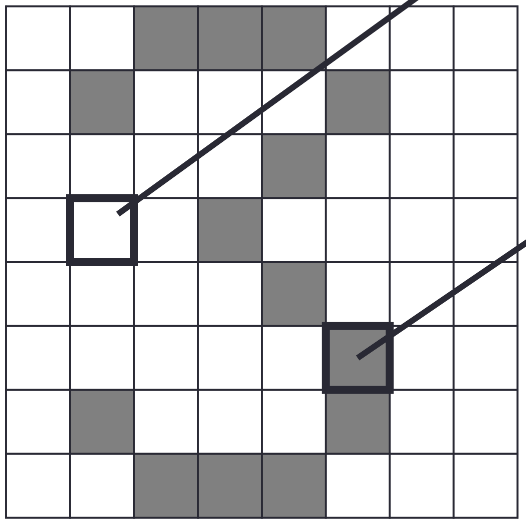
General Naïve Bayes

- What do we need in order to use Naïve Bayes?
  - Inference method (we just saw this part)
    - Start with a bunch of probabilities: P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1 \ldots F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the *parameters* of the model and denoted by $\boldsymbol{\theta}$
    - Up until now, we assumed these appeared by magic, but…
    - …they typically come from training data counts

Example: Conditional Probabilities

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Parameter Estimation

- Estimating the distribution of a random variable (CPTs)
- Elicitation: ask a human (why is this hard?)
- Empirically: use training data (learning!)
  - E.g.: for each outcome x, look at the empirical rate of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

r r b

$$P_{\text{ML}}(\text{r}) = 2/3$$

  - This is the estimate that maximizes the likelihood of the data

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

  - Relative frequencies are the maximum likelihood estimate

Unseen Events and Laplace Smoothing

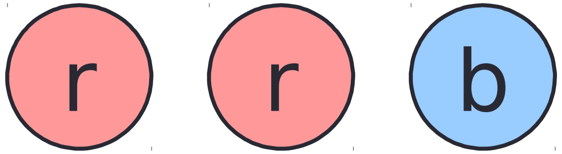
- What happen if you've never seen an event or feature for a given class?
- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

|X| = #class

$$P_{LAP,k}(x) = \frac{c(x)+k}{N+k|X|}$$



$$P_{ML}(X) =$$

$$P_{LAP}(X) =$$

Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems

Input representation and features

- ‘raw’ input x = < $F_{0,0}$ = 0, $F_{0,1}$ =0 , …, $F_{15,15}$ =0>
- ‘raw’ input x = (x_0, x_1, x_2, …, x_{256})
- Features: Extract useful information, e.g.,
  - Before: Feature vales are on/off, based on whether intensity is more or less than 0.5 in underlying image
  - Intensity and symmetry x = (x_0, x_1, x_2)

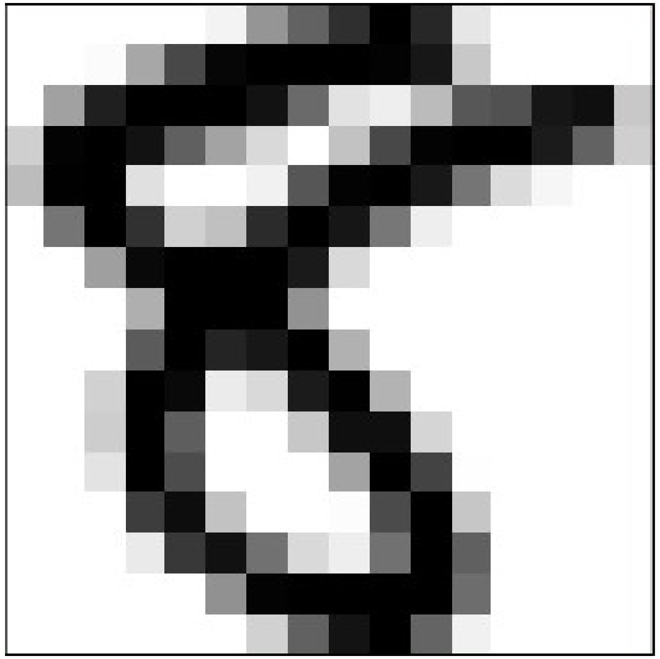
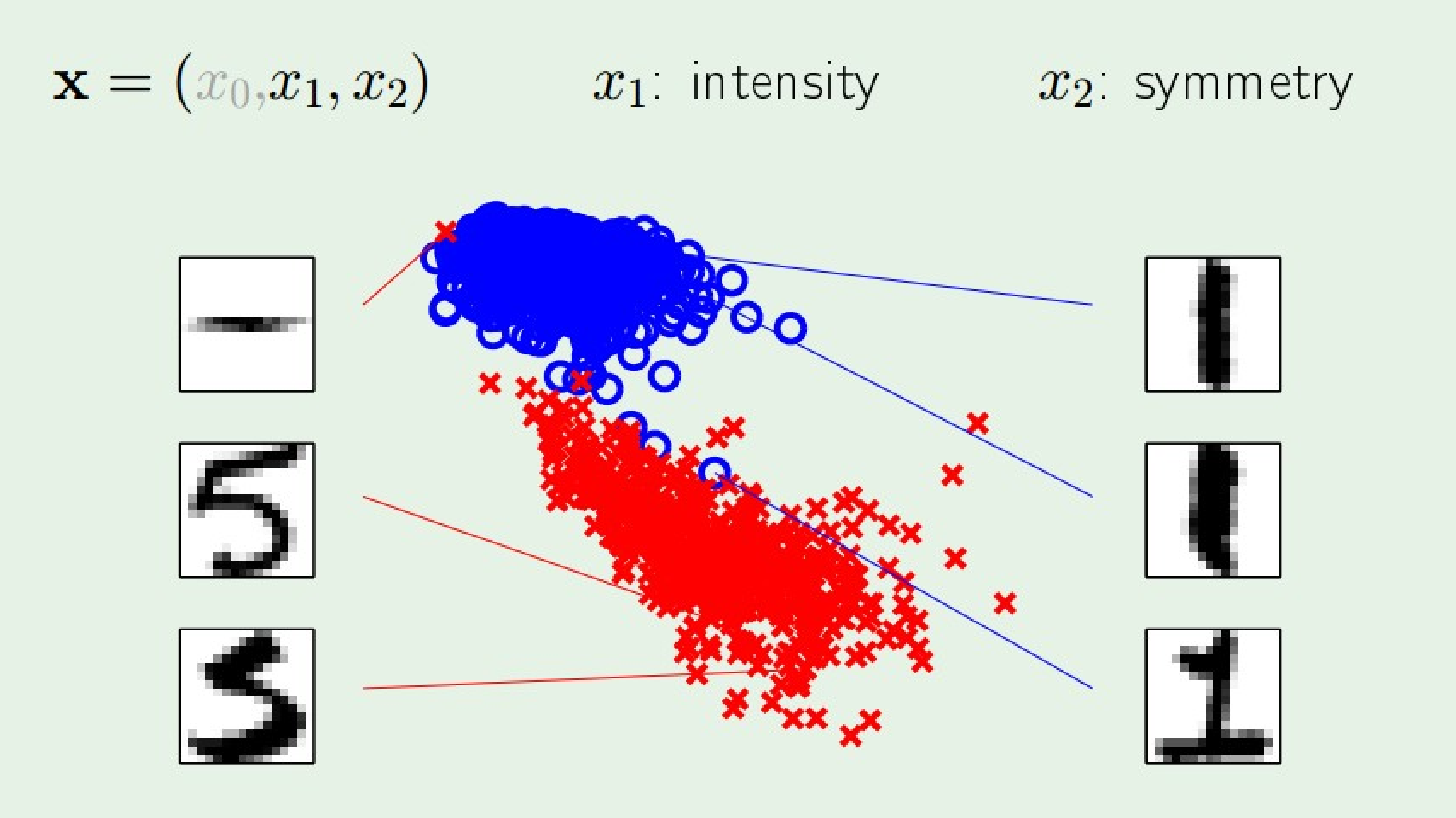


Illustration of features



Linear Regression

Credit Approval Again

- Classification: Credit Approval (yes/no)
- Regression: Credit line (dollar amount)
- Input x =

Age	23 years
Annual salary	$30,000
Years in job	1 year
Current depth	$15,000
…	…

- Idea: Assign weight to each attribute/feature based on how important it is.
- Linear regression output:

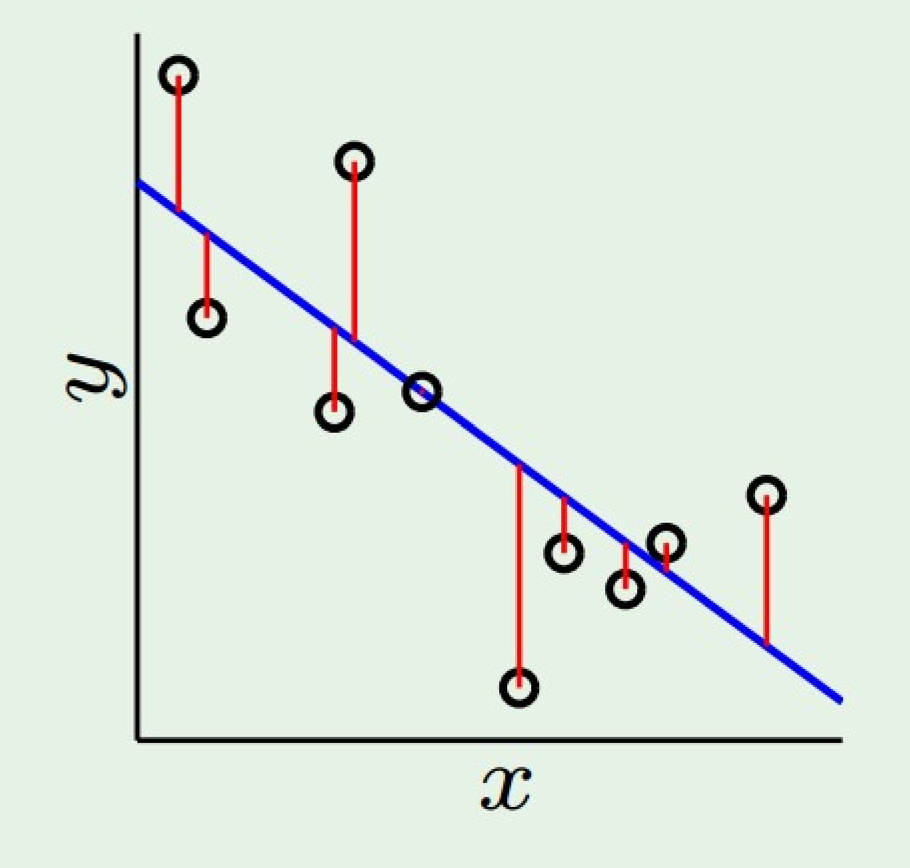
How to measure the error

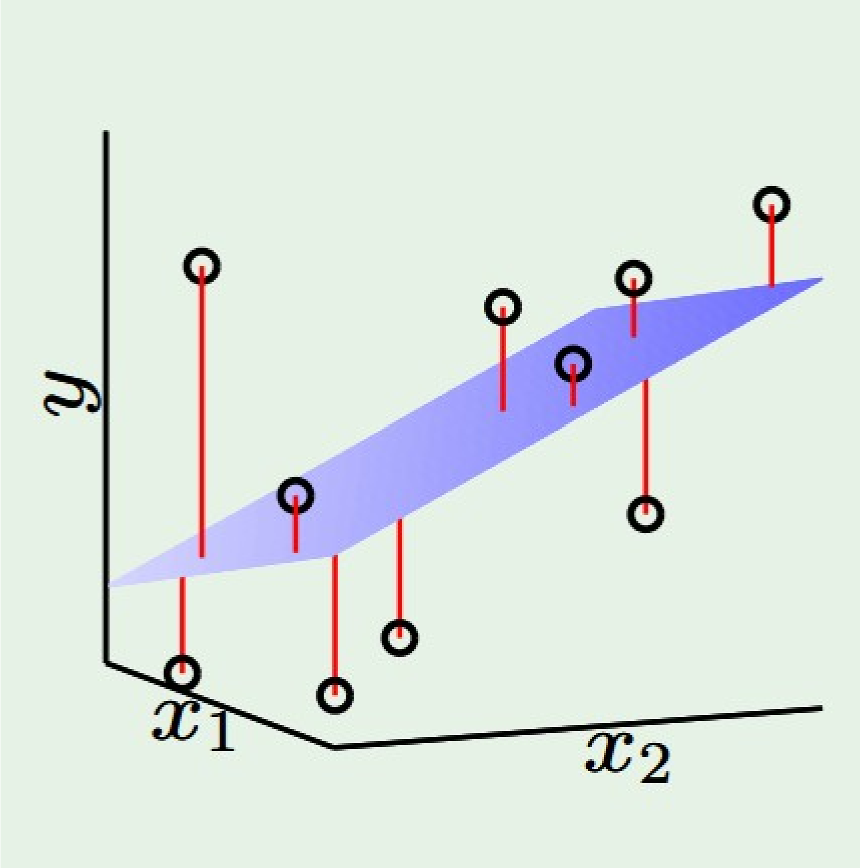
- How well does approximate ?
- In classification, count the number of misclassified.
- In linear regression, we use squared error 2

- In-sample error:

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$$

Illustration of linear regression





The expression for E_{in}

$$
\begin{aligned}
E_{\text{in}}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^{\intercal}\mathbf{x}_n - y_n)^2 \\
&= \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2
\end{aligned}
$$

where $\quad \mathrm{X} = \begin{bmatrix} —\mathbf{x}_1{}^{\intercal}— \\ —\mathbf{x}_2{}^{\intercal}— \\ \vdots \\ —\mathbf{x}_N{}^{\intercal}— \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Minimizing E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\mathrm{X}^{\intercal}(\mathrm{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$$

$$\mathrm{X}^{\intercal}\mathrm{X}\mathbf{w} = \mathrm{X}^{\intercal}\mathbf{y}$$

$\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y}$ where $\mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$

$\mathrm{X}^{\dagger}$ is the 'pseudo-inverse' of X

The linear regression algorithm

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

$$\underbrace{\mathrm{X} = \begin{bmatrix} —\mathbf{x}_1^{\intercal}— \\ —\mathbf{x}_2^{\intercal}— \\ \vdots \\ —\mathbf{x}_N^{\intercal}— \end{bmatrix}}_{\text{input data matrix}}, \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

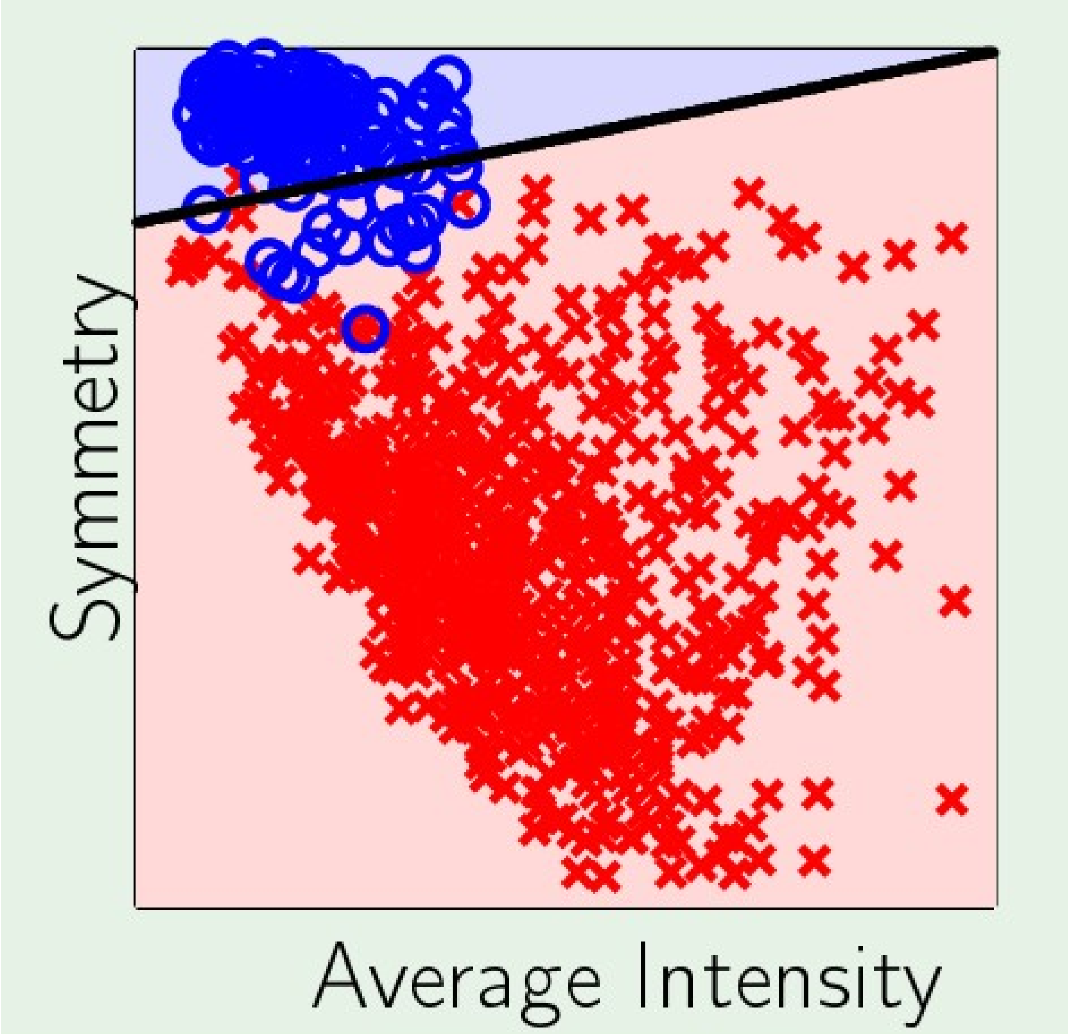
2: Compute the pseudo-inverse $\mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$.

3: Return $\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y}$.

Linear regression for classification

- Linear regression learns a real-valued function $y = f(x) \epsilon R$
- Binary-valued functions are also real-valued! $\pm 1 \epsilon R$
- Use linear regression to get **w** where $\boldsymbol{w}^T \boldsymbol{x}_n \approx y_n = \pm 1$
- In this case, $\text{sign}(\boldsymbol{w}^T \boldsymbol{x}_n)$ is likely to agree with $y_n = \pm 1$

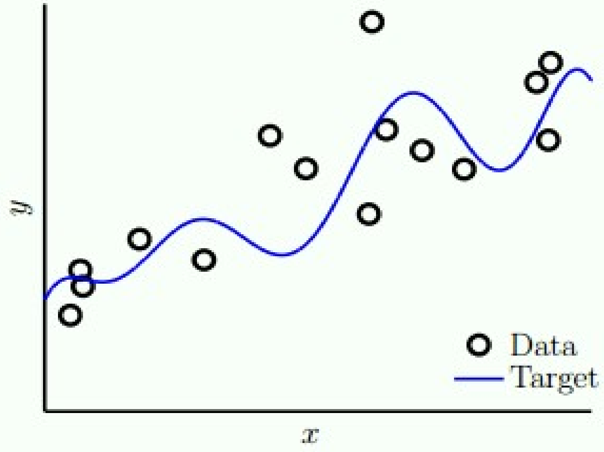
Linear regression boundary



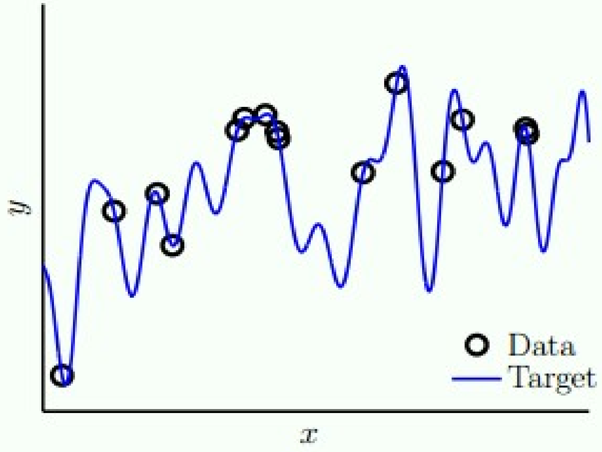
Overfitting

- Happen when a classifier fits the training data too tightly and results in a lot of error when try to predict outside data.
- In other word, fitting the data more than is warranted.
- Overfitting is a general problem because
  - There are noises in data. Try to fit noises is not a good idea
  - The true model (f) is very complex and our training data cannot really represent it well.

Case Study: 2nd vs 10th Order Polynomial Fit



10th order f with noise.



50th order f with no noise.

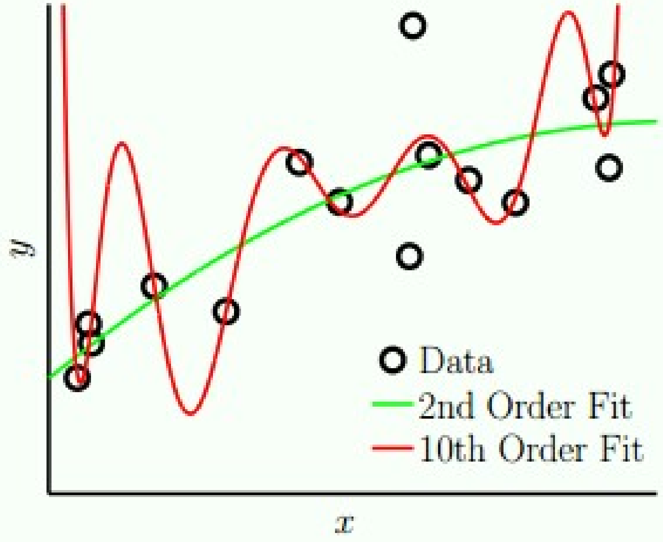
$\mathcal{H}_2$: 2nd order polynomial fit

$\mathcal{H}_{10}$: 10th order polynomial fit

$\longleftarrow$ special case of linear models with feature transform $x \mapsto (1, x, x^2, \cdots)$.

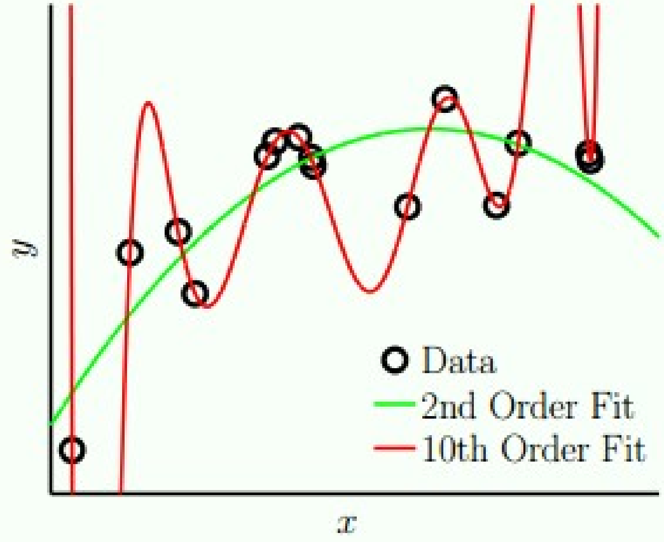
Which model do you pick for which problem and why?

Case Study: 2nd vs 10th Order Polynomial Fit



simple noisy target

	2nd Order	10th Order
E_{in}	0.050	0.034
E_{out}	0.127	**9.00**



complex noiseless target

	2nd Order	10th Order
E_{in}	0.029	10^{-5}
E_{out}	0.120	**7680**

Go figure:

Simpler $\mathcal{H}$ is better even for the more complex target with no noise.

Training and Testing

- Divided data set into two sets:
  - Training set
  - Test set
  - (sometime there will be one more set called Held out set for tuning parameters
- Experimentation cycle
  - Learning parameters (e.g. model probabilities or weights) on training set
  - Compute accuracy of test set
  - Very important: never “peek” at the test set and never let test set influence your learning.
- Evaluation
  - Accuracy or Error from the training set (out-of-sample error)

Resource:

- Learning from data
  - http://work.caltech.edu/telecourse.html
- Andrew Ng Machine Learning
  - https://www.coursera.org/learn/machine-learning
  - https://www.youtube.com/watch?v=UzxYlbK2c7E&list=PLA89DCFA6ADACE599
- In-depth introduction to machine learning in 15 hours of expert videos
  - https://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/
- Python ML library: http://scikit-learn.org/stable/
- WekaMOOC : https://weka.waikato.ac.nz/explorer