Applied Machine Learning

Spring 2019, CS 519

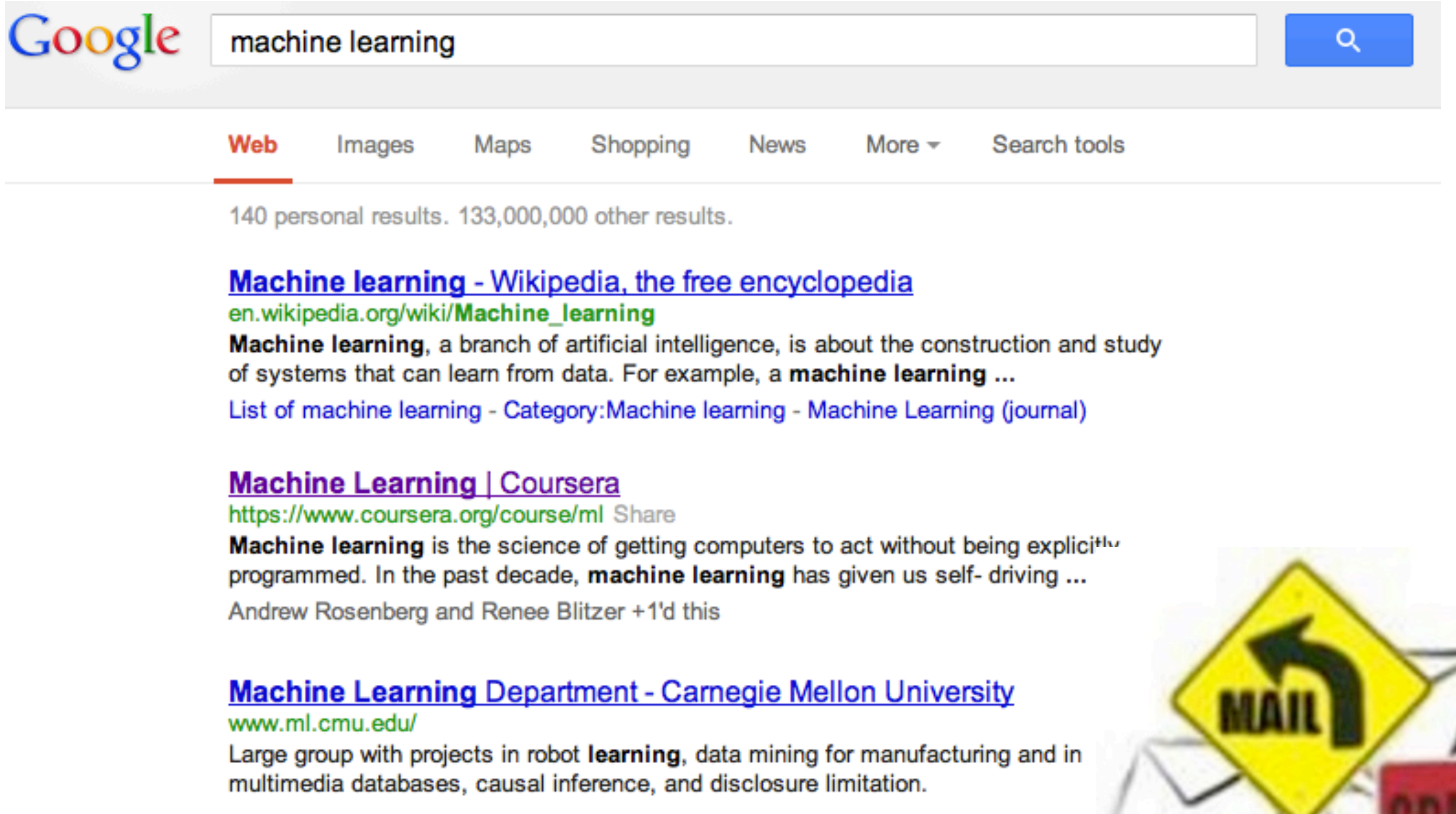
Prof. Liang Huang

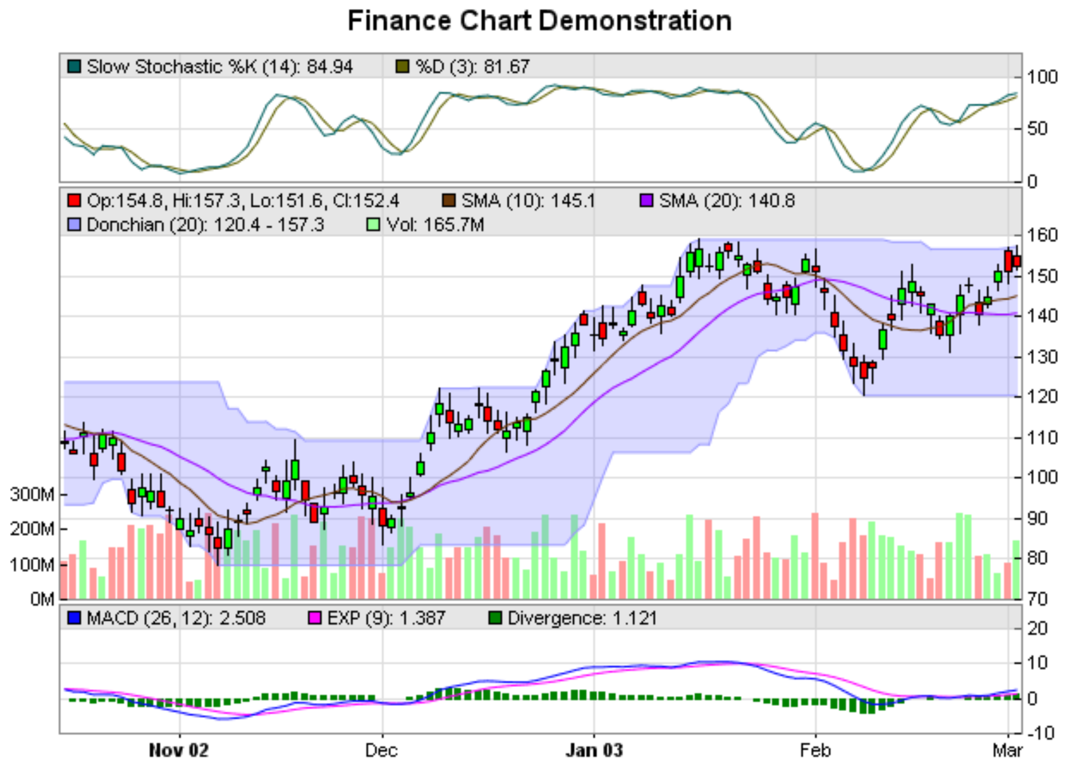
School of EECS
Oregon State University

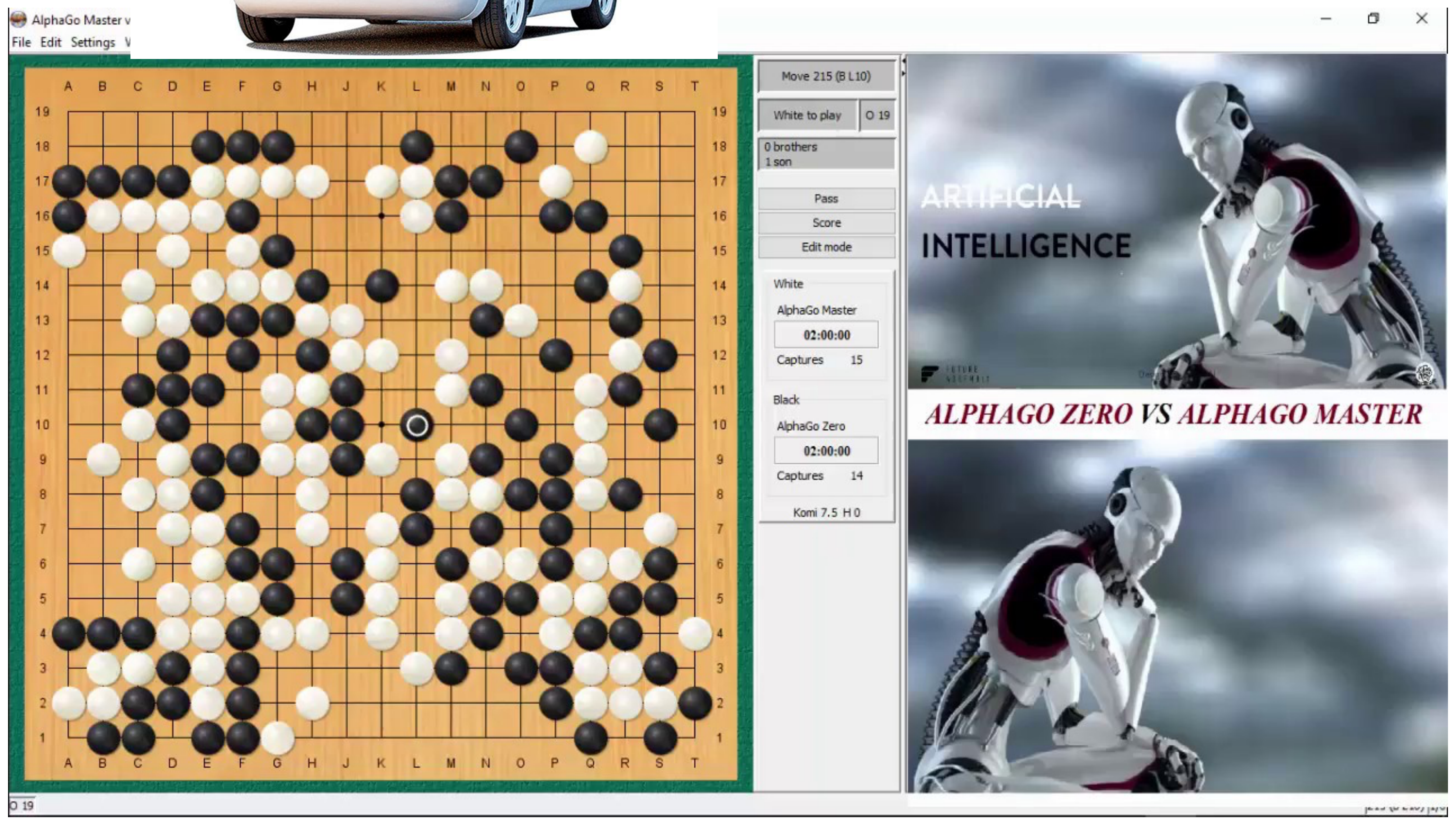
liang.huang@oregonstate.edu

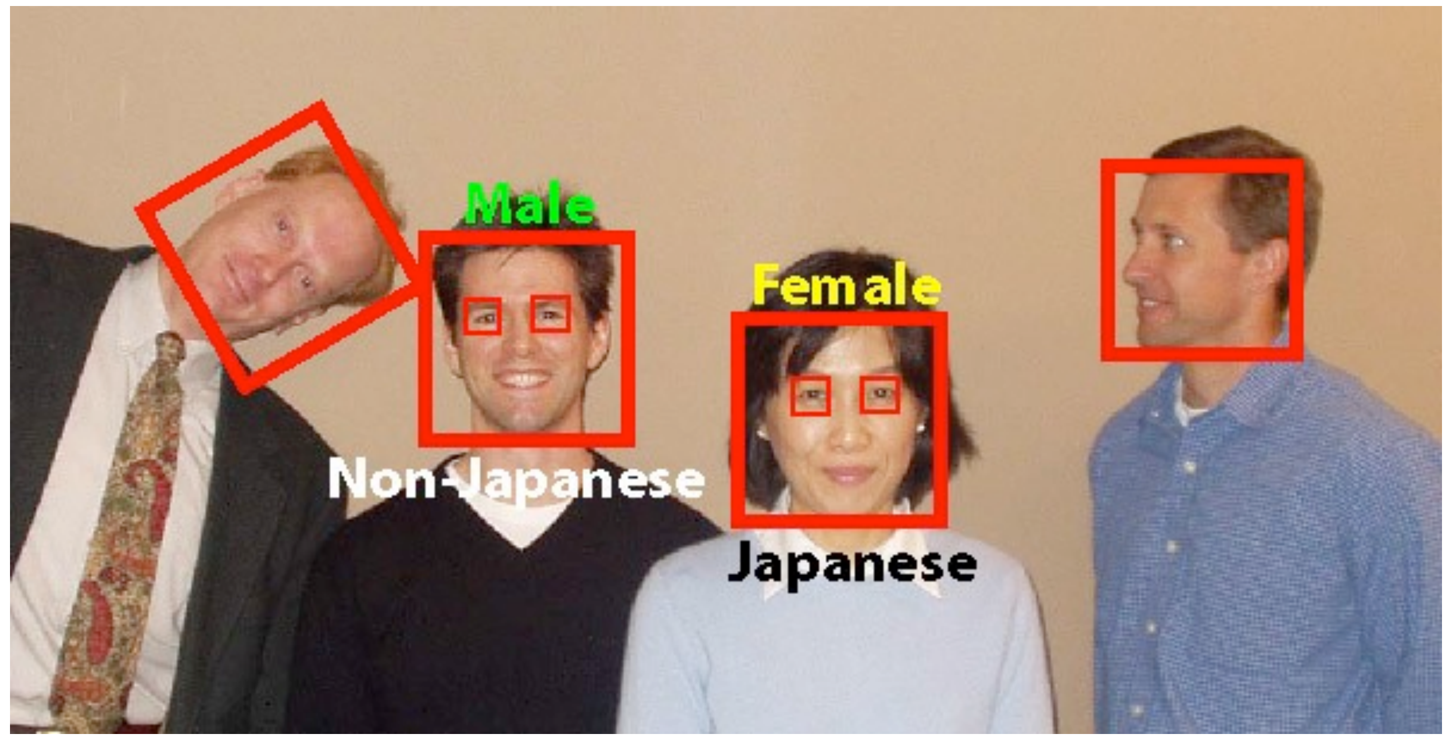
Machine Learning is Everywhere

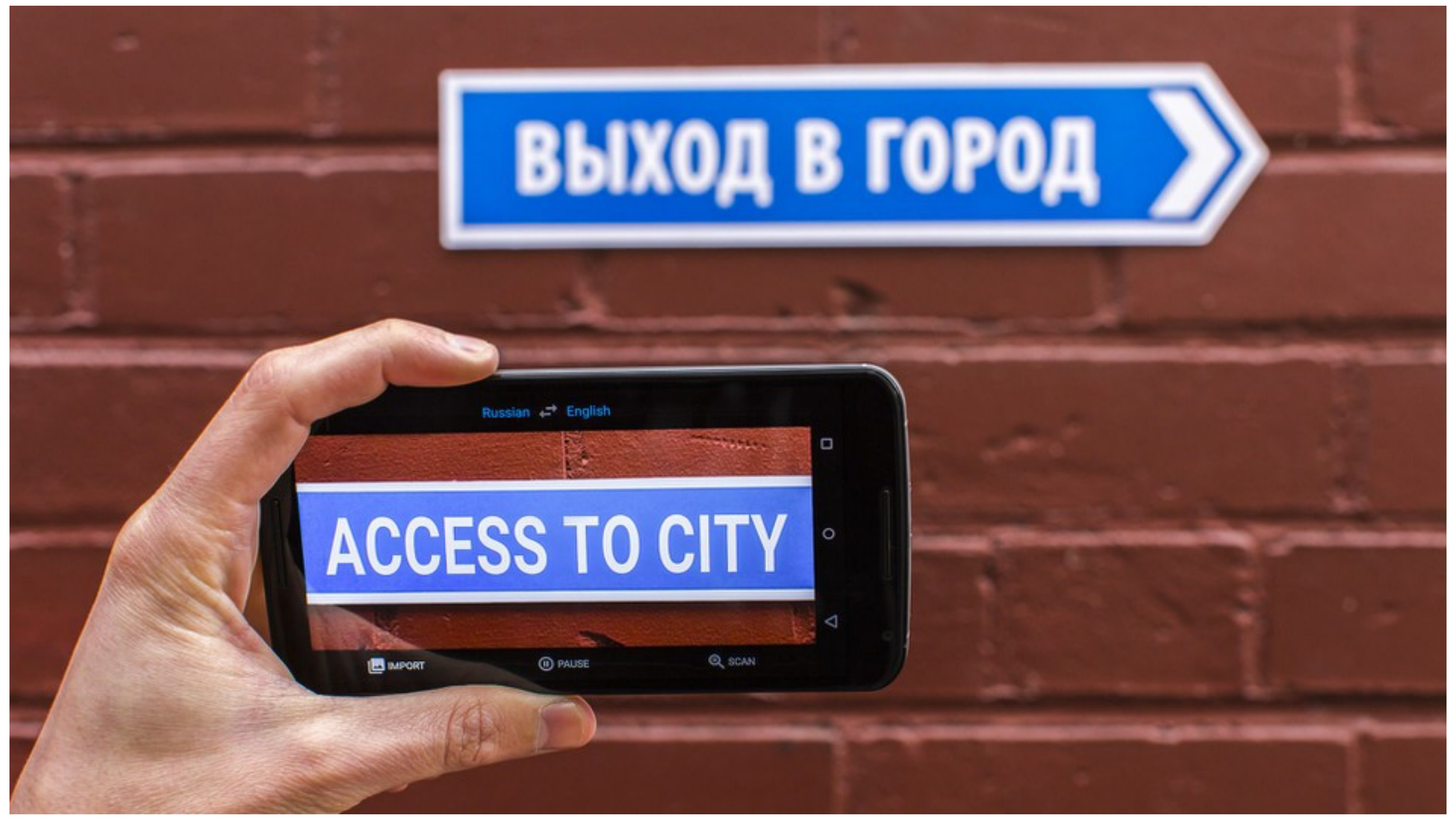
- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates)



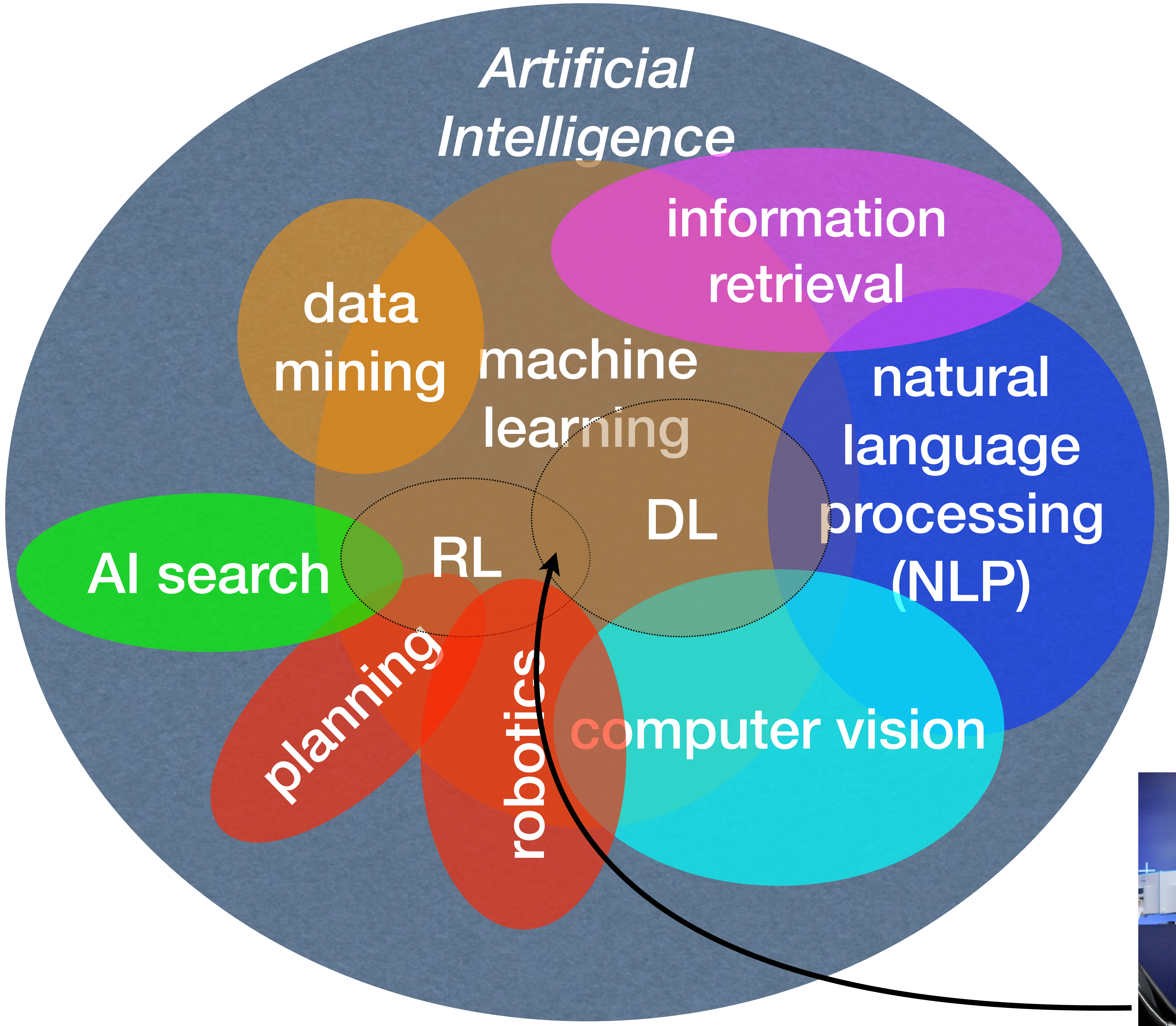








AI Subfields and Breakthroughs



IBM Deep Blue, 1997
AI search (no ML)

IBM Watson, 2011
NLP + very little ML

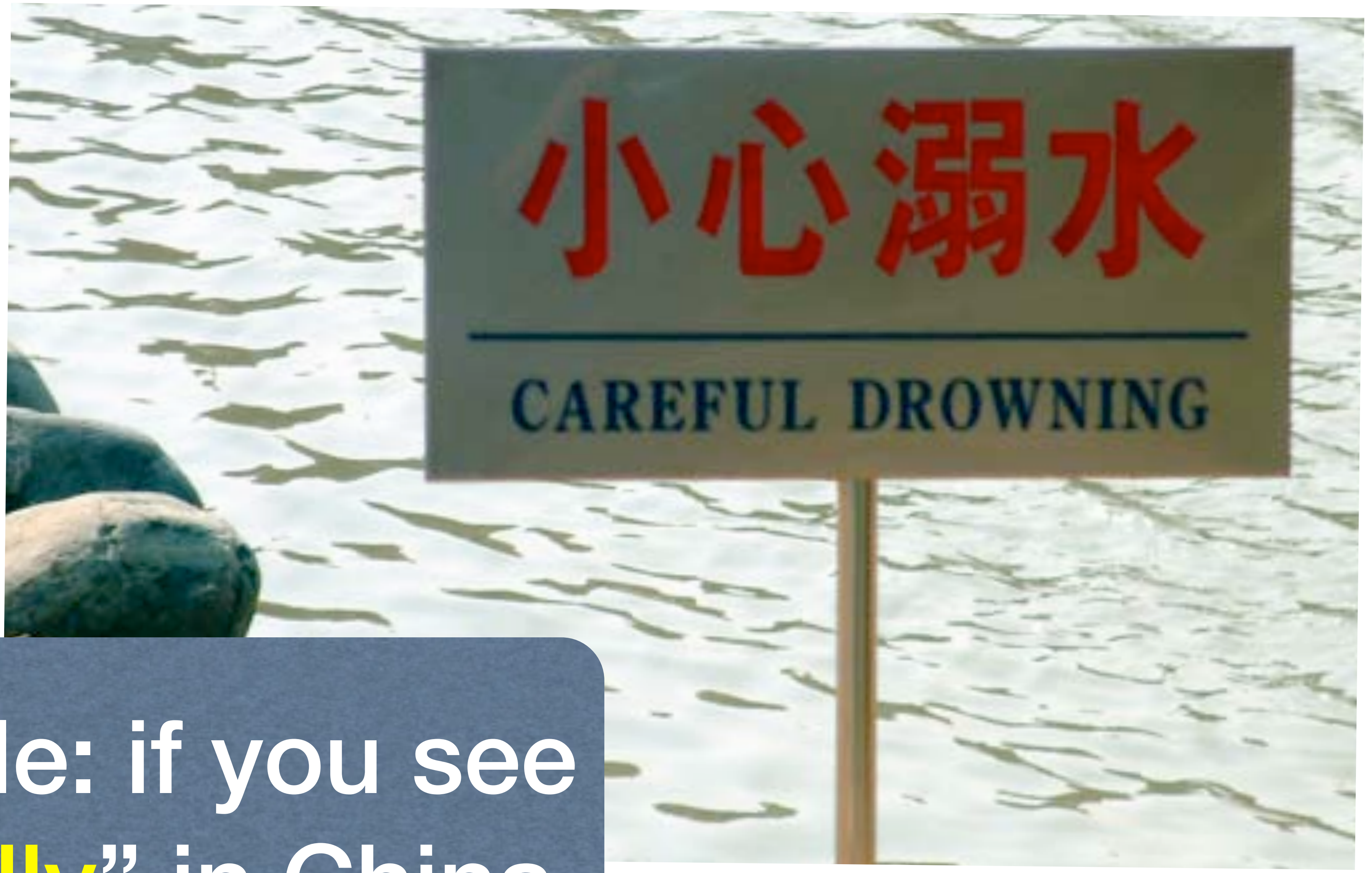
Google DeepMind AlphaGo, 2017
deep reinforcement learning + AI search

The Future of Software Engineering

- "See, when AI comes, I'll be long gone (being replaced by autonomous cars) but the programmers in those companies will be too, by automatic program generators." --- an Uber driver to an ML prof

Uber uses tons of AI/ML:
route planning, speech/dialog,
recommendation, etc.

Machine Learning Failures



liang's rule: if you see "X carefully" in China, just don't do it.

Machine Learning Failures

Machine Learning Failures

clear evidence that AI/ML is used in real life.

- Part II: Basic Components of Machine Learning Algorithms;
  Different Types of Learning

What is Machine Learning

- Machine Learning = Automating Automation
  - Getting computers to program themselves
  - Let the data do the work instead!

Traditional Programming

rule-based translation
(1950-2000)

I love Oregon **Input** → **Computer** → **Output** 私はオレゴンが大好き
Program → **Computer**

Machine Learning

learning-based translation
(1990-now)

I love Oregon **Input** → **Computer** → **Program**
私はオレゴンが大好き **Output** → **Computer**

Google Translate
(2003-now)

Magic?

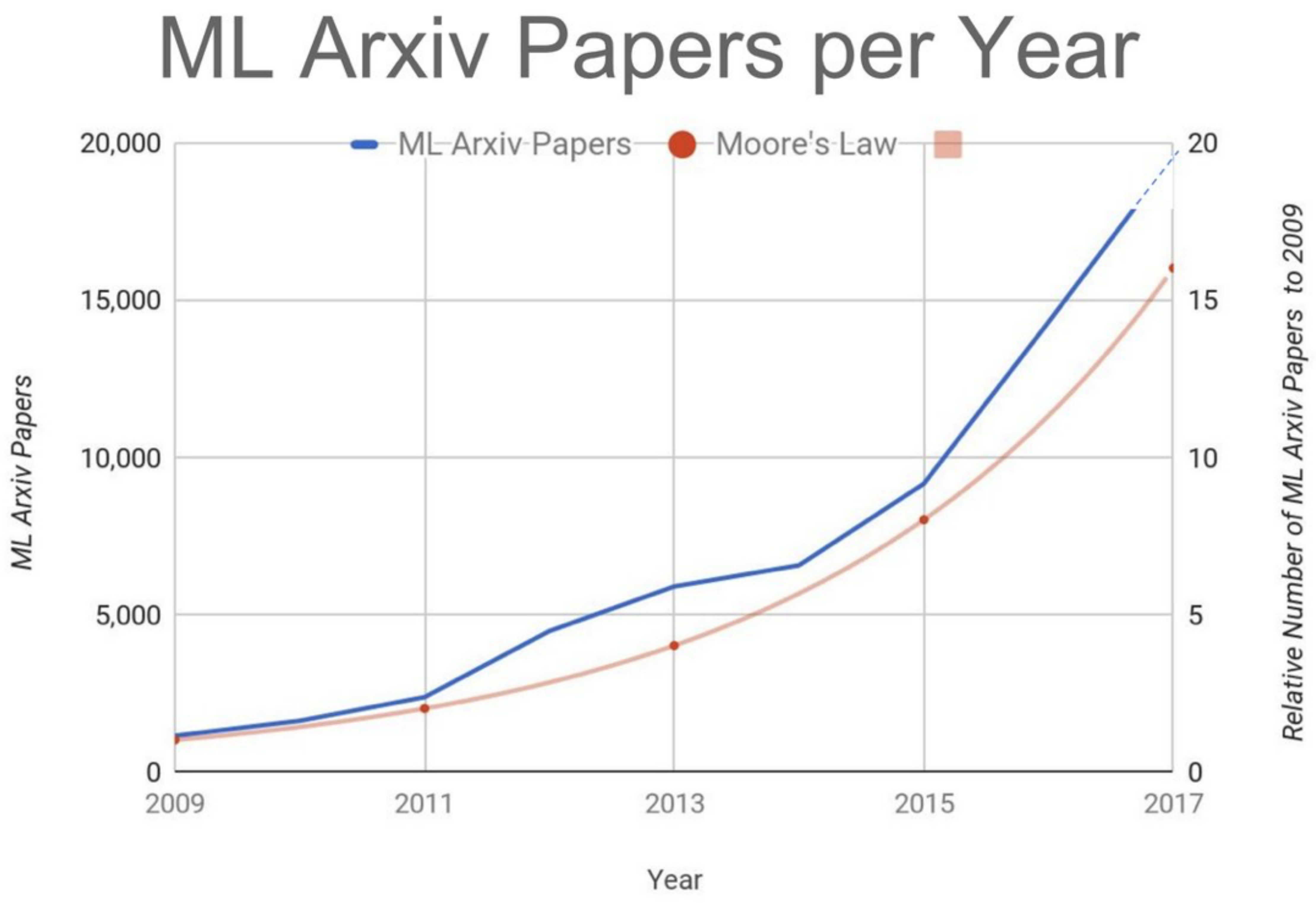
No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs

"There is no better data than more data"

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorit
  - –**Representation**
  - –**Evaluation**
  - –**Optimization**



Representation

- Separating Hyperplanes
- Support vectors
- Decision trees
- Sets of rules / Logic programs
- Instances (Nearest Neighbor)
- Graphical models (Bayes/Markov nets)
- Neural networks
- Model ensembles
- Etc.

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization

- Combinatorial optimization
  - E.g.: Greedy search, Dynamic programming
- Convex optimization
  - E.g.: Gradient descent, Coordinate descent
- Constrained optimization
  - E.g.: Linear programming, Quadratic programming

Gradient Descent

- if learning rate is too small, it'll converge very slowly
- if learning rate is too big, it'll diverge

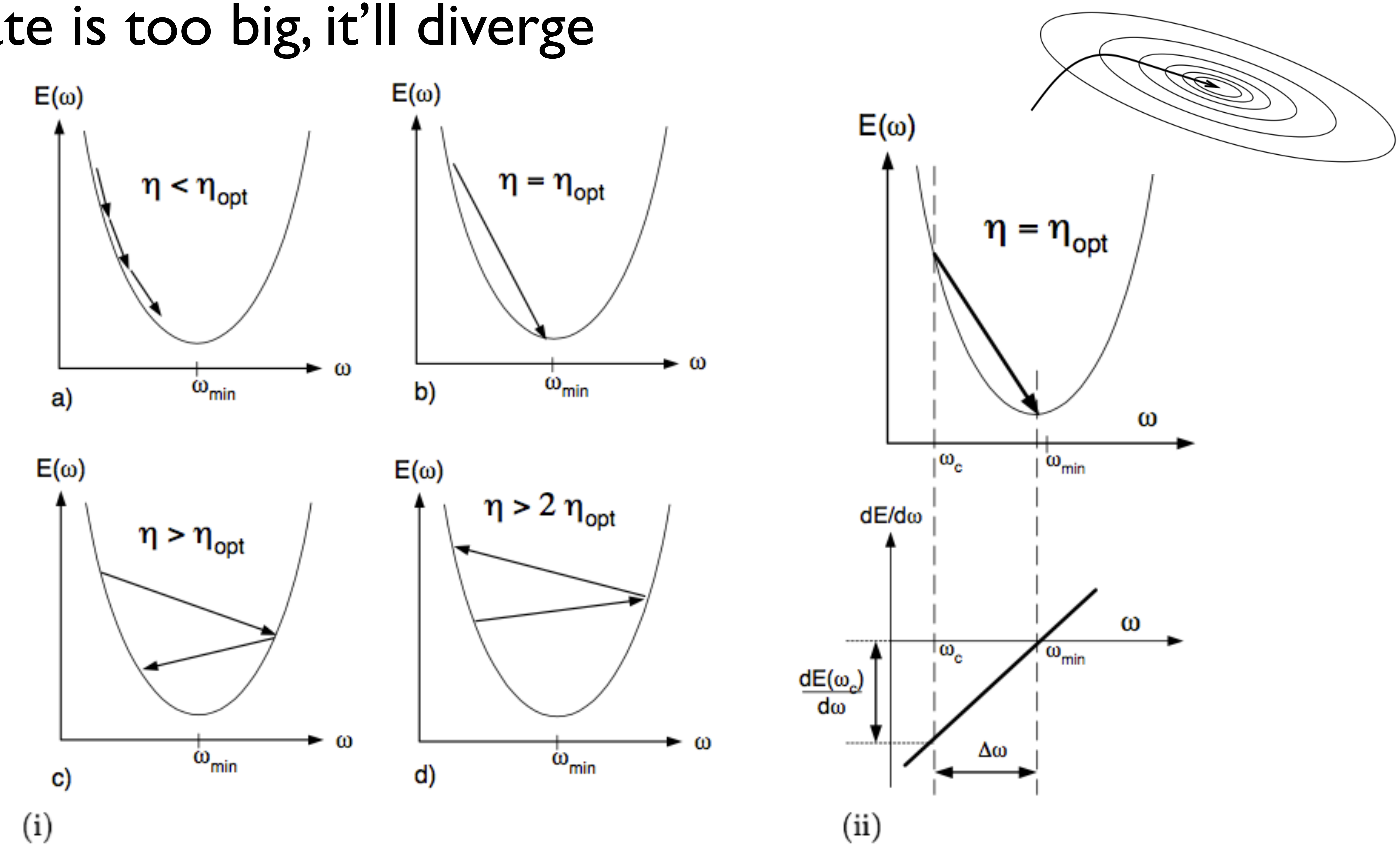


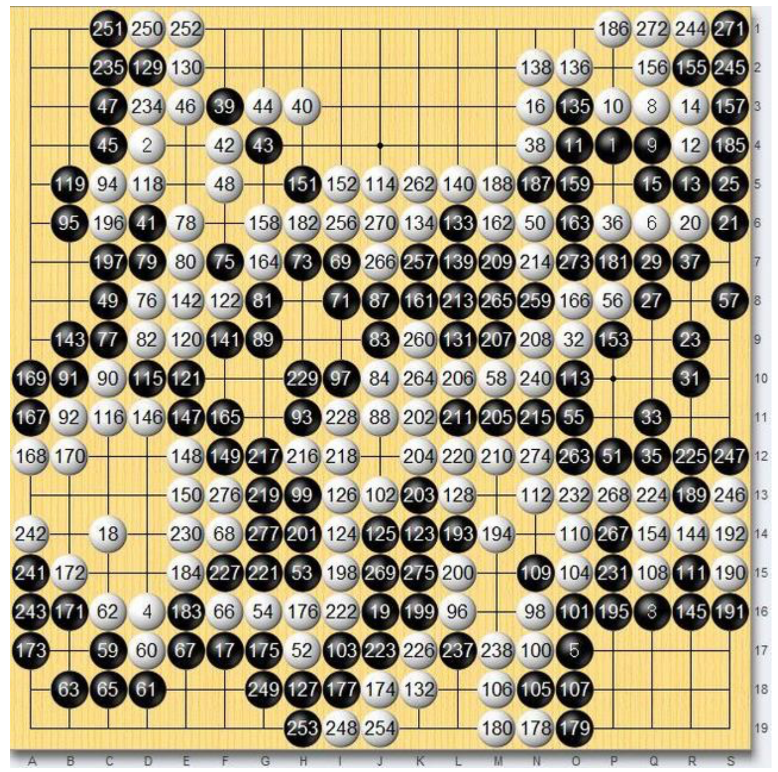
Fig. 6. Gradient descent for different learning rates.

Types of Learning

- Supervised (inductive) learning
  - Training data includes desired outputs
- Unsupervised learning
  - Training data does not include desired outputs
- Semi-supervised learning
  - Training data includes a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions

cat dog

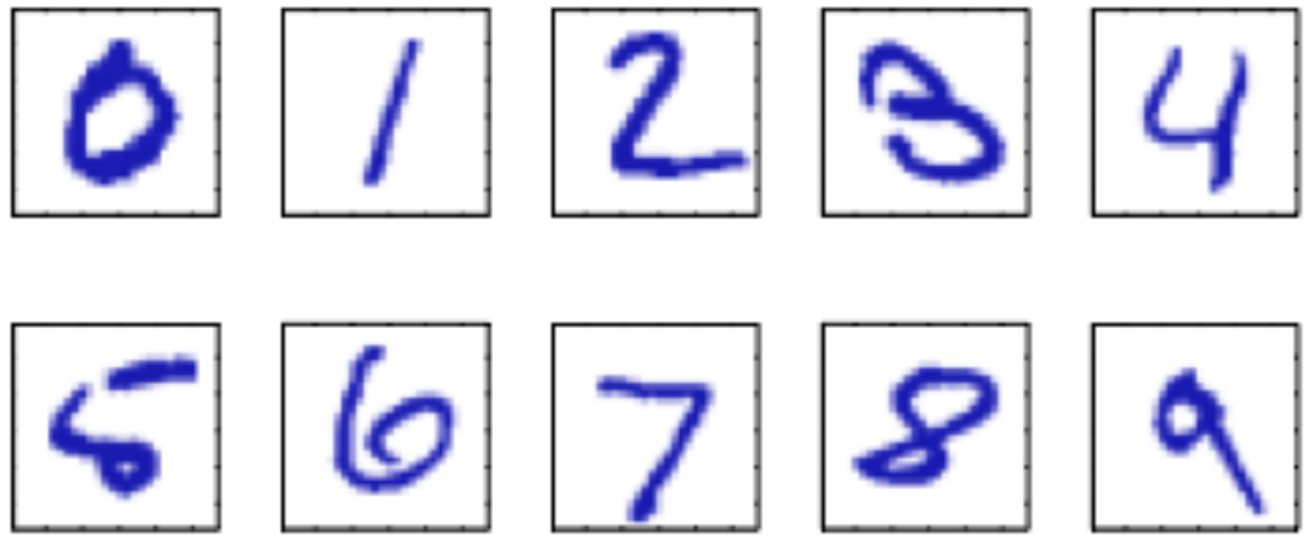
cat dog

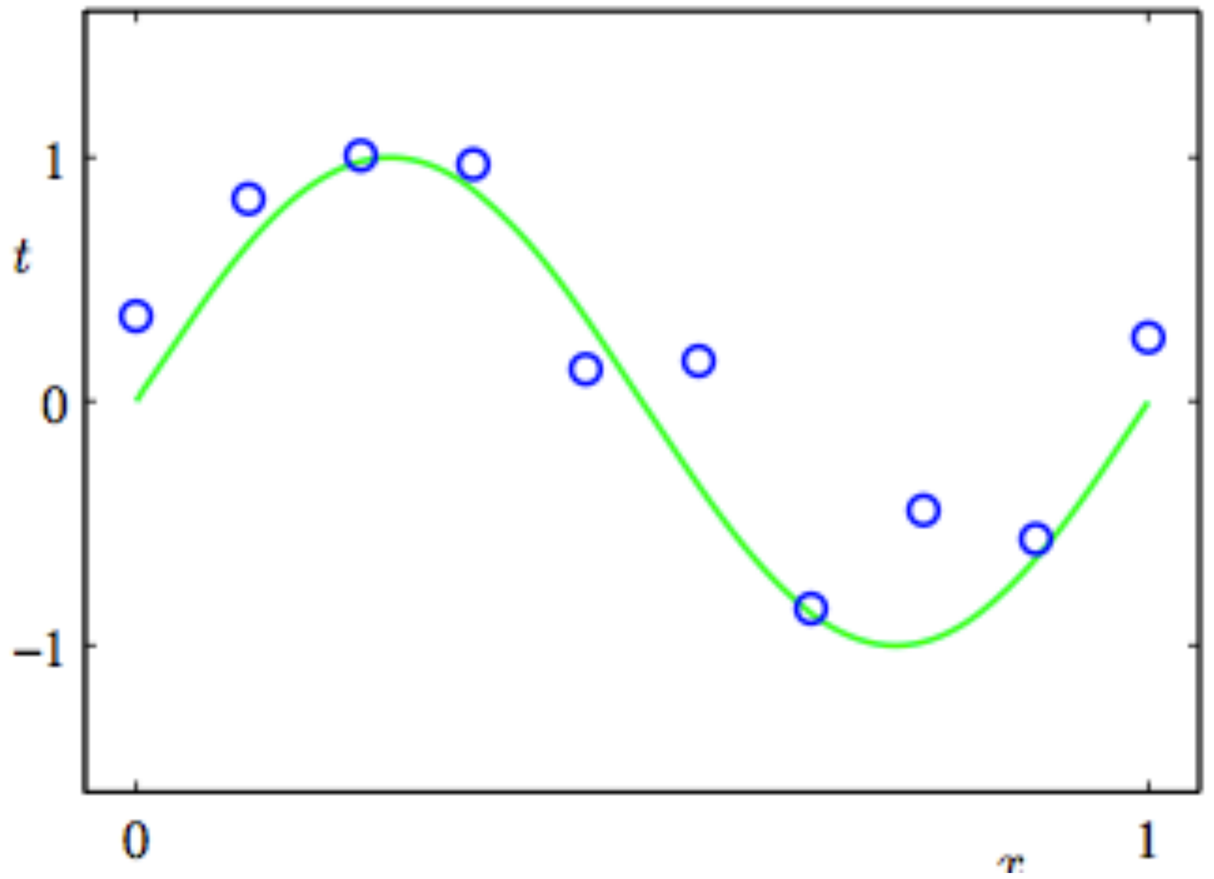


rules → **white win**

Supervised Learning

- Given examples (X, f(X)) for an unknown function f
- Find a good approximation of function f
  - Discrete f(X): Classification (binary, multiclass, structured)
  - Continuous f(X): Regression



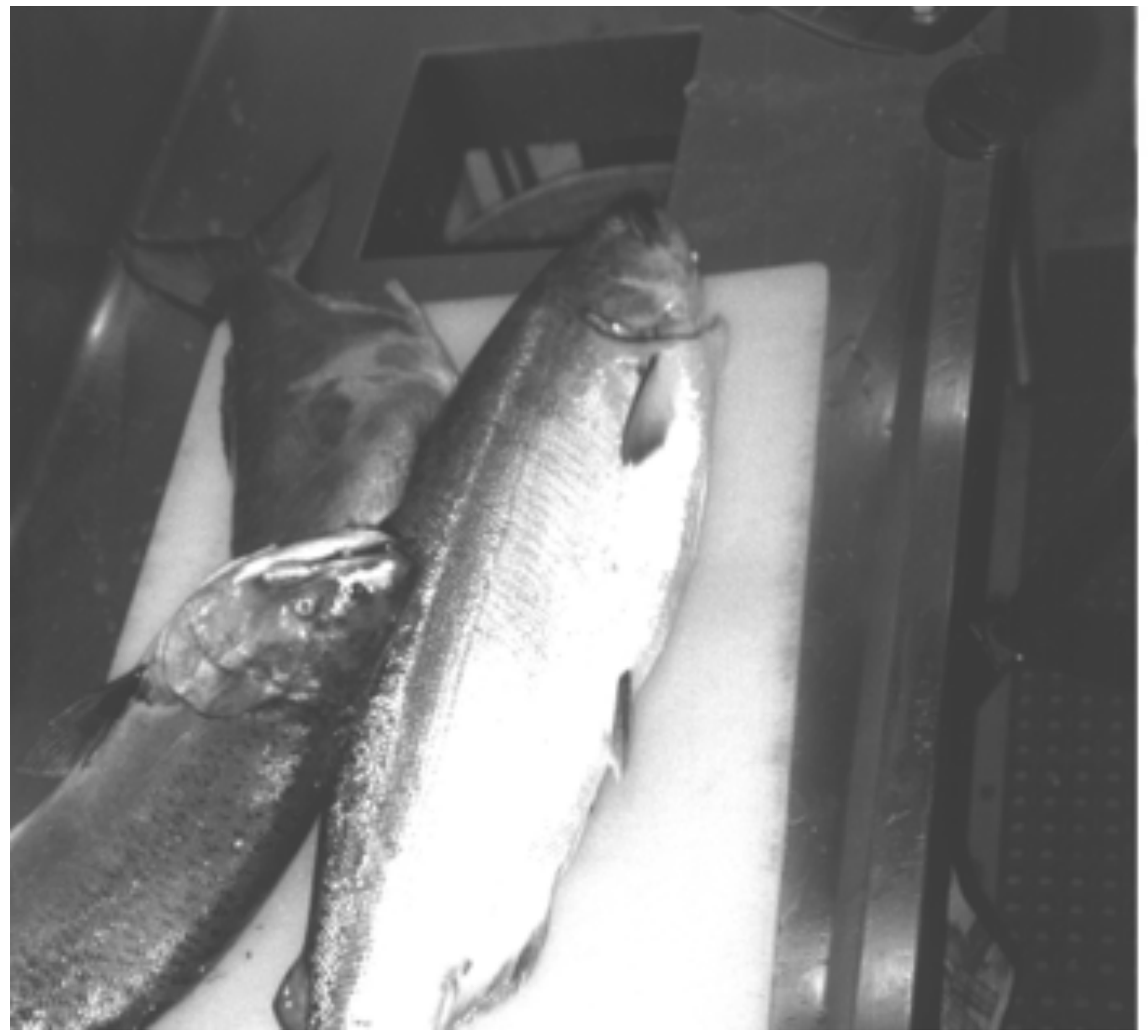


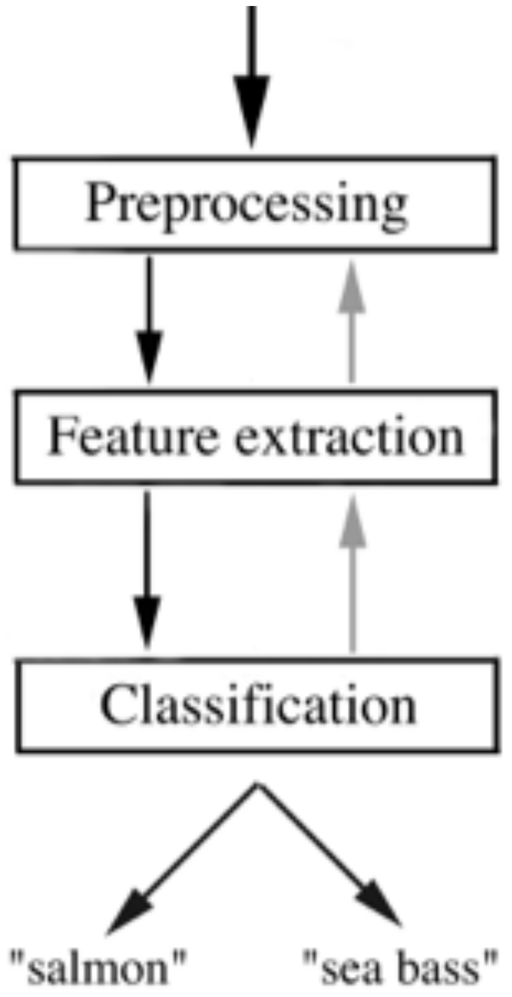
When is Supervised Learning Useful

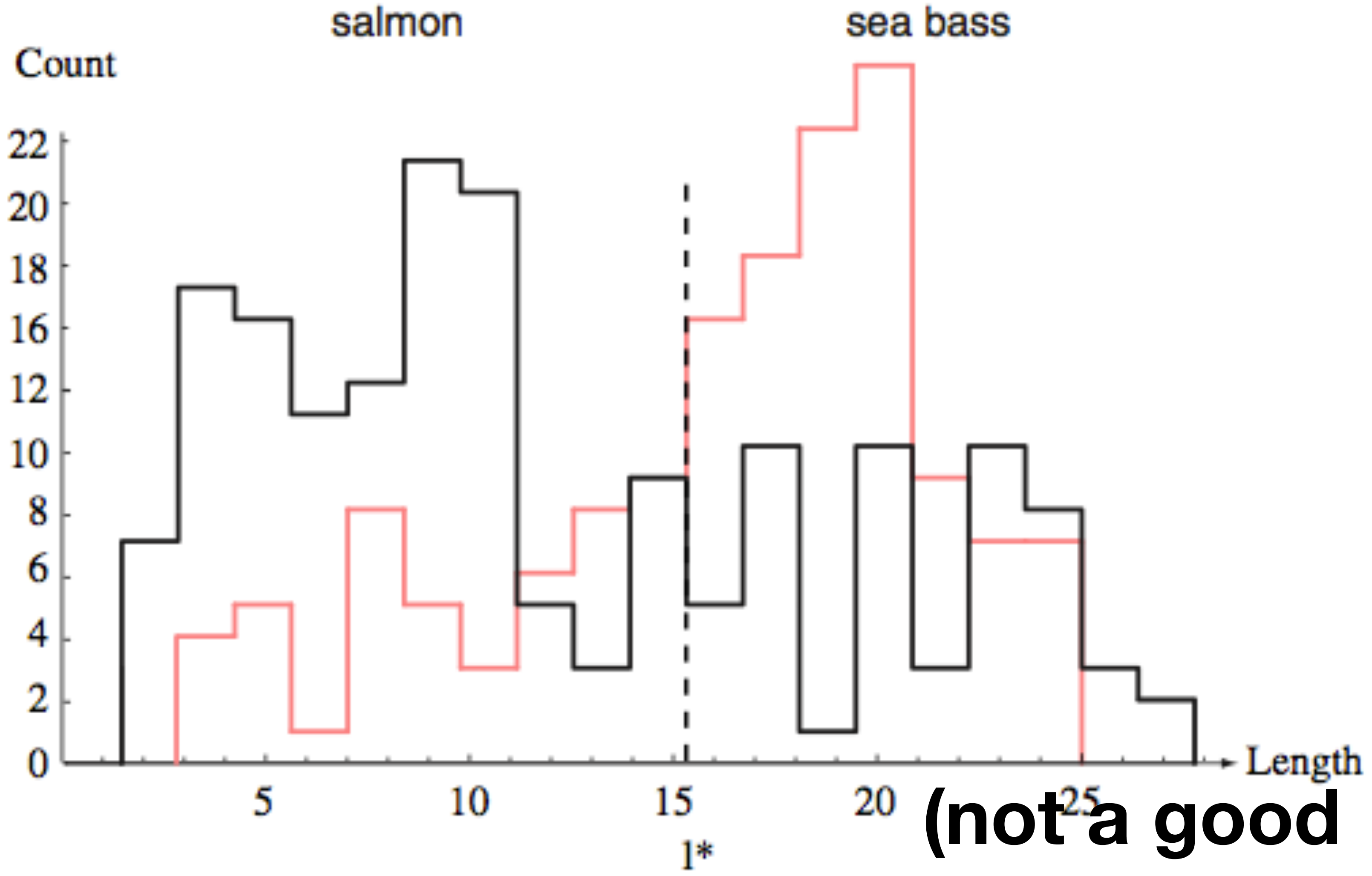
- when there is no human expert
  - input *x*: bond graph for a new molecule
  - output *f*(*x*): predicted binding strength to AIDS protease
- when humans can perform the task but can't describe it
  - computer vision: face recognition, OCR
- where the desired function changes frequently
  - stock price prediction, spam filtering
- where each user needs a customized function
  - speech recognition, spam filtering

Supervised Learning: Classification

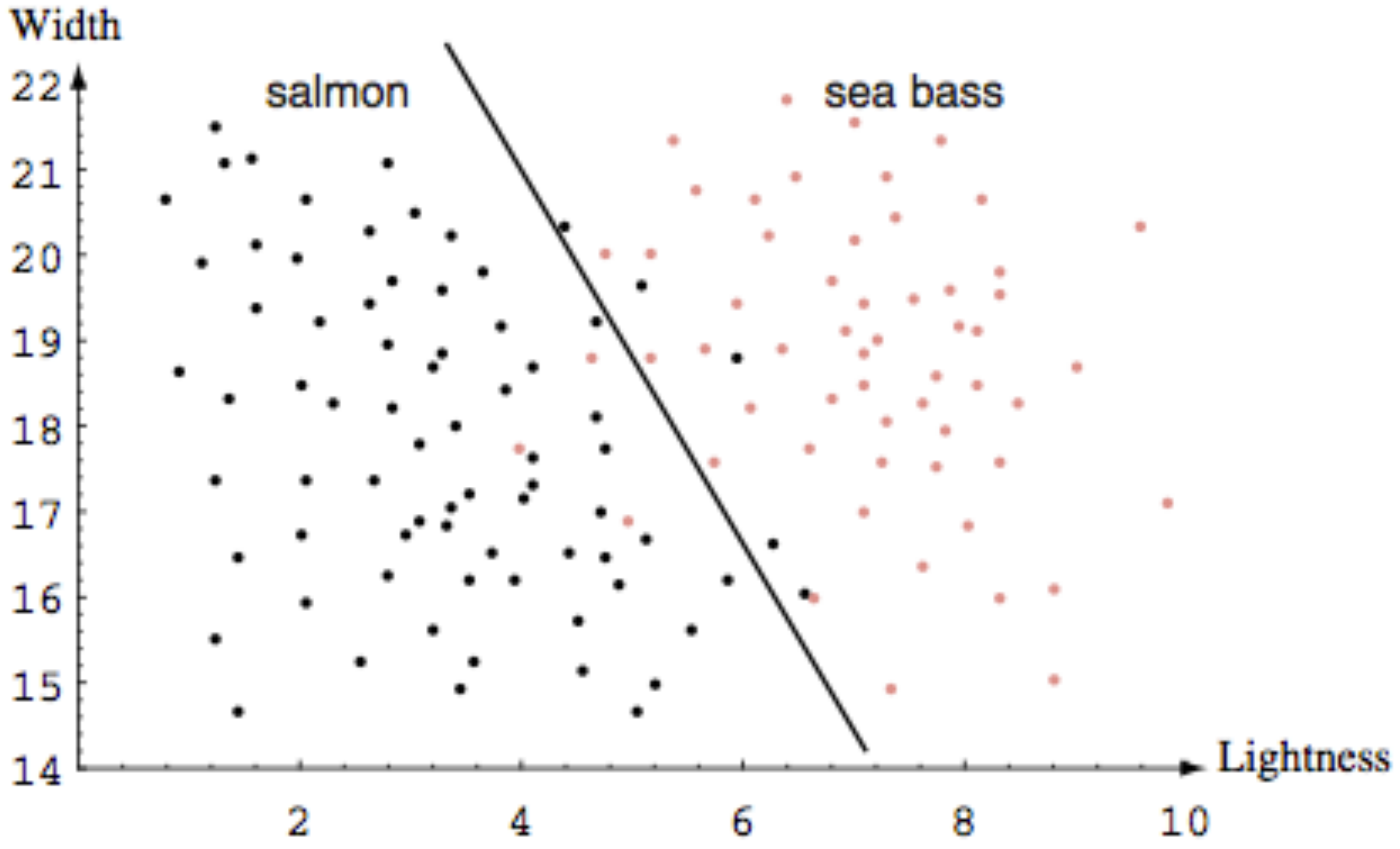
- input X: feature representation ("observation")

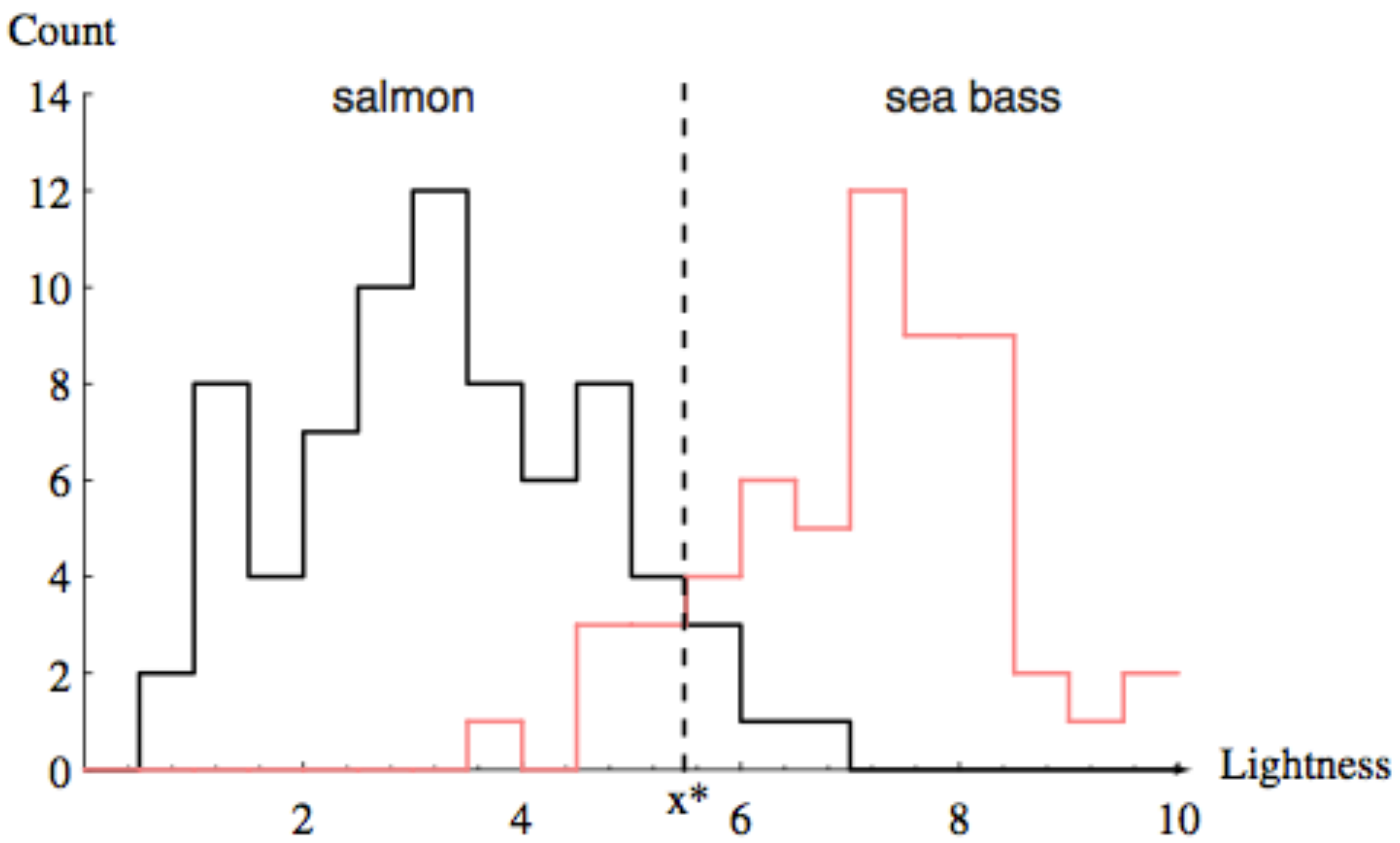






(not a good feature)

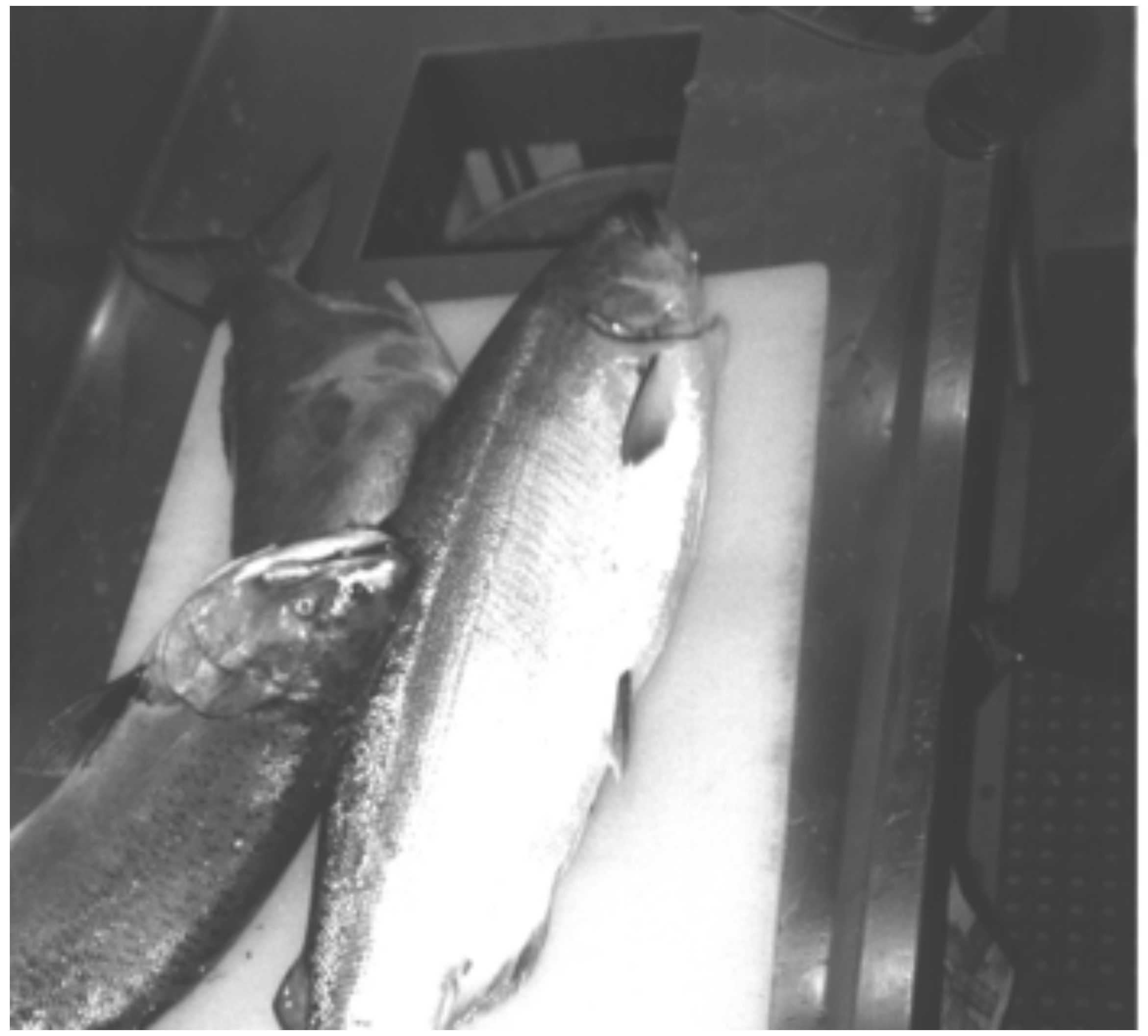


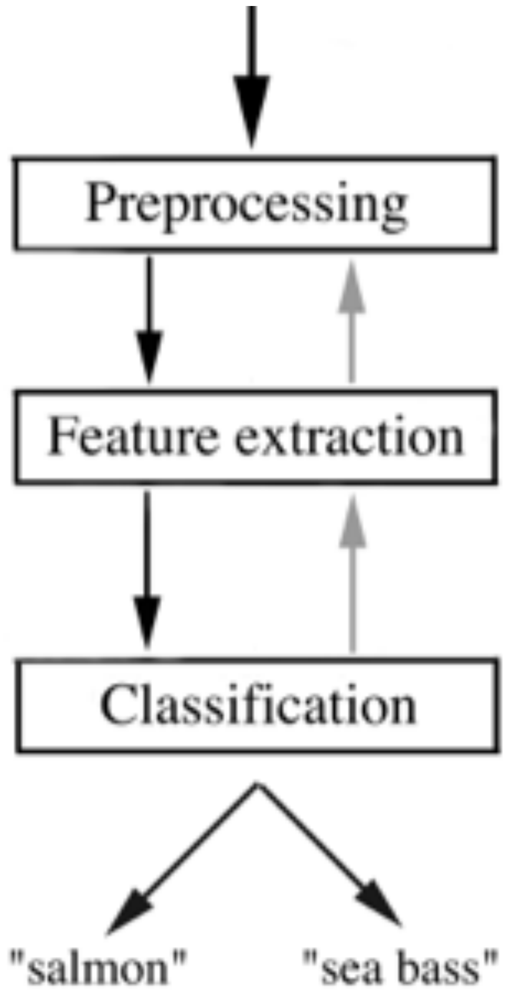


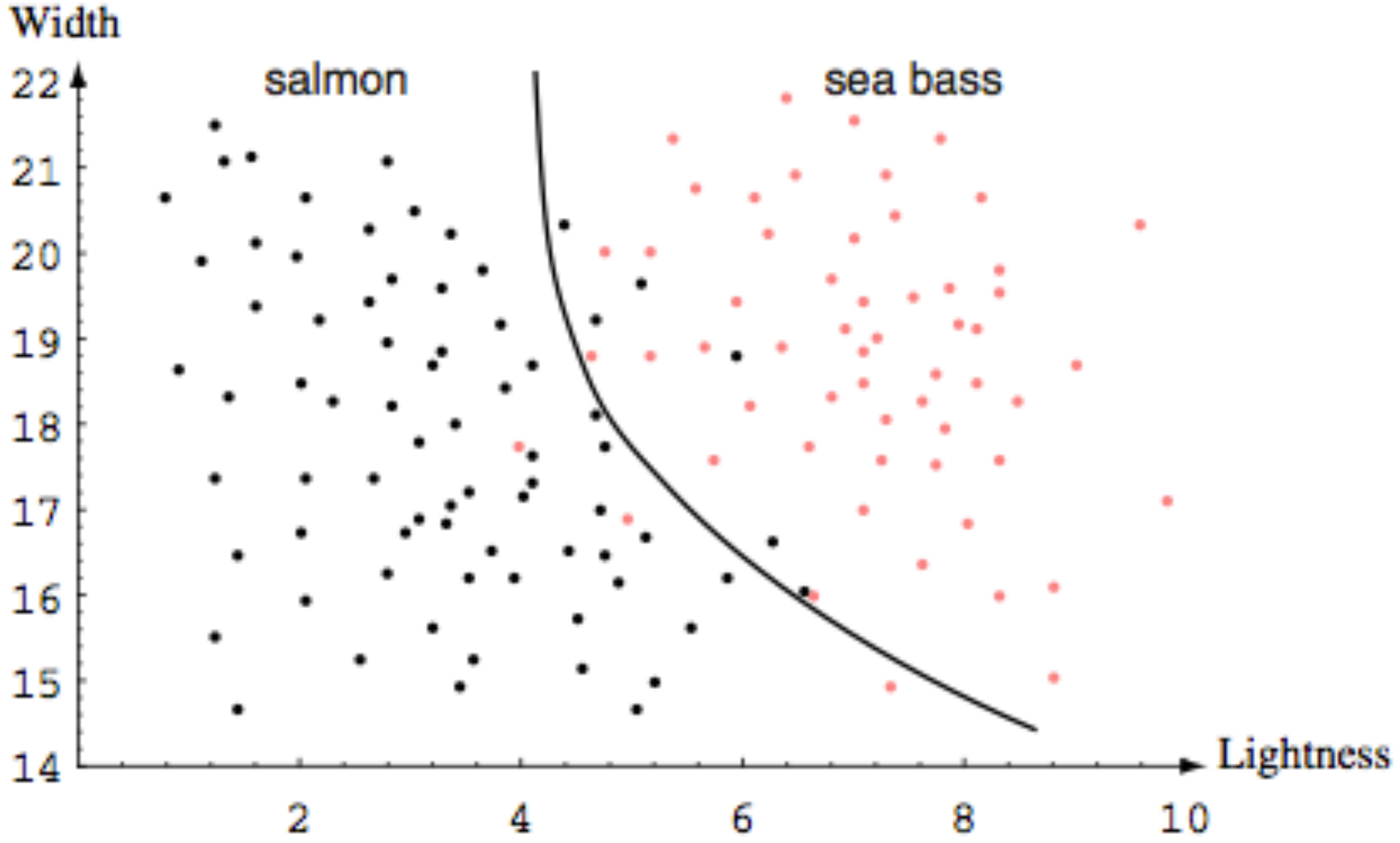
(a good feature)

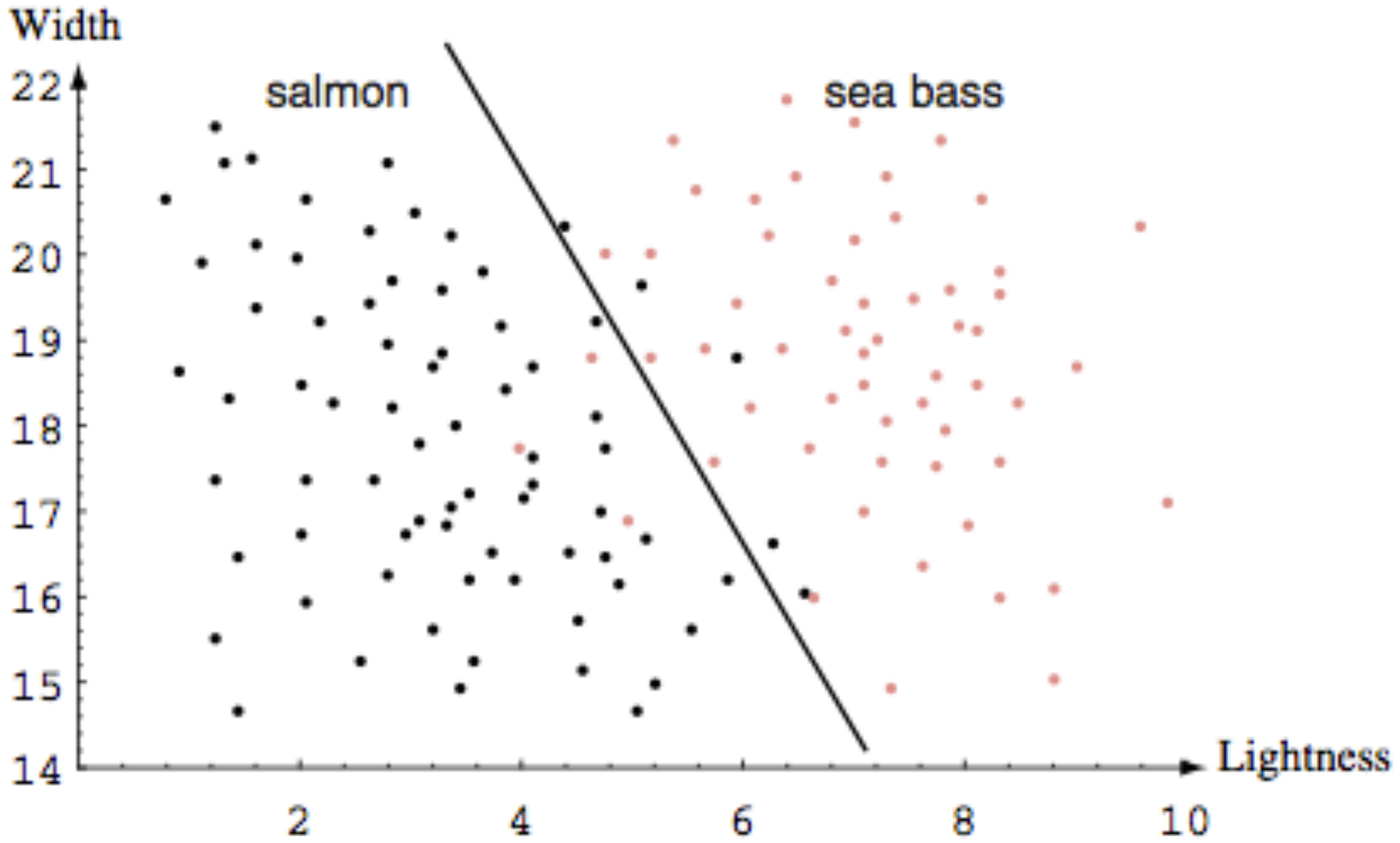
Supervised Learning: Classification

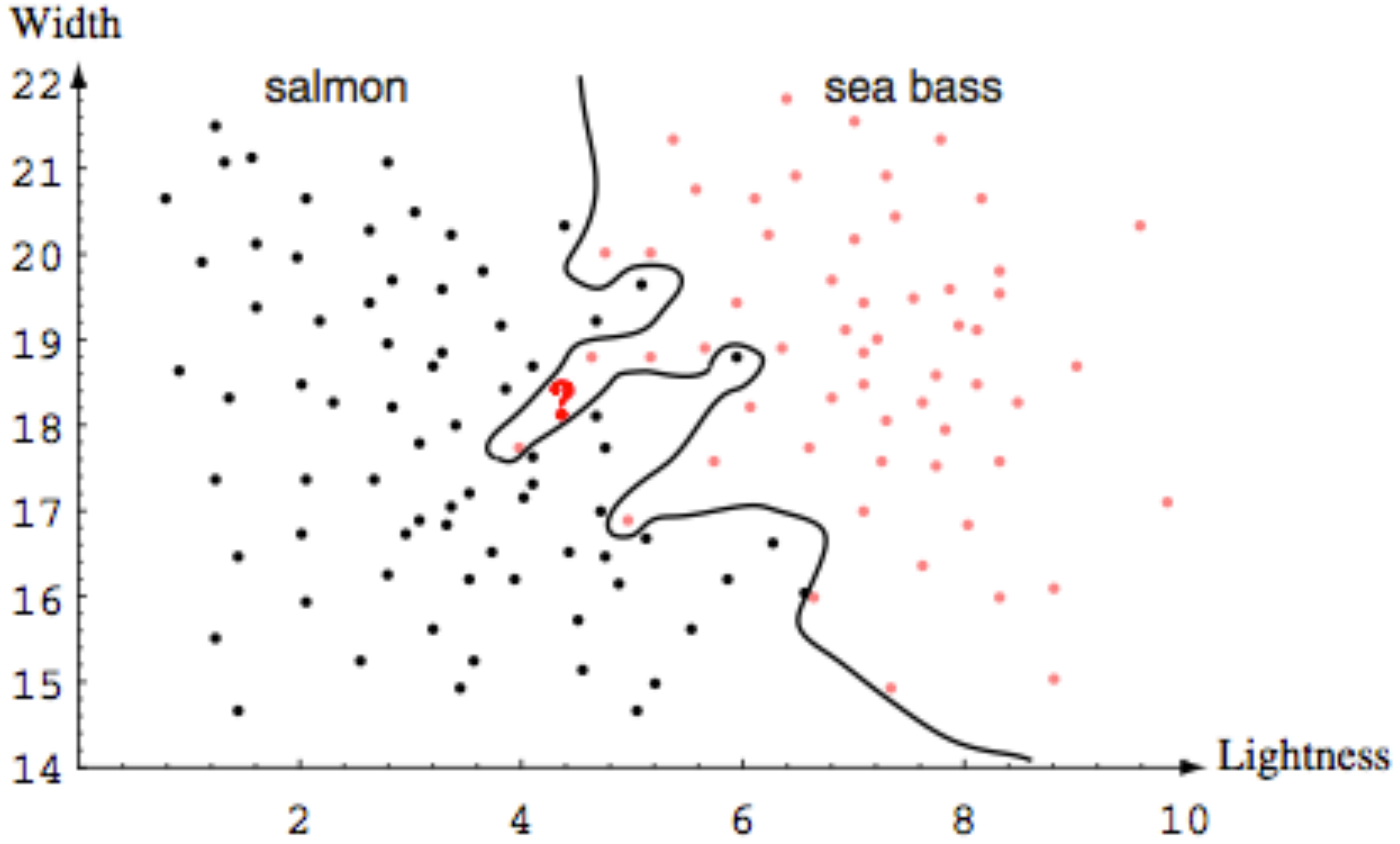
- input *X*: feature representation ("observation")





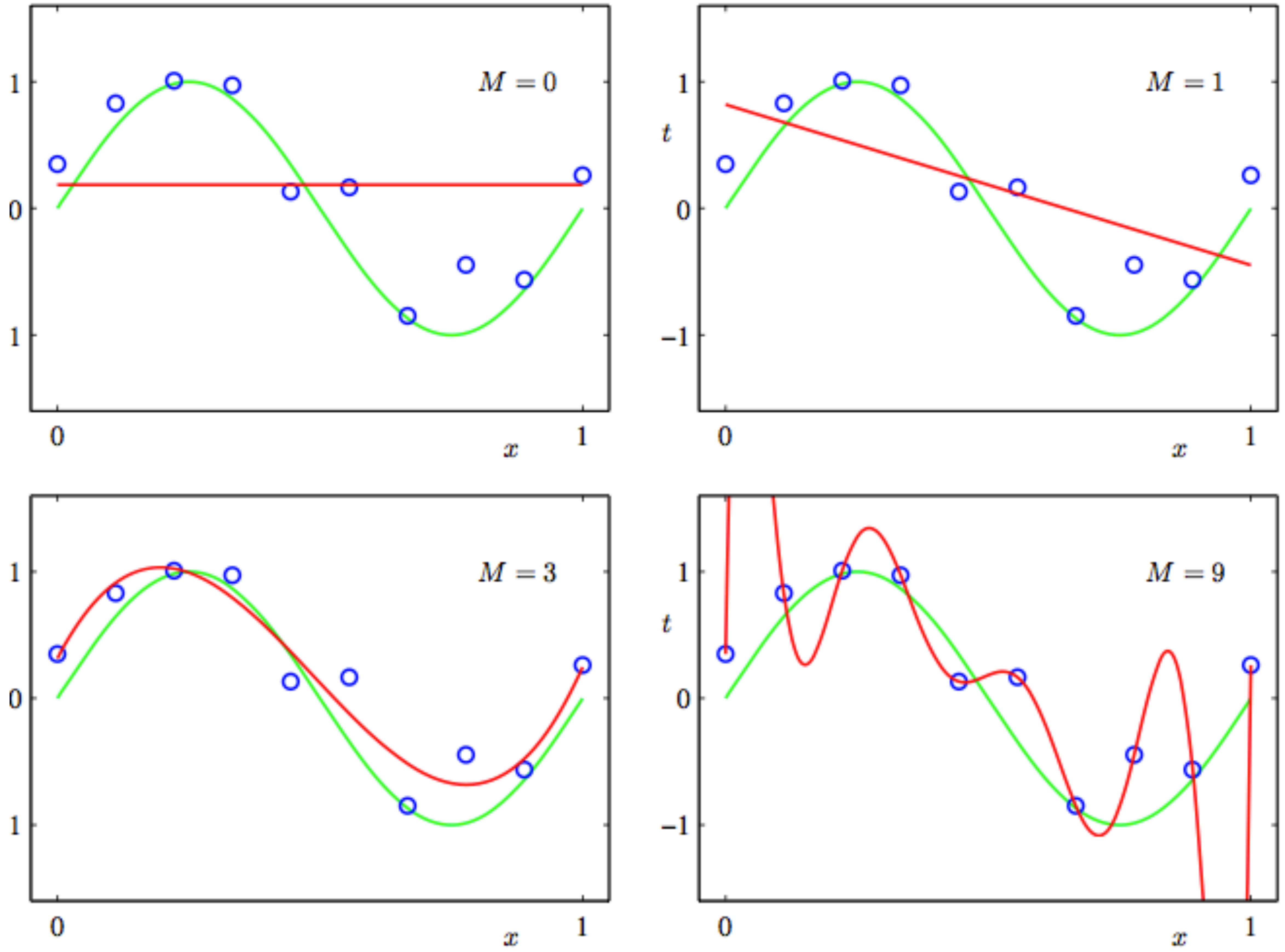






Supervised Learning: Regression

- linear and non-linear regression
- overfitting and underfitting (same as in classification)



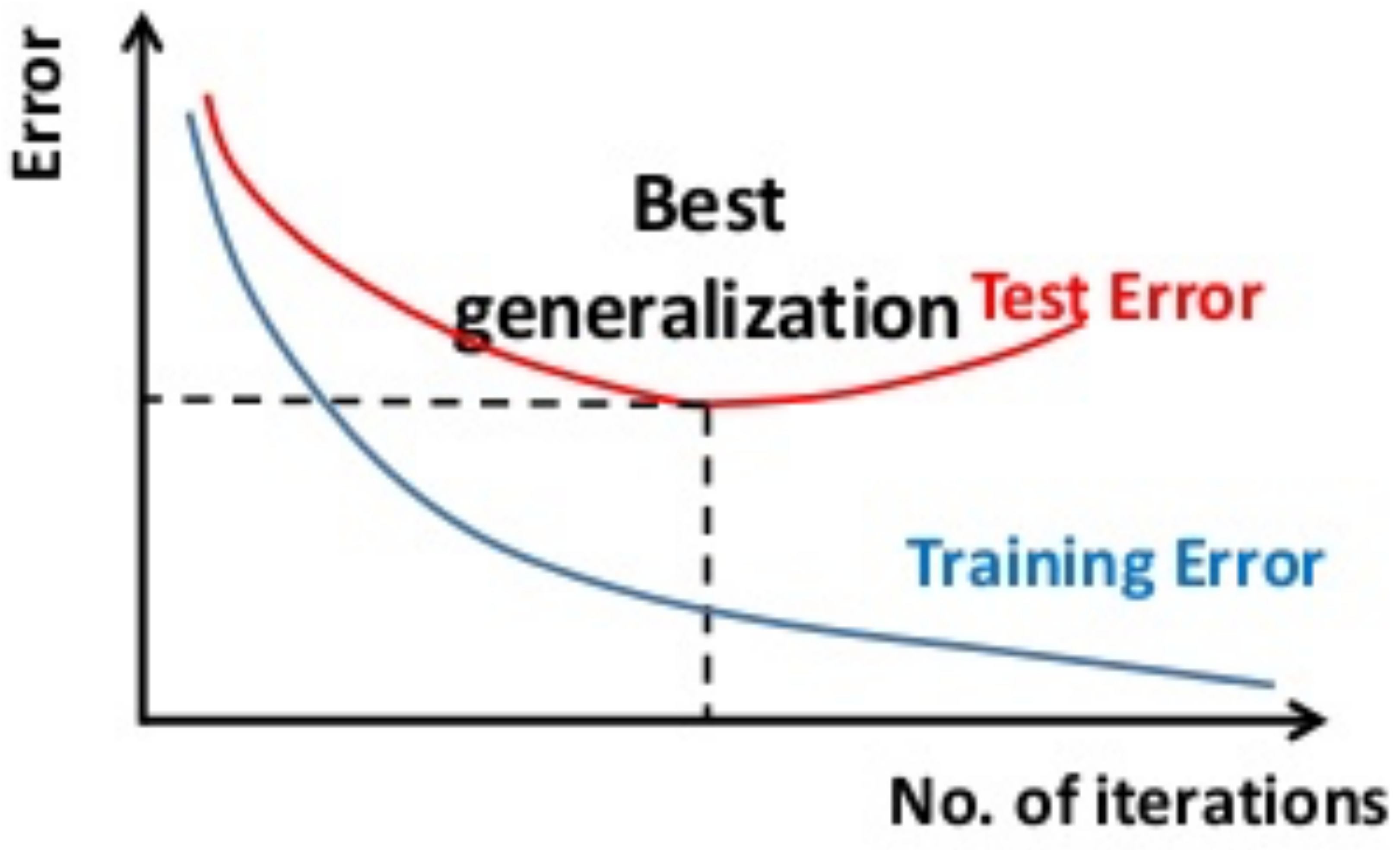
What We'll Cover (updated in 2019)

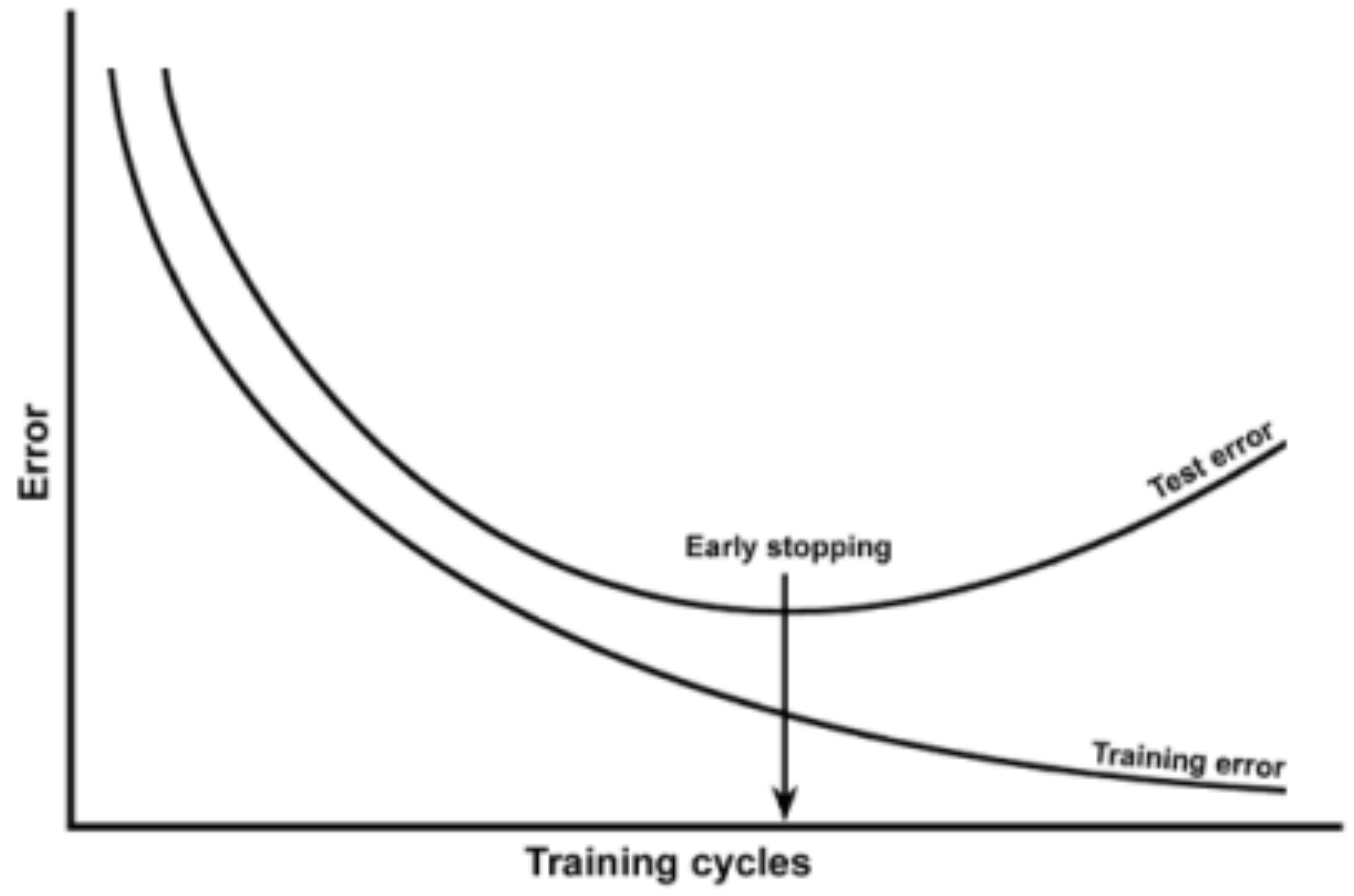
- Unit 1: Intro to ML, Nearest Neighbor Review of Linear Algebra, numpy, etc.
  - week 1: intro to ML, over/under-generalization, *k*-NN
  - week 2: tutorials on linear algebra, numpy, plotting, and data processing
- Unit 2: Linear Classification and Perceptron Algorithm
  - week 3: perceptron and convergence theory
  - week 4: perceptron extensions, practical issues, and logistic regression
- Unit 3 (weeks 5-6): Regression and Housing Price Prediction
- Unit 4 (weeks 7-8): Support Vector Machines and Kernels
- Unit 5 (weeks 9-10): Applications: Text Categorization and Sentiment Analysis

- Part III: Training, Test, and Generalization Errors; Underfitting and Overfitting; Methods to Prevent Overfitting; Cross-Validation and Leave-One-Out

Training, Test, & Generalization Errors

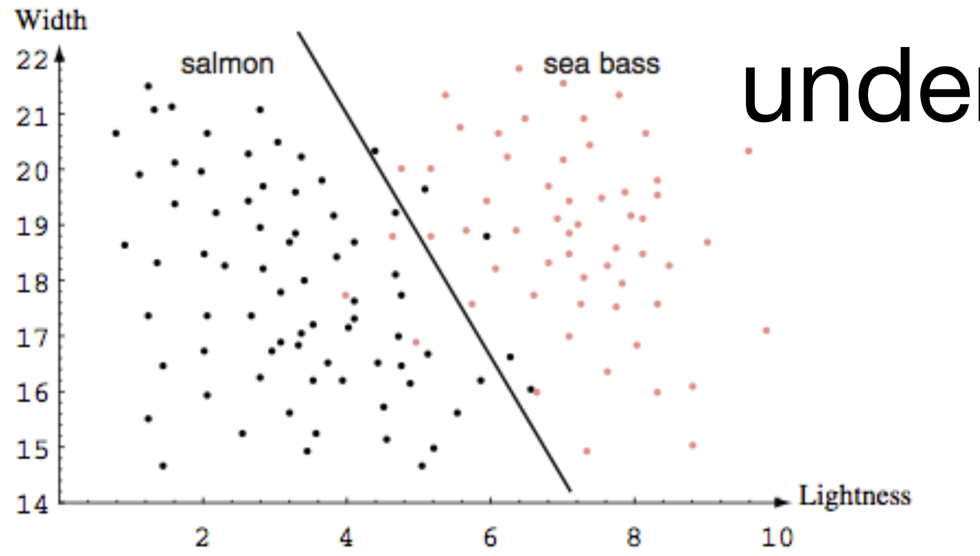
- in general, as training progresses, training error decreases
  - test error initially decreases, but eventually increases!
    - at that point, the model has overfit to the training data (memorizes noise or outliers)
- but in reality, you don't know the test data a priori ("blind-test")
  - generalization error: error on previously unseen data
  - expectation of test error assuming a test data distribution
  - often use a held-out set to simulate test error and do early stopping



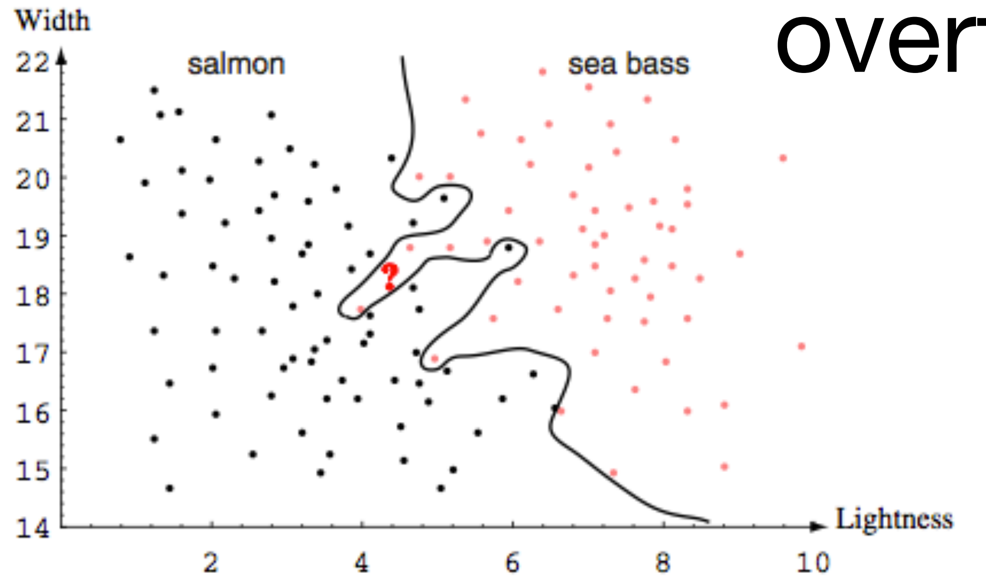


Under/Over-fitting due to Model

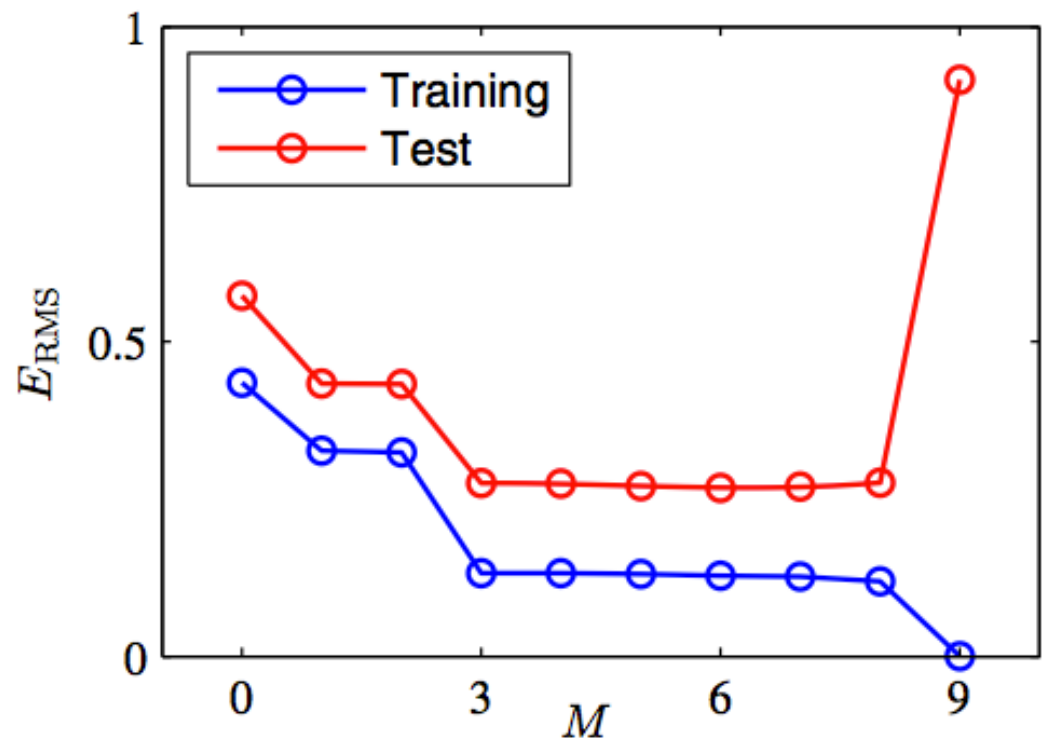
- underfitting / overfitting occurs due to under/over-training (last slide)
- underfitting / overfitting also occurs because of model complexity
  - underfitting due to oversimplified model ("*as simple as possible, but not simpler!*")
  - overfitting due to overcomplicated model (memorizes <u>noise</u> or <u>outliers</u> in data!)
    - extreme case: the model memorizes the training data, but no generalization!



underfitting



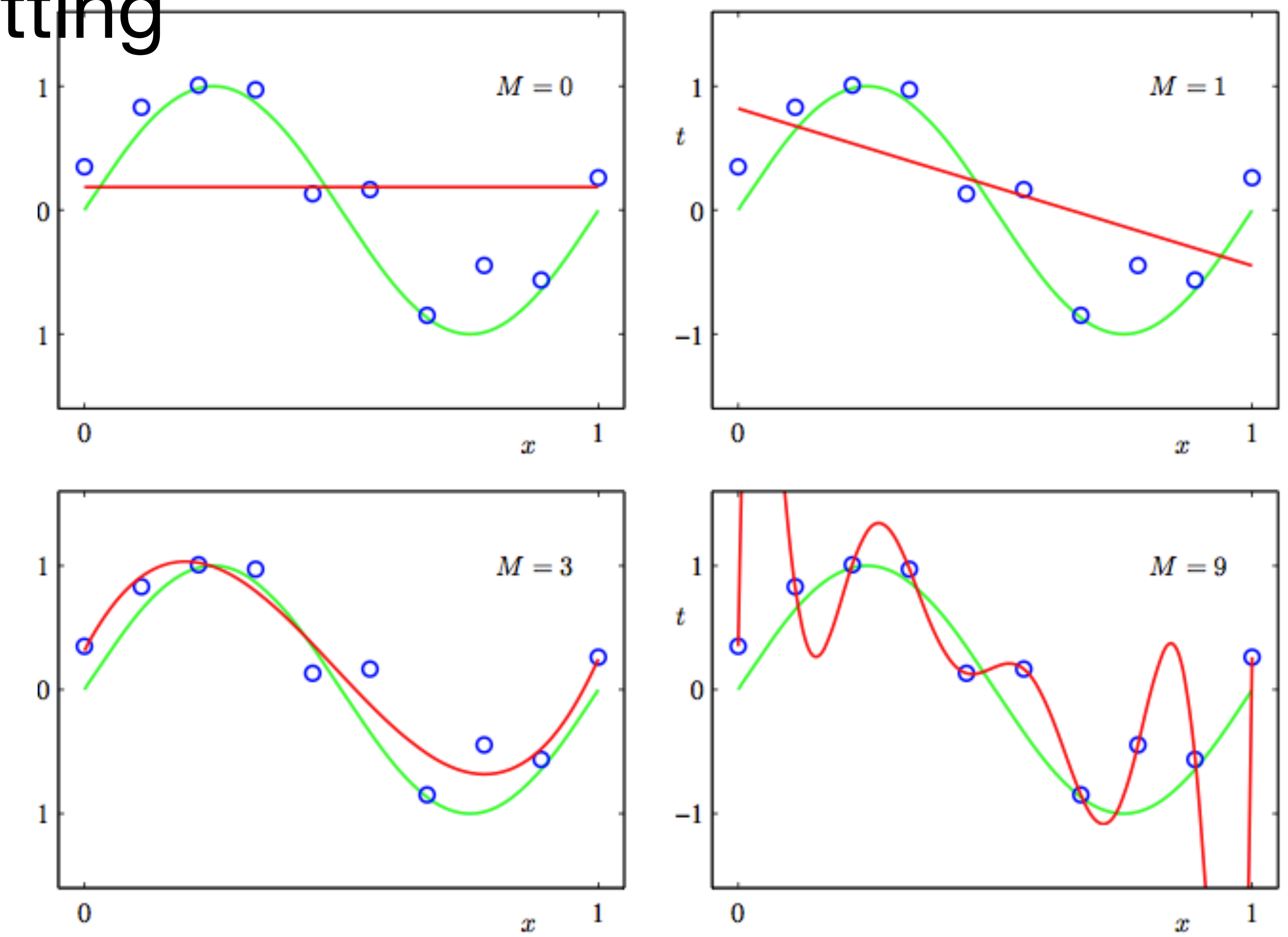
overfitting



(model complexity)

underfitting

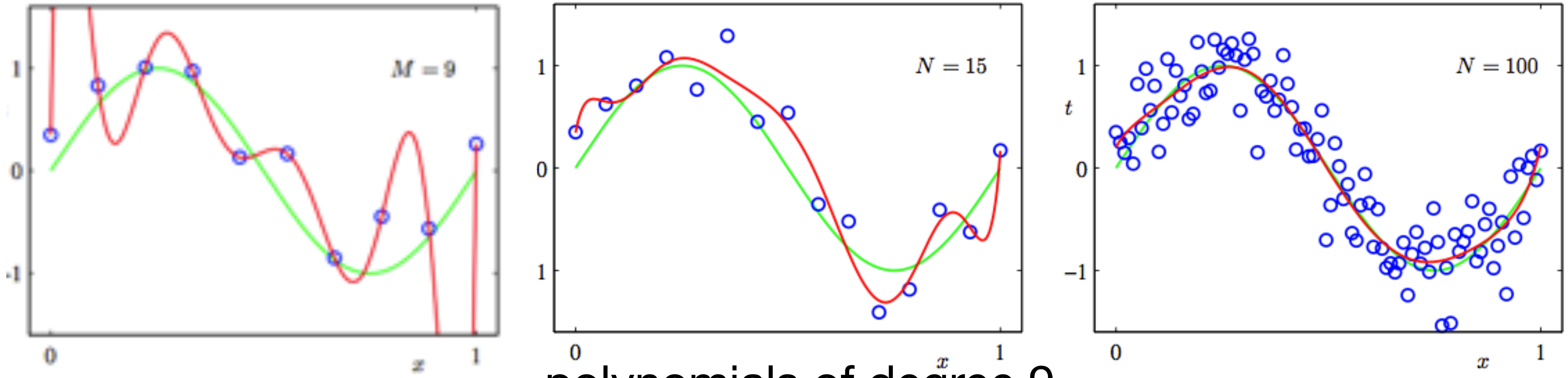
underfitting



overfitting

Ways to Prevent Overfitting

- use held-out training data to simulate test data (early stopping)
  - reserve a small subset of training data as “development set” (aka “validation set”, “dev set”, etc)
- regularization (explicit control of model complexity)
- more training data (overfitting is more likely on small data)
  - assuming same model complexity



polynomials of degree 9

Leave-One-Out Cross-Validation

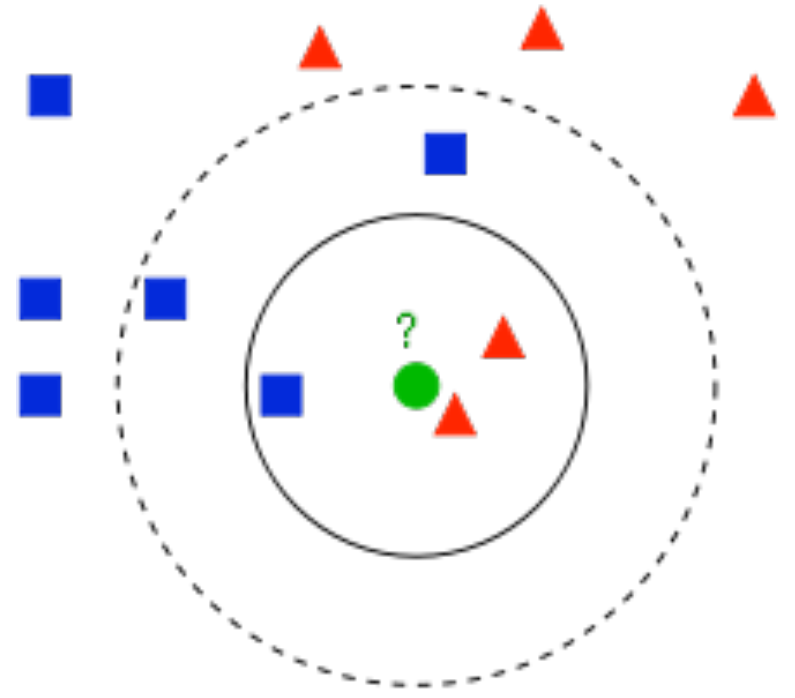
- what's the best held-out set?
  - random? what if not representative?
  - what if we use every subset in turn?
- leave-one-out cross-validation
  - train on all but the last sample, test on the last; etc.
  - average the validation errors
  - or divide data into N folds,
    train on folds 1..(N-1), test on fold N; etc.
- this is the best approximation of generalization error

This work by Sebastian Raschka is licensed under a
Creative Commons Attribution 4.0 International License.

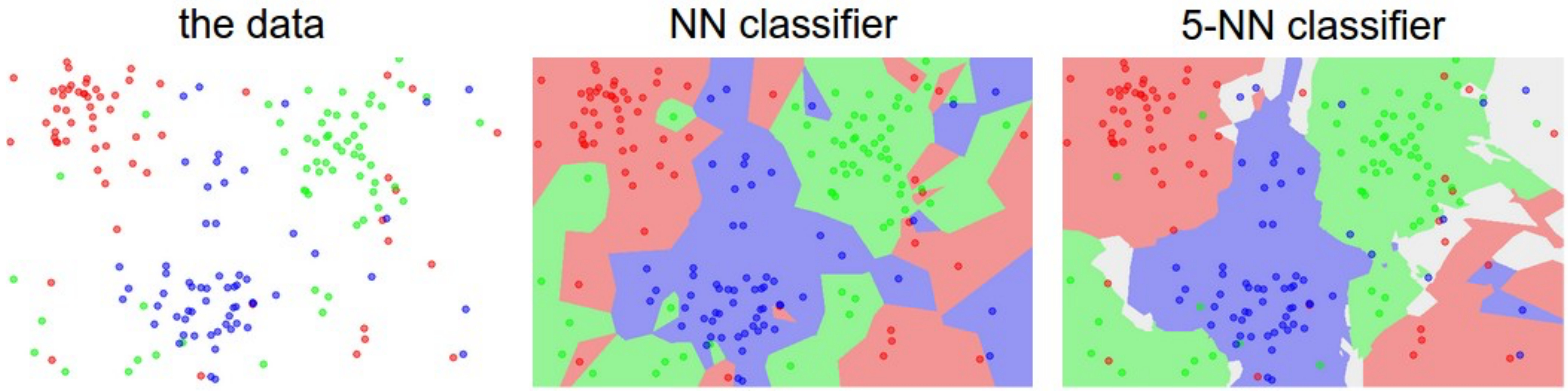
- Part IV: *k*-Nearest Neighbor Classifier

Nearest Neighbor Classifier

- for any test example *x*, assign its label using the majority vote of the closest neighbors of *x* in training set
  - extremely simple: <u>no training procedure!</u>
- 1-NN: extreme overfitting (extremely non-linear); *k*-NN is better
  - as *k* increases, the boundaries become smoother
    - *k*=+∞? majority vote (extreme underfitting!)



k=1: red
k=3: red
k=5: blue



Quiz Question

- what are the leave-one-out cross-validation errors for the following data set, using 1-NN and 3-NN?

(a) Consider the following data set with two real-valued inputs x (i.e. the coordinates of the points) and one binary output y (taking values + or -). We want to use k-nearest neighbours (K-NN) with Euclidean distance to predict y from x.

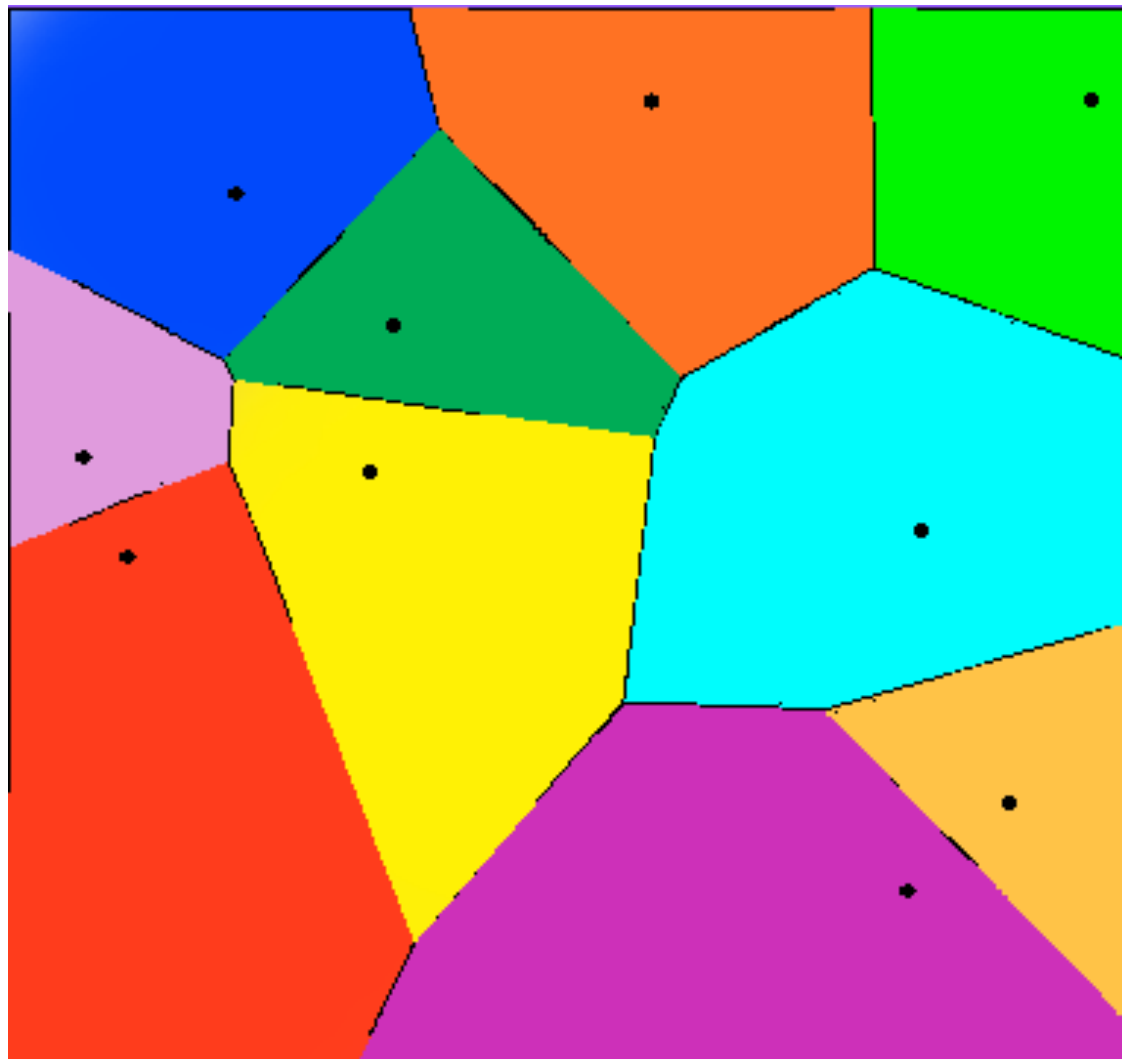
Calculate the leave-one-out cross-validation error of 1-NN on this data set. That is, for each point in turn, try to predict its label y using the rest of the points, and count up the number of misclassification errors.

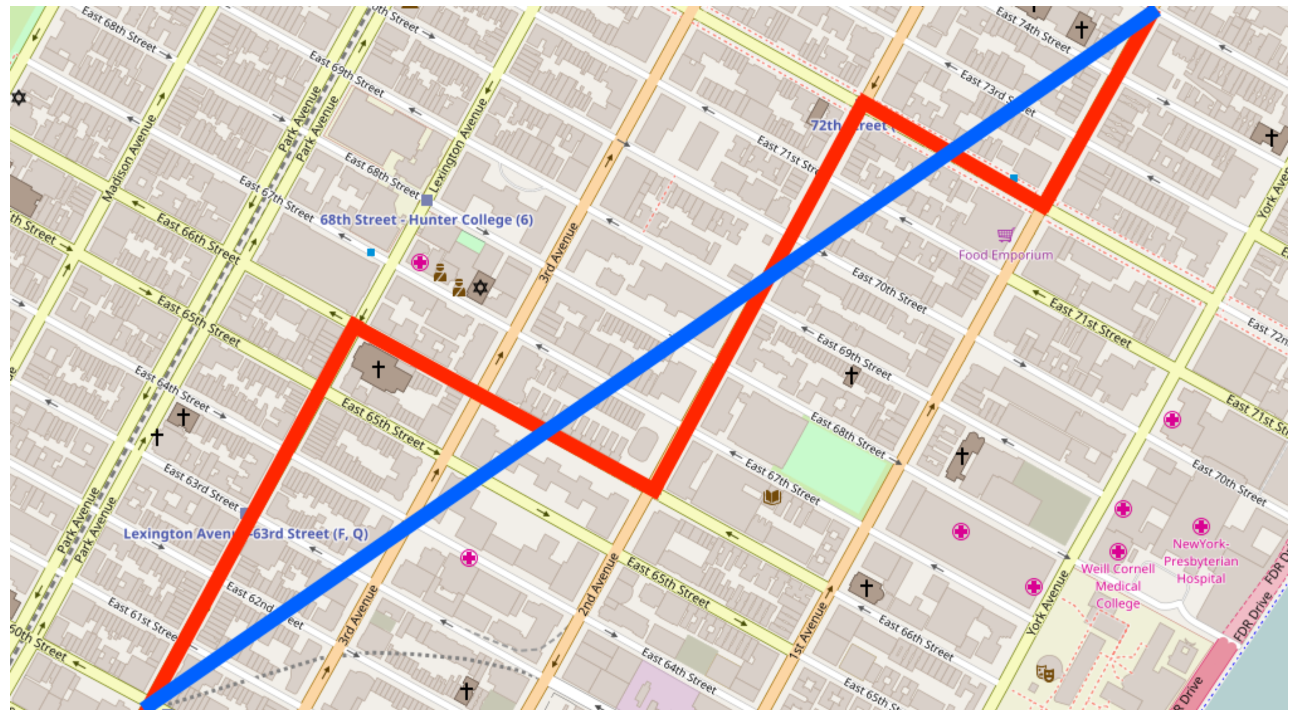
Ans: 1-NN: 5/10; 3-NN: 1/10

Euclidean vs. Manhattan Distances (added in 2019)

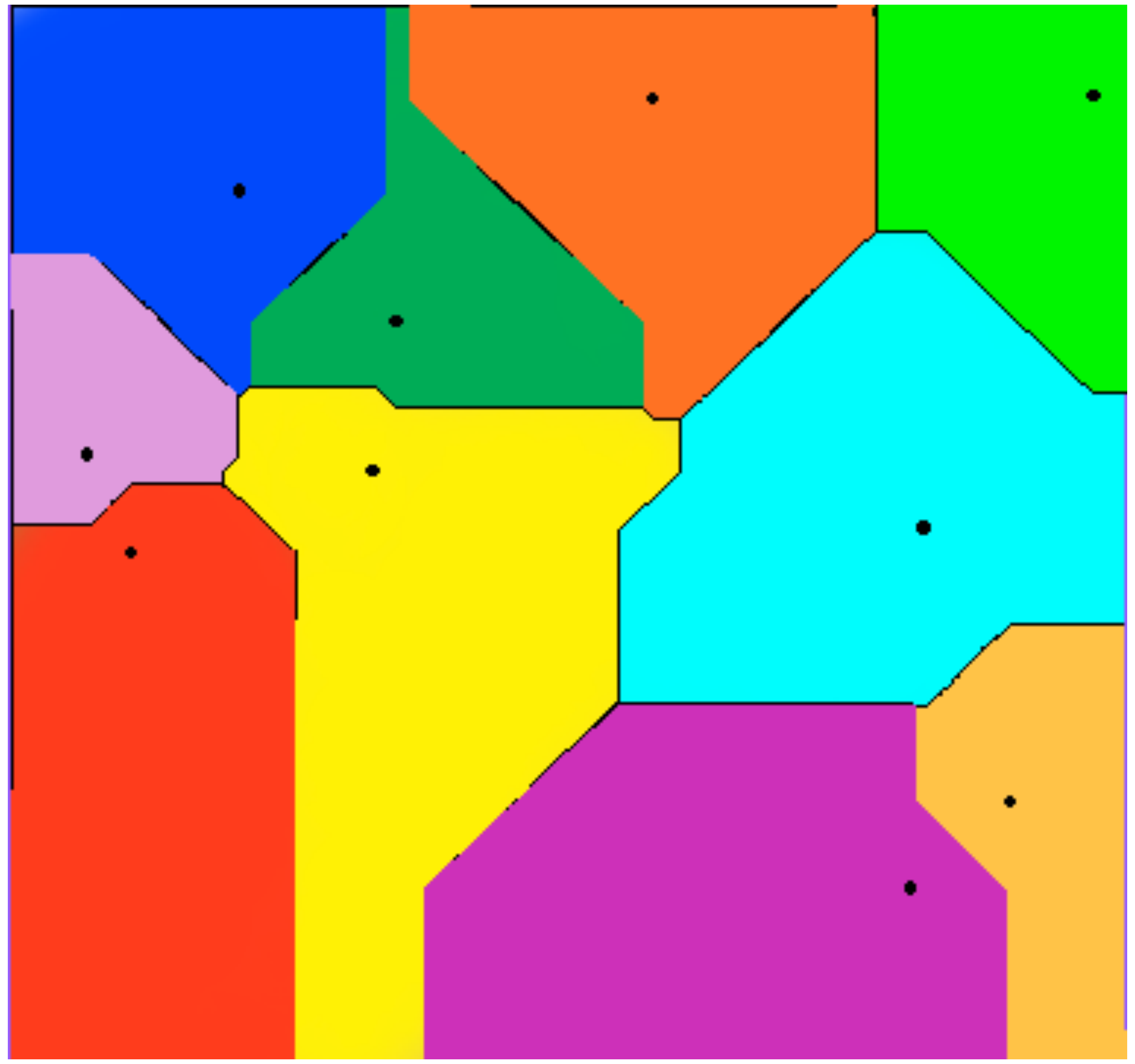
k-NN can use either Euclidean (default) or Manhattan distances
(both are special cases of ℓ_p-norm or Minkowski distance)

Euclidean Distance (ℓ_2-norm)





Manhattan Distance (ℓ_1-norm)



- Vectors $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$
- $\mathbf{L_p}$ norms or Minkowski distance:
$$L_p(x, y) = \left[|x_1 - y_1|^p + \cdots + |x_d - y_d|^p\right]^{1/p}$$
- $\mathbf{L_2}$ norm: Euclidean distance:
$$L_2(x, y) = \sqrt{|x_1 - y_1|^2 + \cdots + |x_d - y_d|^2}$$
- $\mathbf{L_1}$ norm: Manhattan distance:
$$L_1(x, y) = |x_1 - y_1| + \cdots + |x_d - y_d|$$
- $\mathbf{L_\infty}$ norm:(Chebyshev distance) L_p norms are known to be distance metrics
$$L_\infty(x, y) = \max\{|x_1 - y_1|, \dots, |x_d - y_d|\}$$

Bonus Track: Deep Learning (added in 2019)

- 2019 Turing Award (Nobel prize in CS) goes to the "big three" of deep learning
- deep neural nets born in mid-1980s (or as early as 1960s) with *backpropagation*
  - but it didn't work at that time, and quickly died out by mid-1990s
  - rebirth in 2006 (Hinton) and landmark win in 2012 (Hinton group's AlexNet on ImageNet)
- what changes in these ~30 years "suddenly" made it work?
  - according to Hinton: just a lot more data and computing power! (e.g. GPUs)
- rebranded as "deep learning" (which was controversial); super hot after 2012
- what's the difference between deep learning and pre-DL ML?
  - CS = automation;   ML = automating CS;   DL = automating ML = automation^3
  - you'll understand this around week 4; but this course will <u>not</u> teach DL per se

- Part V: viewing and processing HW1 data on the terminal

HW1: Adult Income >50K?

training/dev sets:

Age,	*Sector,*	*Education,*	*Marital_Status,*	*Occupation,*	*Race,*	*Sex,*	*Hours,*	*Country,*	***Target***
40,	Private,	Doctorate,	Married-civ-spouse,	Prof-specialty,	White,	Female,	60,	United-States,	>50K
44,	Local-gov,	Some-college,	Married-civ-spouse,	Exec-managerial,	Black,	Male,	38,	United-States,	>50K
55,	Private,	HS-grad,	Divorced,	Sales,	White,	Male,	40,	England,	<=50K

test data (semi-blind):

30,	Private,	Assoc-voc,	Married-civ-spouse,	Tech-support,	White,	Female,	40,	Canada,	???

- 2 numerical features: age and hours-per-week
  - option 1: keep them as numerical features
  - option 2: we can treat them as <u>binary</u> features
    - e.g., age=22, hours=38, ...
- 7 categorical features: convert to <u>binary</u> features
  - country, race, occupation, etc.
  - e.g., country=United_States, education=Doctorate,...

Interesting Facts in HW1 Data

- only ~25% positive (>50K); data was from 1994 (~$27K per capita)
- education is probably the single most important factor
  - education=Doctorate is extremely positive (80%)
  - education=Prof-school is also very positive (75%)
  - education=Masters is also positive (55%)
  - education=9th (high school dropout) is extremely negative (6%)
- "married" is good (45%), "never married" is extremely bad (5%)
- "self-emp-inc" is the best sector (59%), but "self-emp-not-inc" 30%
- hours-per-week=1 is 100% positive; country=Iran is 70% positive
- exec-managerial and prof-specialty are best occupations (48% / 46%)
- interesting combinations (e.g. "edu=Doc and sector=self-emp-inc": 100%)

Looking at HW1 data on terminal

- you are highly recommended to use Linux or Mac terminals
- basic familiarity with the terminal is a must for a data scientist!

```
$ cat income.train.txt.5k | cut -f 2 -d ','| sort | uniq -c
 150  Federal-gov
 340  Local-gov
3694  Private
 183  Self-emp-inc
 424  Self-emp-not-inc
 208  State-gov
   1  Without-pay
```

```
sector=Self-emp-inc: 59.02%
education=Masters: 55.38%
education=Prof-school: 74.70%
education=Doctorate: 80.00%
hours-per-week=99: 60.00%
hours-per-week=68: 100.00%
hours-per-week=1: 100.00%
country-of-origin=Taiwan: 58.33%
country-of-origin=Iran: 70.00%
country-of-origin=Cambodia: 66.67%
```

```
$ cat income.train.txt.5k | grep "Prof-spec" | wc -l
646
$ cat income.train.txt.5k | grep "Prof-spec" | grep -c ">"
294
```

```
$ cat income.train.txt.5k | sort -nk1 | head -1
17
$ cat income.train.txt.5k | sort -nk1 | tail -1
90
```

- Part VI-VII: data-preprocessing: binarization
  - please watch the videos (week1, parts 6-7) and try the ipython notebook from the course homepage
  - you should start working on HW1 as early as possible!