Data Mining MTAT.03.183

Introduction to Machine Learning

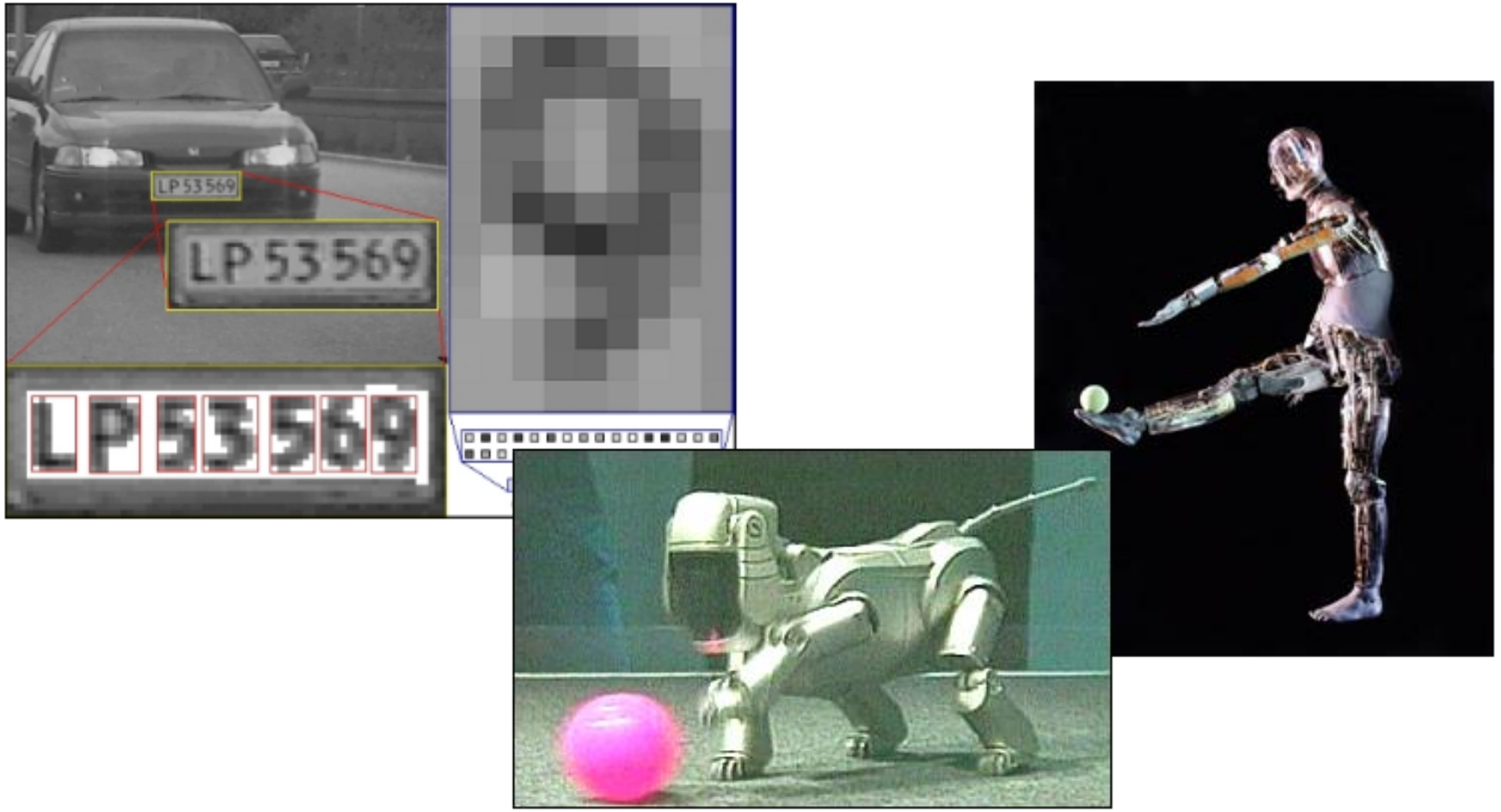
Jaak Vilo

2017 Spring

Acknowledgement

- Konstantin Tretyakov – 2009 course materials
- Lecture Notes for E Alpaydın 2010 Introduction to Machine Learning

- **Machine Learning course** (spring)
  - By Sven Laur et al.
  - 2017-2018 => Data Mining Fall, ML Spring
    - Meelis Kull


LP53569
LP 53 569
LP 53 569

Coming up next

- **"Machine learning"**
  - Terminology, foundations, general framework.
- **Supervised machine learning**
  - Basic ideas, algorithms & toy examples.
- **Statistical challenges**
  - P-values, significance, consistency, stability
- **State of the art techniques**
  - SVM, kernel methods, graphical models, latent variable models, boosting, bagging, LASSO, on-line learning, deep learning, reinforcement learning, …

Learning Associations

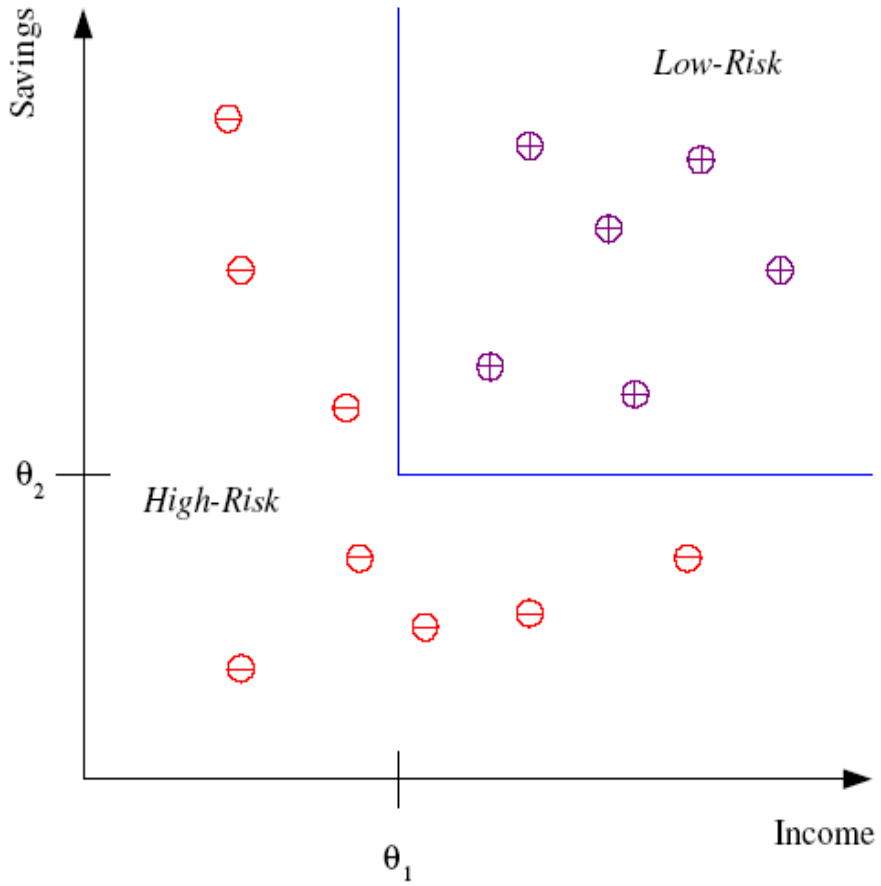
- Basket analysis:

  $P(Y \mid X)$ probability that somebody who buys X also buys Y where X and Y are products/services.

  Example: $P(\text{chips} \mid \text{beer}) = 0.7$

Classification

- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their *income* and *savings*



Discriminant: IF *income* > θ_1 AND *savings* > θ_2

THEN low-risk ELSE high-risk

Classification: Applications

- Aka Pattern recognition
- Face recognition: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency.
- Medical diagnosis: From symptoms to illnesses
- Biometrics: Recognition/authentication using physical and/or behavioral characteristics: Face, iris, signature, etc
- ...

Face Recognition

Training examples of a person

Test images

ORL dataset,
AT&T Laboratories, Cambridge UK

Regression

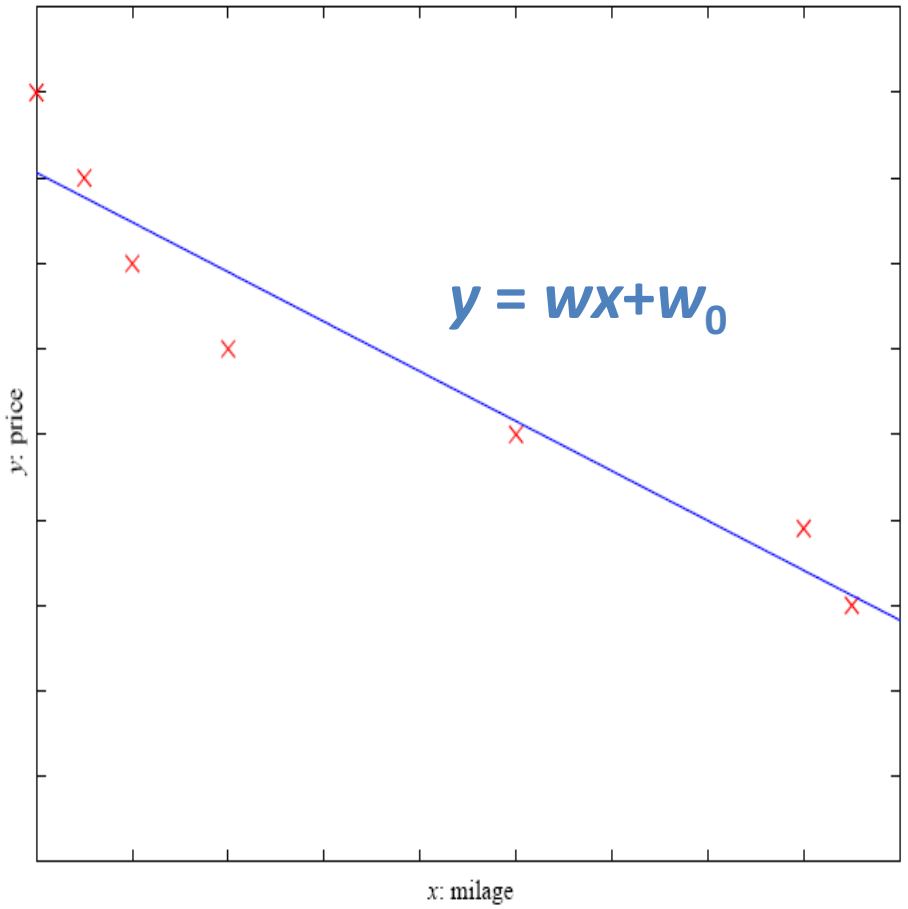
- Example: Price of a used car
- x : car attributes

  y : price

  $$y = g(x \mid \theta)$$

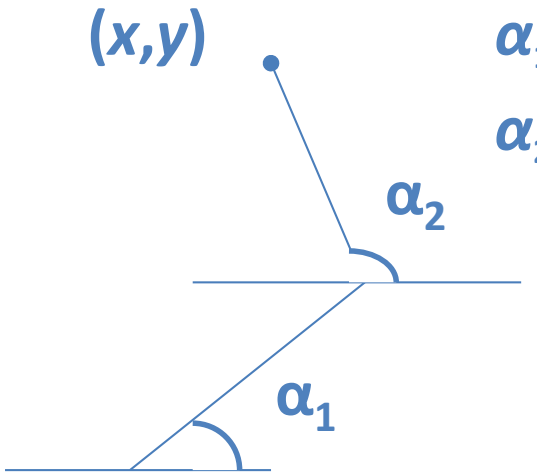
  $g(\ )$ model,

  θ parameters



Regression Applications

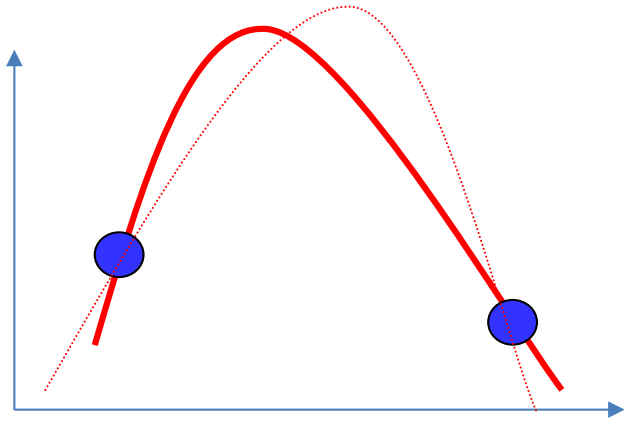
- Navigating a car: Angle of the steering
- Kinematics of a robot arm

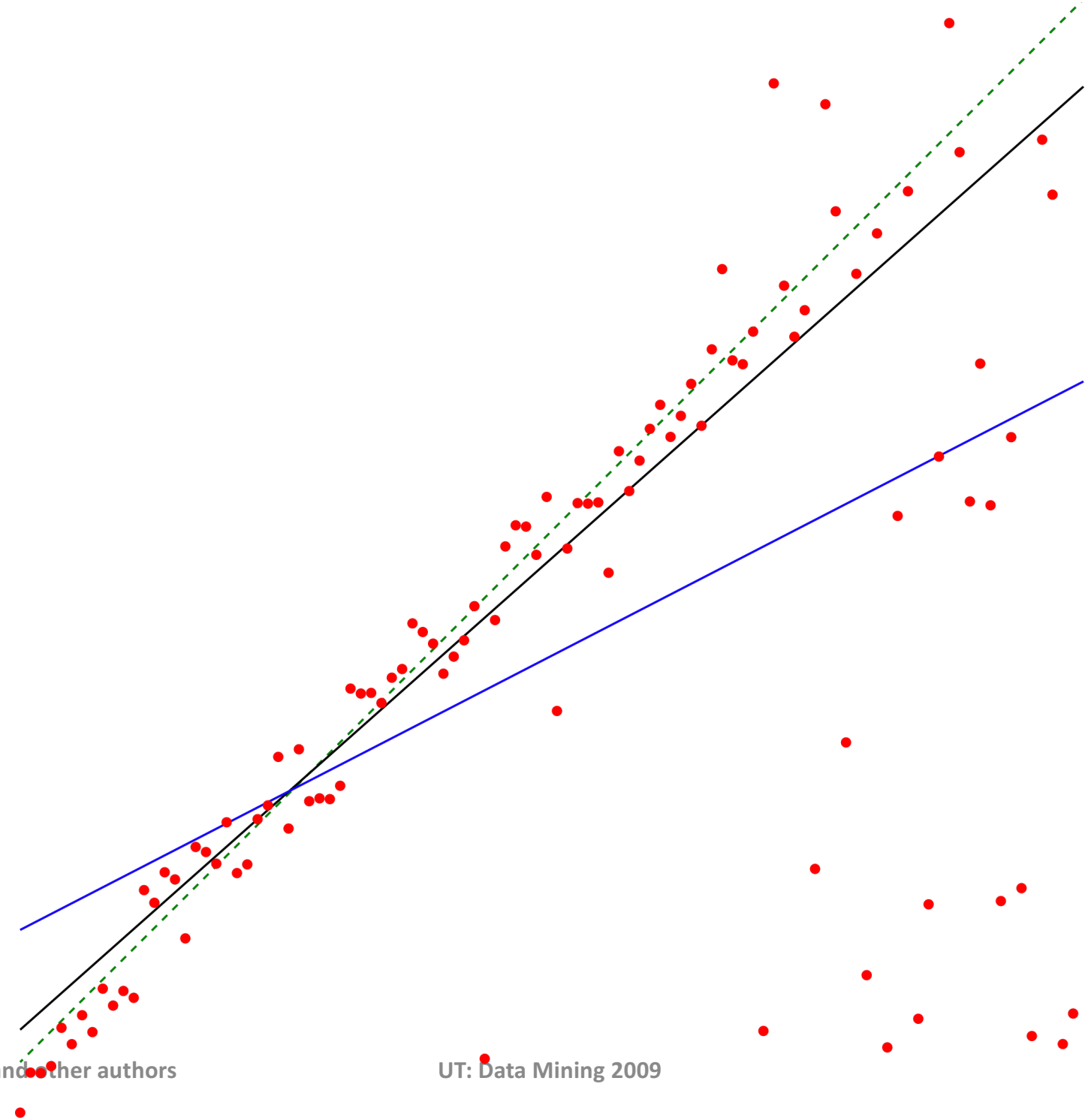


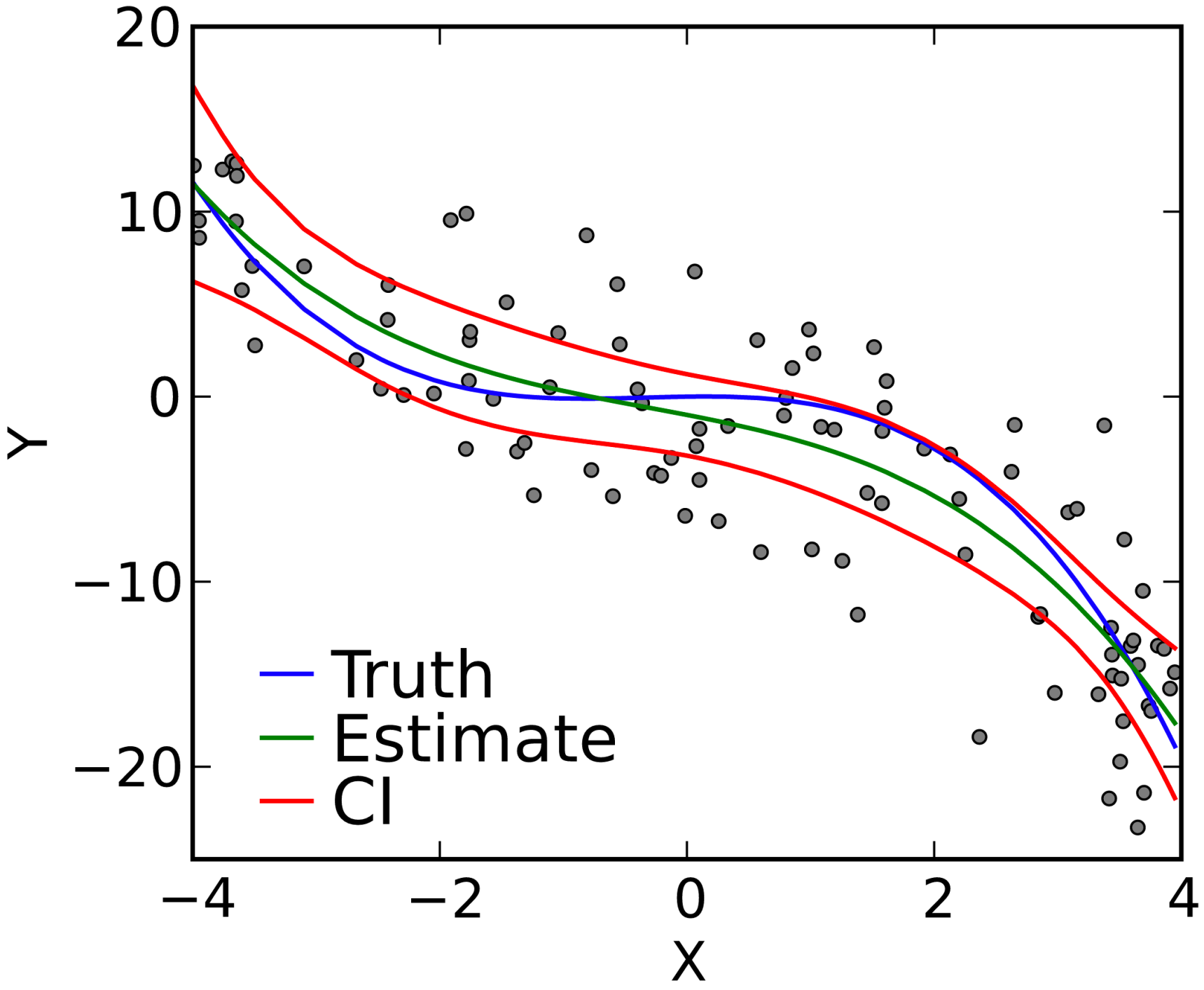
$\alpha_1 = g_1(x,y)$

$\alpha_2 = g_2(x,y)$

- **Response surface design**






20
10
0
−10
−20
Y
−4
−2
0
2
4
X
Truth
Estimate
CI

Multivariate linear regression

all the same, but instead of one feature, x is a k-dimensional vector

$$\mathbf{x}_i = (x_{i1}, x_{i2}, .., x_{ik})$$

the model is the linear combination of all features:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

via the matrix representation: $\hat{\mathbf{y}} = \mathbf{X}\beta$

$$\begin{pmatrix} \hat{y_1} \\ \vdots \\ \hat{y_n} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ \vdots & \vdots & \vdots & \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix}$$

Objectives of regression

- Fit the data with a **“function”**
  - Be able to interpret the function, parameters
- **”Simpler” models are better;** regularisation
- Optimisation of “best fit” – heuristics
- Bias, variance, logistic regression, …

Supervised Learning: Uses

- Prediction of future cases: Use the rule to predict the output for future inputs
- Knowledge extraction: The rule is easy to understand
- Compression: The rule is simpler than the data it explains
- Outlier detection: Exceptions that are not covered by the rule, e.g., fraud

Supervised learning

Observation	Outcome
Summer of 2003 was **cold**	Winter of 2003 was **warm**
Summer of 2004 was **cold**	Winter of 2004 was **cold**
Summer of 2005 was **cold**	Winter of 2005 was **cold**
Summer of 2006 was **hot**	Winter of 2006 was **warm**
Summer of 2007 was **cold**	Winter of 2007 was **cold**
Summer of 2008 was **warm**	Winter of 2008 was **warm**
Summer of 2009 was **cold**	Winter of 2009 was **cold**
Summer of 2010 was **hot**	Winter of 2010 was **cold**
Summer of 2011 was **warm**	**Winter of 2011 will be ?**

Supervised learning

Observation		Outcome
Study=hard,	Professor=☹	I get a C
Study=slack,	Professor=☺	I get an A
Study=hard,	Professor=☹	I get an A
Study=slack,	Professor=☹	I get a D
Study=slack,	Professor=☺	I get an A
Study=slack,	Professor=☺	I get an A
Study=hard,	Professor=☺	I get an A
Study=slack,	Professor=☺	I get a B
	?	**I get an A**

Supervised learning

Day	Observation	Outcome
Mon	I was **not** using magnetic bracelet™	In the evening I had a headache
Tue	I was using magnetic bracelet™	In the evening I had less headache
Wed	I was using magnetic bracelet™	In the evening no headache!
Thu	I was using magnetic bracelet™	The headache is gone!!
Fri	I was **not** using magnetic bracelet™	No headache!!

Supervised learning

Day	Observation	Outcome
Mon	I was **not** using magnetic bracelet™	In the evening I had a headache
Tue	I was using magnetic bracelet™	In the evening I had less headache
Wed	I was using magnetic bracelet™	In the evening no headache!
Thu	I was using magnetic bracelet™	The headache is gone!!
Fri	I was **not** using magnetic bracelet™	No headache!!

Magnetic bracelet™ cures headache

Science

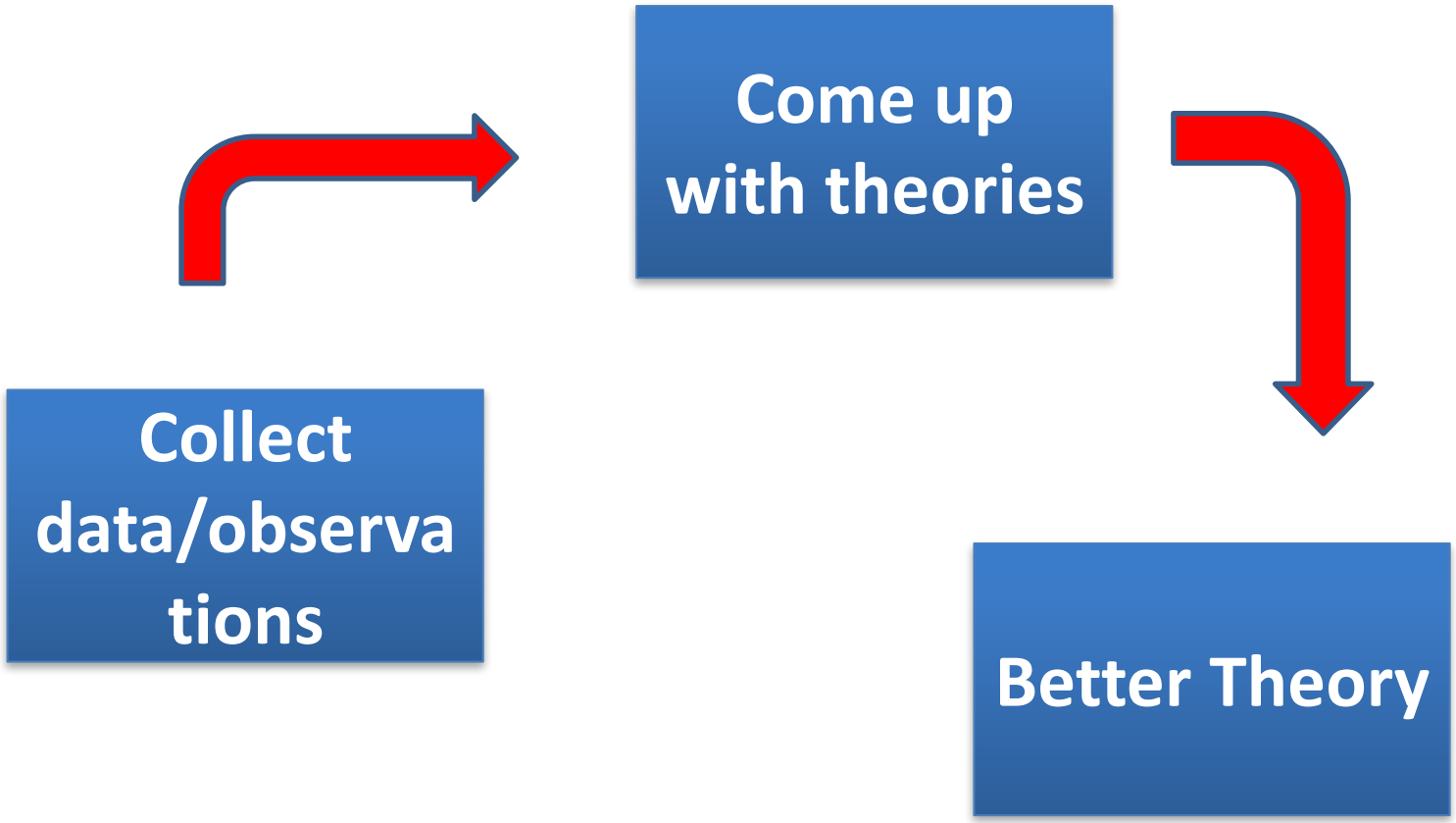
Collect data/observations

Supervised learning

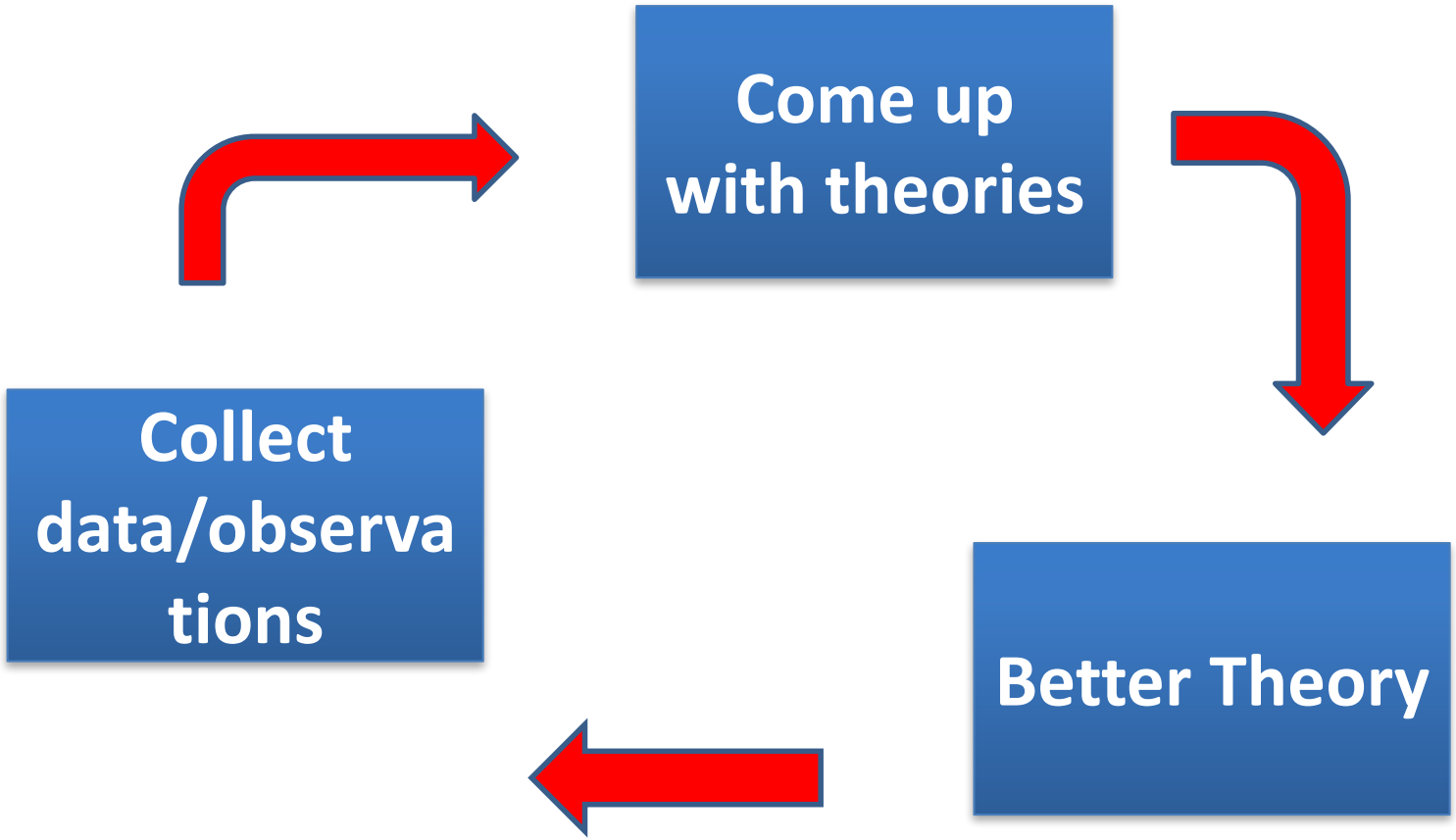
Come up with theories

Collect data/observations

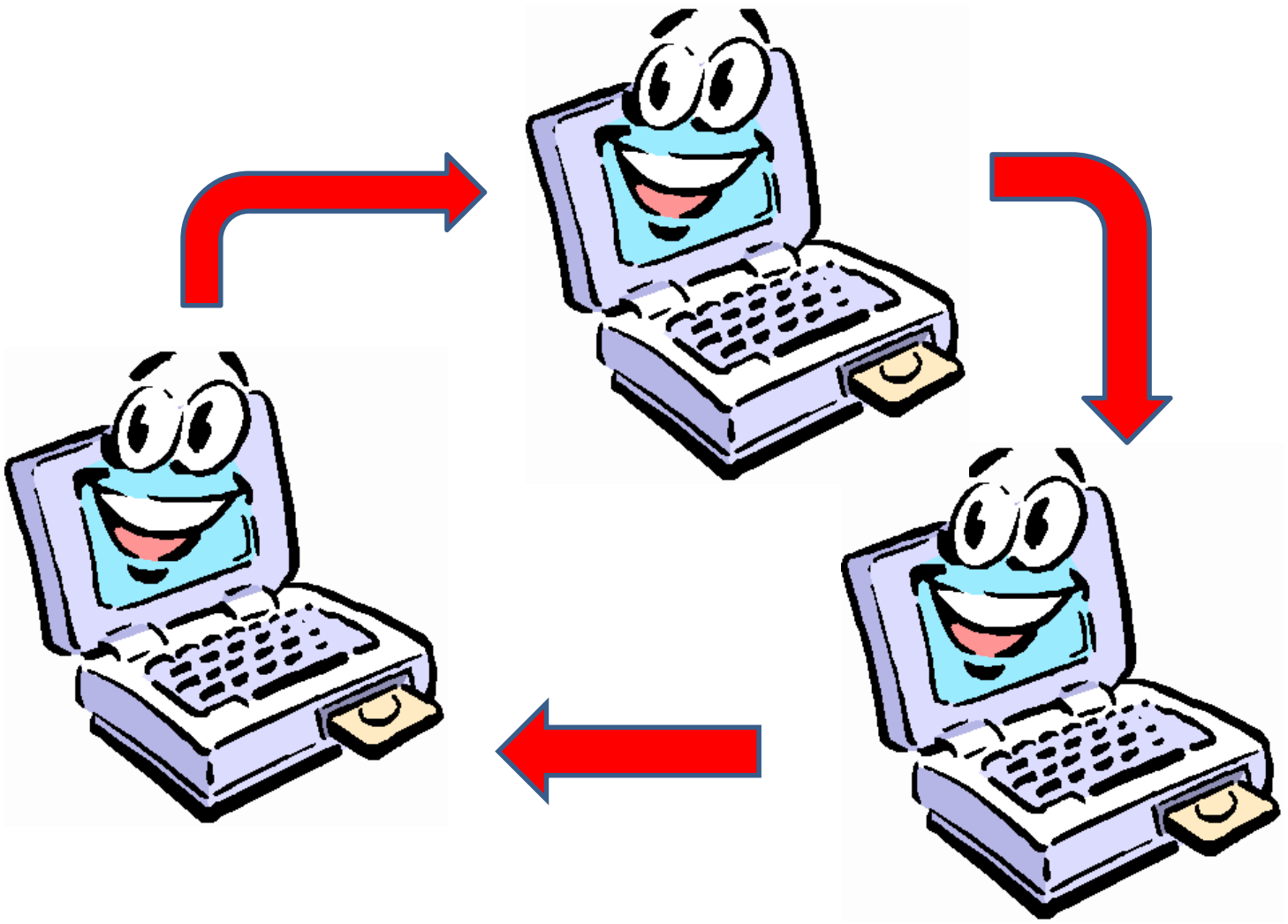
Supervised learning



Supervised learning



Science ☺



Supervised learning

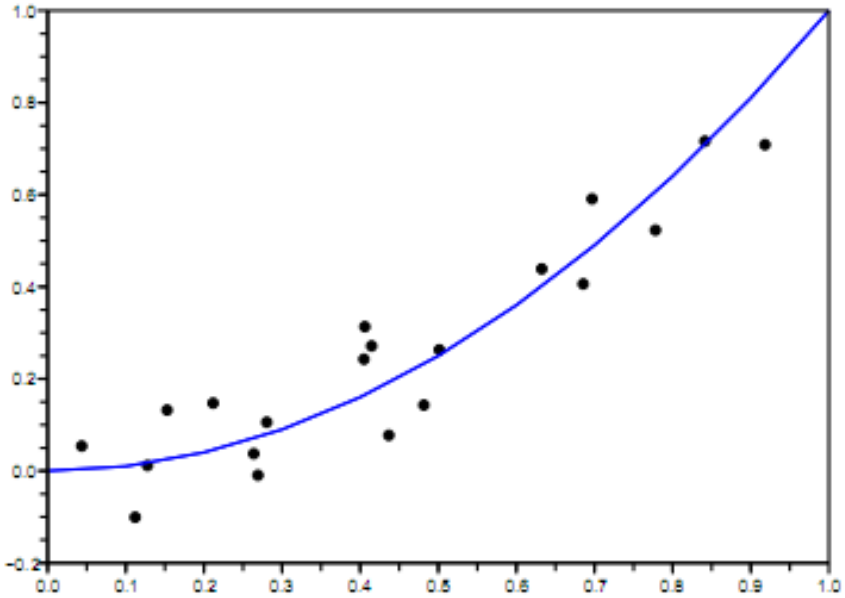
- Formally,

Let $\mathcal{X}$ and $\mathcal{Y}$ be some sets and let there be a dataset of *training samples*:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \mid x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$$

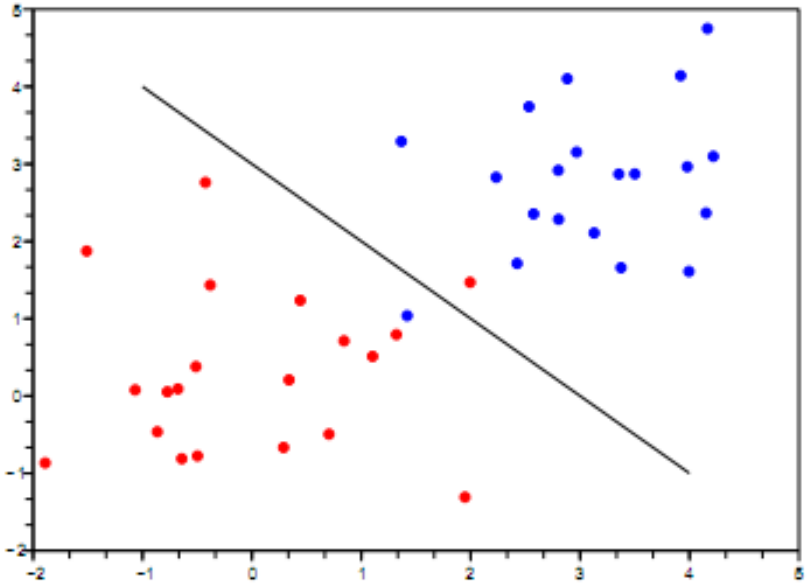
Find a function $f_D : \mathcal{X} \to \mathcal{Y}$ *generalizing the functional relationship present in the data.*

Regression



- $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$.
- $D = \{(0.50, 0.26), (0.43, 0.08), (0.26, 0.00), \ldots\}$
- $f_D(x) = x^2$.

Classification

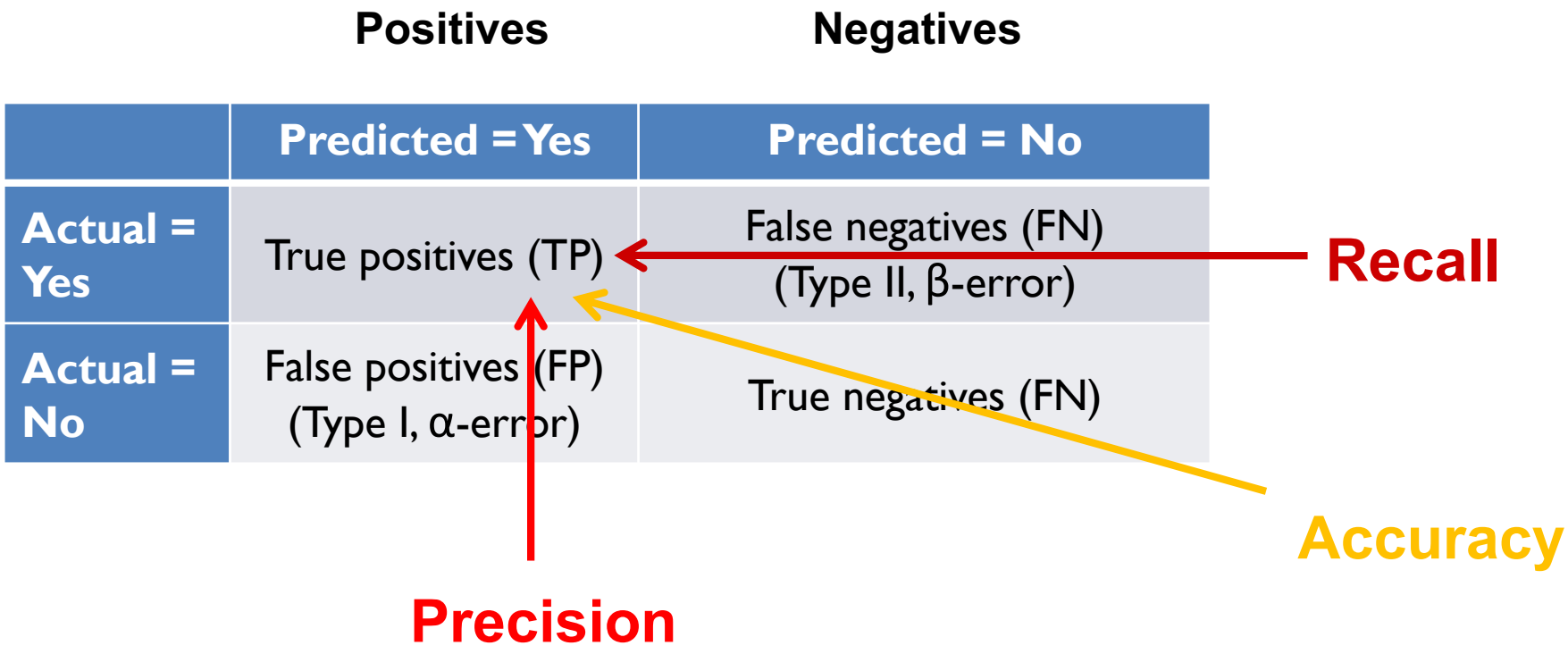


- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{\text{blue}, \text{red}\}$.
- $D = \{((1.3, 0.8), \text{red}), ((2.5, 2.3), \text{blue}), \ldots\}$
- $f_D(x_1, x_2) = \text{if } x_1 + x_2 > 3 \text{ then blue else red.}$

Re-visit the 2x2 tables

	Positives	Negatives
	Predicted = Yes	**Predicted = No**
Actual = Yes	True positives (TP)	False negatives (FN) (Type II, β-error)
Actual = No	False positives (FP) (Type I, α-error)	True negatives (FN)

Classification summary



F-measure = harmonic_mean(Precision, Recall)

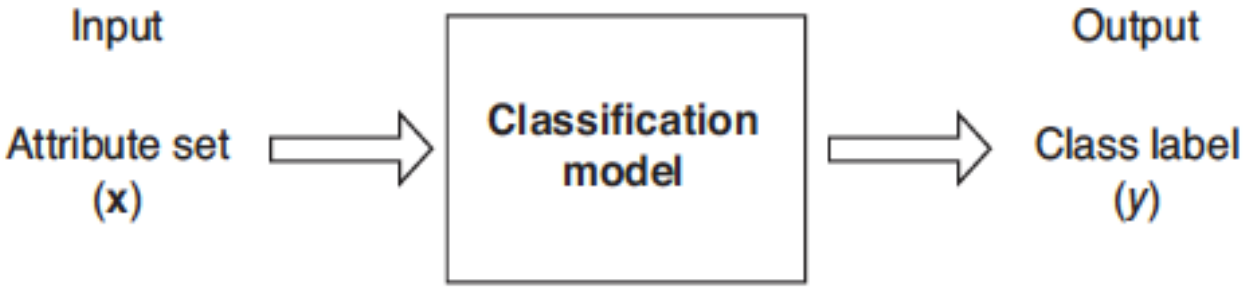
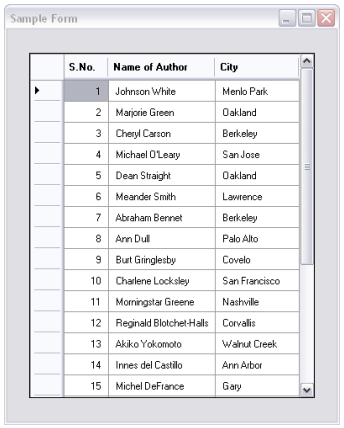
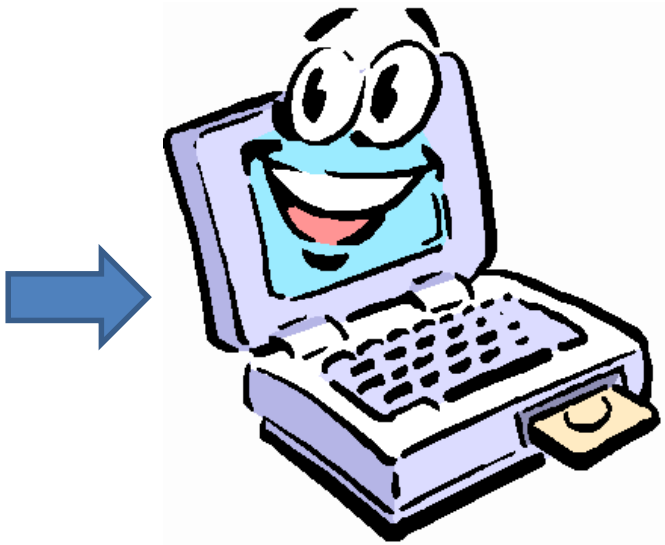


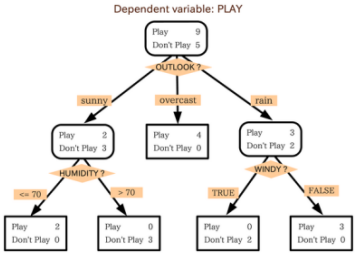
Figure 4.2. Classification as the task of mapping an input attribute set $\mathbf{x}$ into its class label y.

The “Dumb User” Perspective

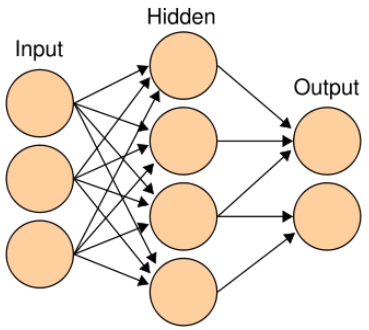
Weka, RapidMiner,
MS SSAS, Clementine,
SPSS, R, scikit-learn,…



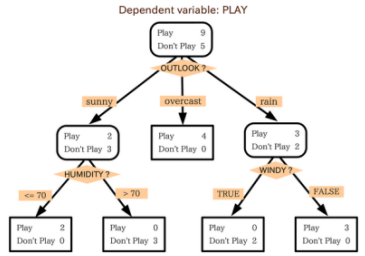




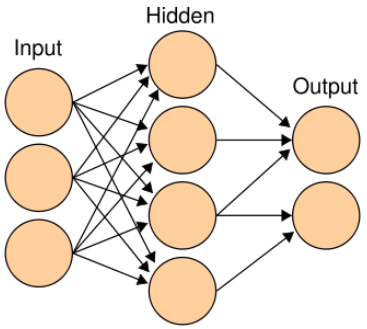
$$f(x) = \sum_i w_i x_i$$



The “Dumb User” Perspective



$$f(x) = \sum_i w_i x_i$$



Validation

Classification demo: Iris dataset

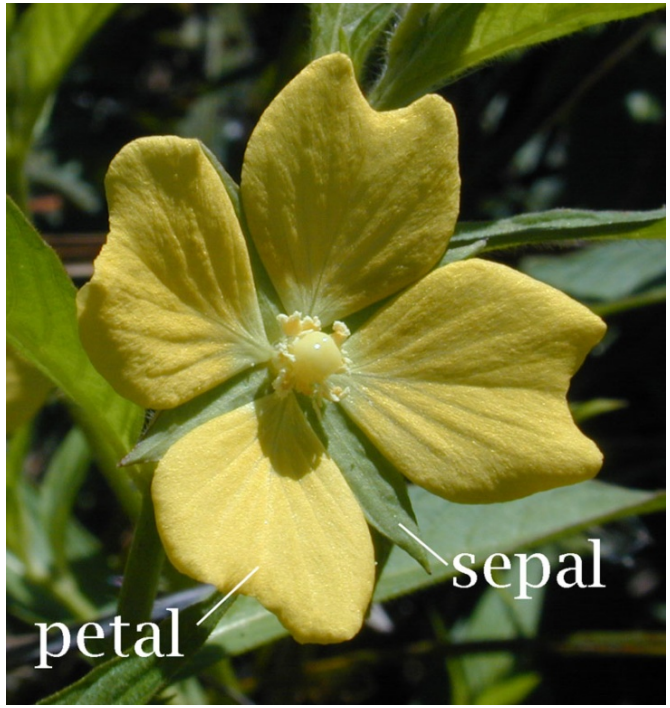
- 150 measurements, 4 attributes, 3 classes

Viewer

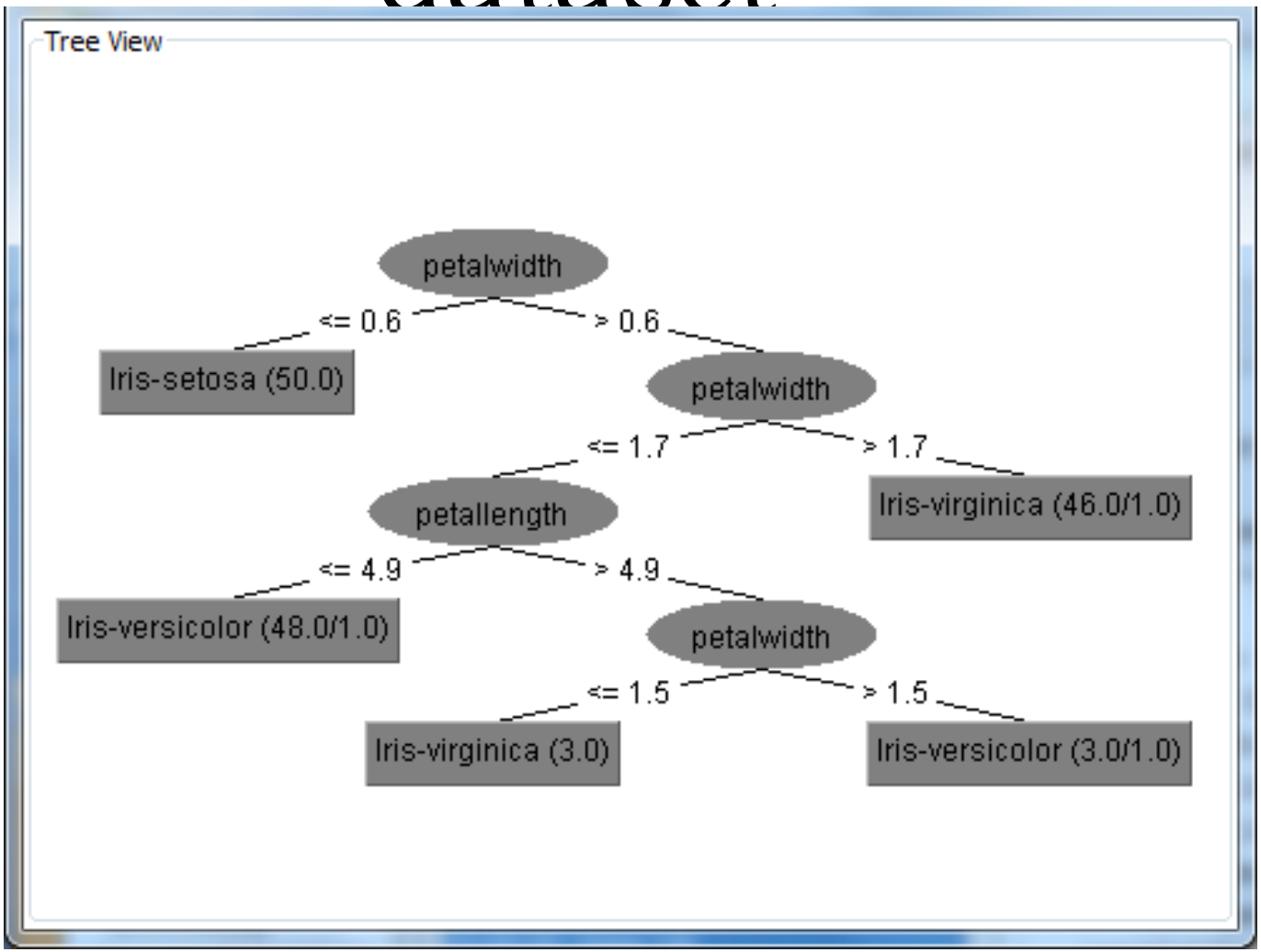
Relation: iris

No.	sepallength Numeric	sepalwidth Numeric	petallength Numeric	petalwidth Numeric	class Nominal
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa
14	4.3	3.0	1.1	0.1	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1.0	0.2	Iris-setosa
24	5.1	3.3	1.7	0.5	Iris-setosa

Undo OK Cancel



Classification demo: Iris dataset



Validation

```
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 49  1 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Correctly Classified Instances	**147**	**98%**
Incorrectly Classified Instances	**3**	**2%**
Kappa statistic	**0.97**	
Mean absolute error	0.0233	
Root mean squared error	0.108	
Relative absolute error	5.2482 %	
Root relative squared error	22.9089 %	
Total Number of Instances	**150**	

Validation

```
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 49  1 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Class	setosa	versic.	virg.	Avg
TP Rate	1	0.98	0.96	0.98
FP Rate	0	0.02	0.01	0.01
Precision	1	0.961	0.98	0.98
Recall	1	0.98	0.96	0.98
F-Measure	1	0.97	0.97	0.98
ROC Area	1	0.99	0.99	0.99

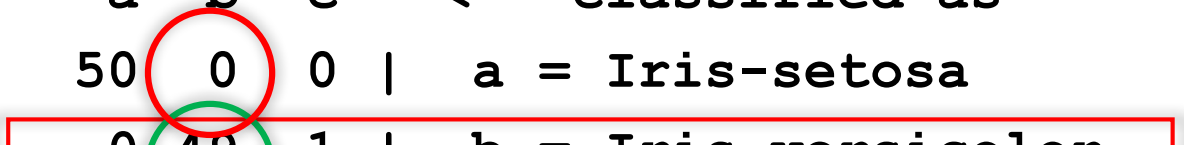
Validation

```
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 49  1 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

	versic.
TP Rate	0.98
FP Rate	0.02
Precision	0.961
Recall	0.98
F-Measure	0.97
ROC Area	0.99

Validation

```
  a   b   c    <-- classified as
 50   0   0 |   a = Iris-setosa
  0  49   1 |   b = Iris-versicolor
  0   2  48 |   c = Iris-virginica
```



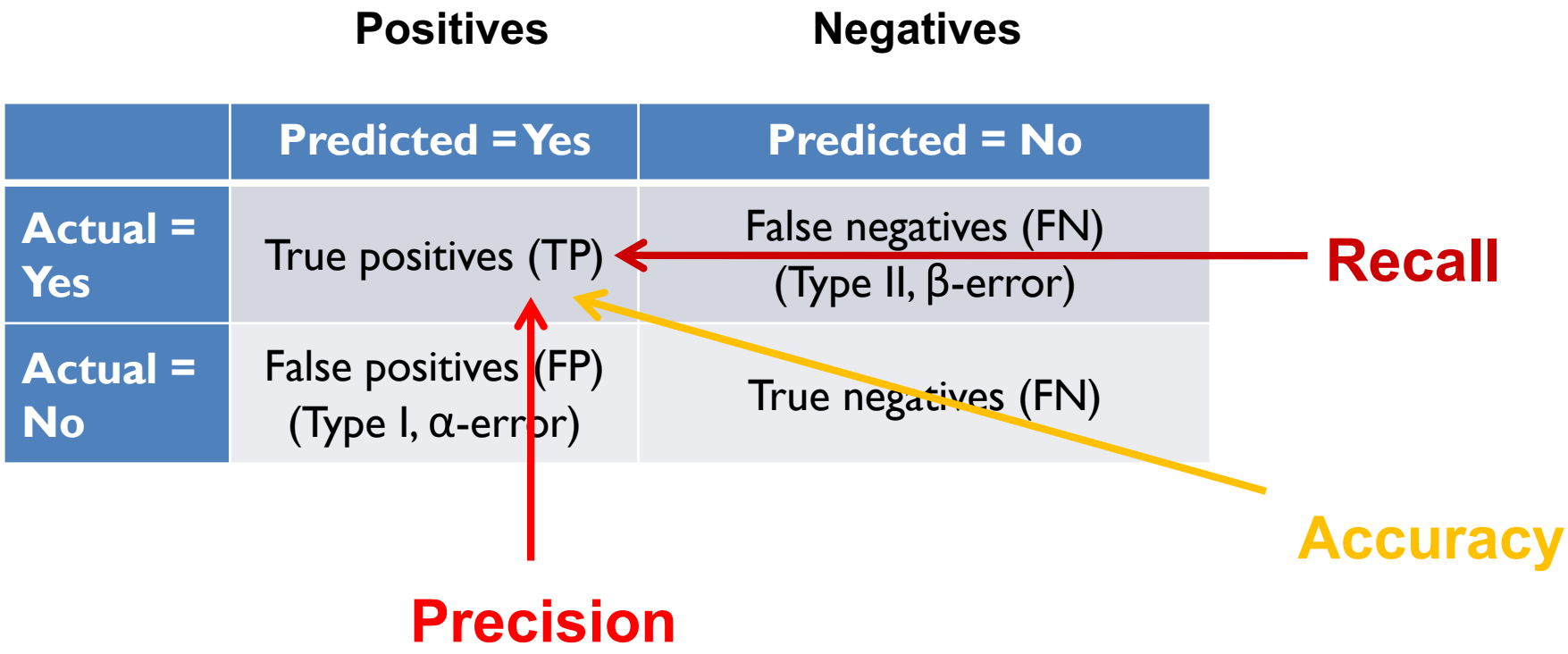
"True positives" "False positives"

	versic.	
TP Rate	0.98	= TP/positive examples
FP Rate	0.02	= FP/negative examples
Precision	0.961	= TP/positives
Recall	0.98	= TP/positive examples
F-Measure	0.97	= 2*P*R/(P + R)
ROC Area	0.99	~ Pr(s(false)<s(true))

Classification summary

	Positives	Negatives
	Predicted = Yes	**Predicted = No**
Actual = Yes	True positives (TP)	False negatives (FN) (Type II, β-error)
Actual = No	False positives (FP) (Type I, α-error)	True negatives (FN)

Classification summary

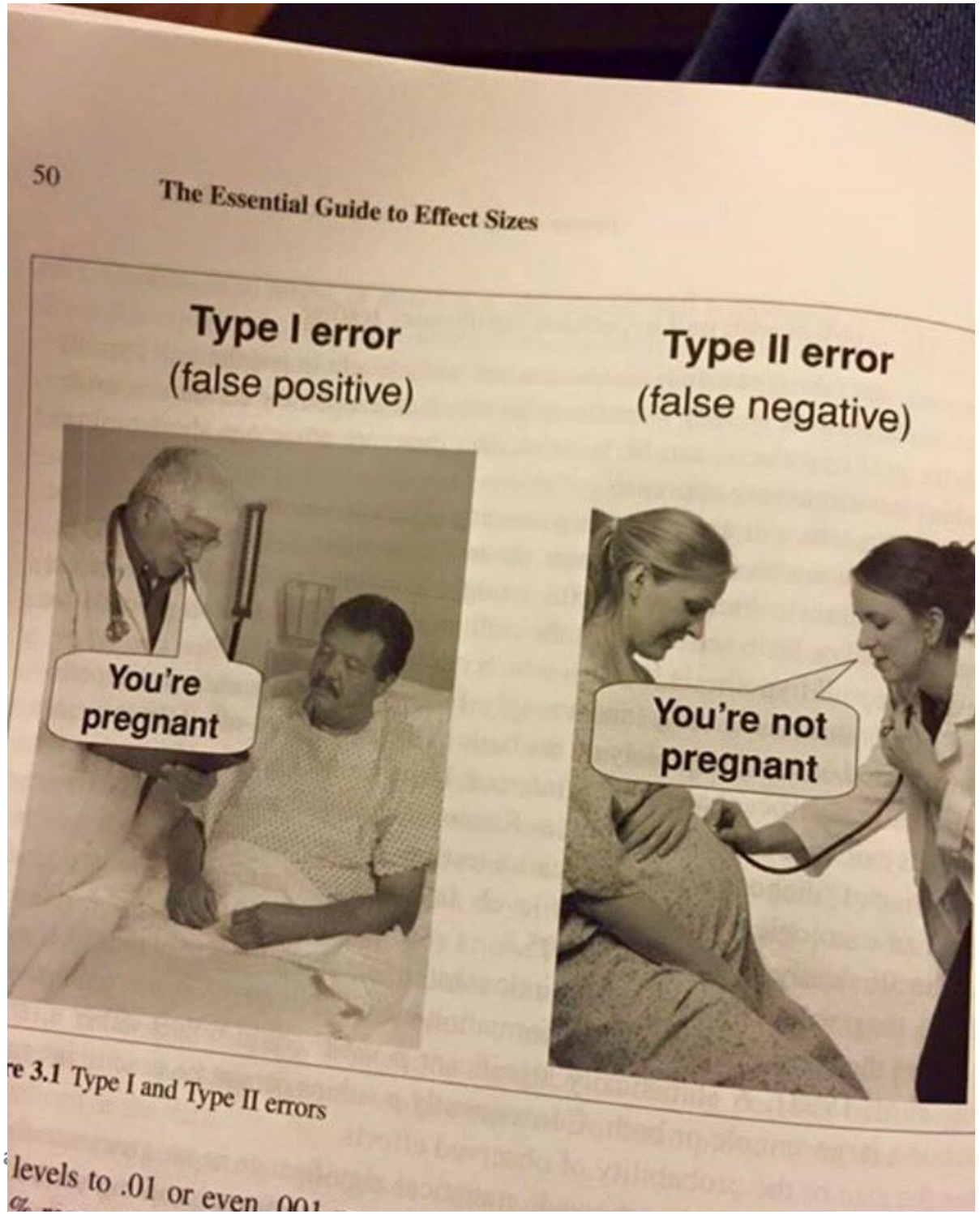


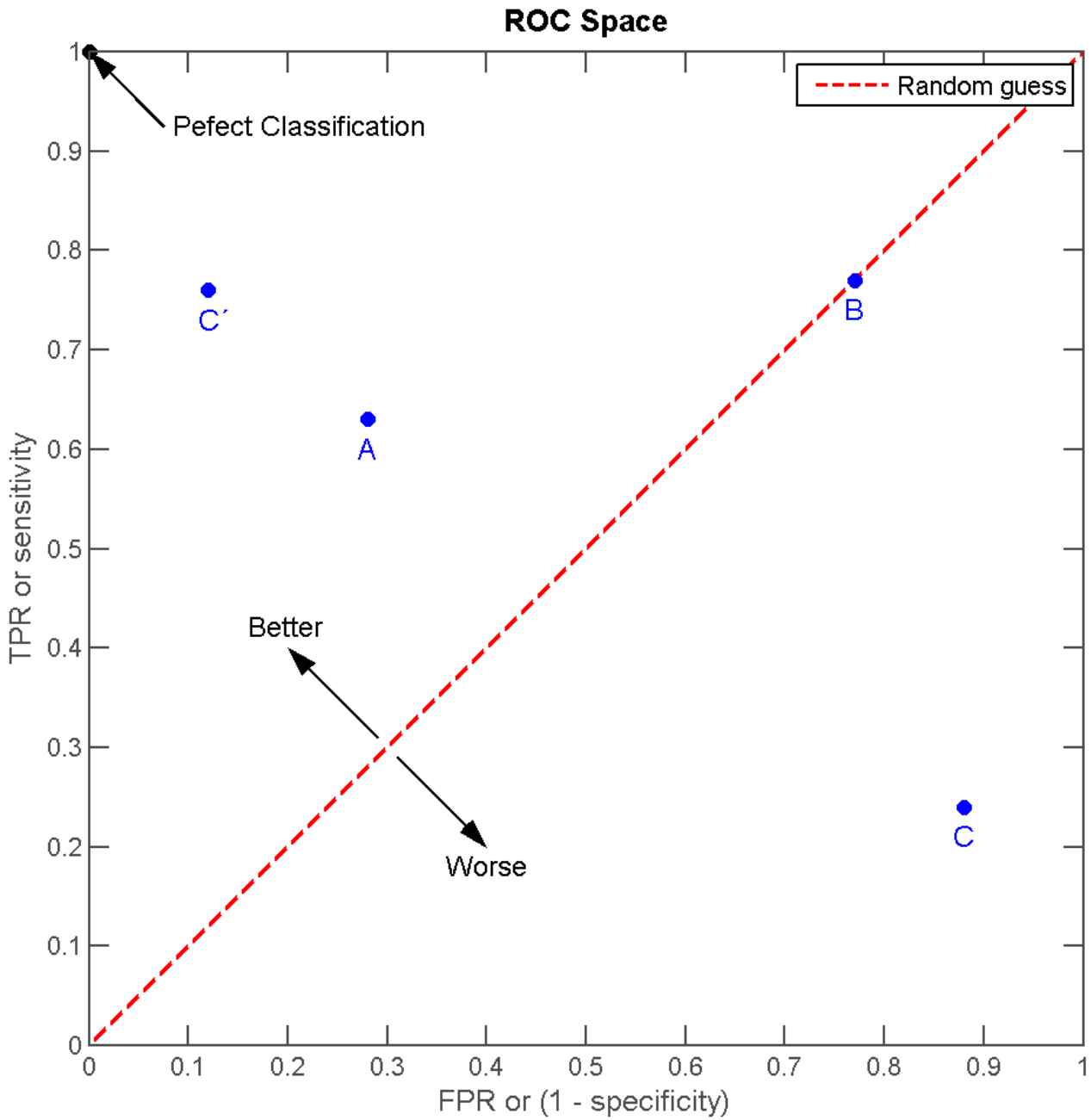
F-measure = harmonic_mean(Precision, Recall)

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

		Predicted condition			
	Total population	Predicted Condition positive	Predicted Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
True condition	condition positive	**True positive**	**False Negative** (Type II error)	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$
	condition negative	**False Positive** (Type I error)	**True negative**	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

A	B	C	C′
TP=63 FP=28 91 FN=37 TN=72 109 100 100 200	TP=77 FP=77 154 FN=23 TN=23 46 100 100 200	TP=24 FP=88 112 FN=76 TN=12 88 100 100 200	TP=76 FP=12 88 FN=24 TN=88 112 100 100 200
TPR = 0.63	TPR = 0.77	TPR = 0.24	TPR = 0.76
FPR = 0.28	FPR = 0.77	FPR = 0.88	FPR = 0.12
PPV = 0.69	PPV = 0.50	PPV = 0.21	PPV = 0.86
F1 = 0.66	F1 = 0.61	F1 = 0.22	F1 = 0.81
ACC = 0.68	ACC = 0.50	ACC = 0.18	ACC = 0.82

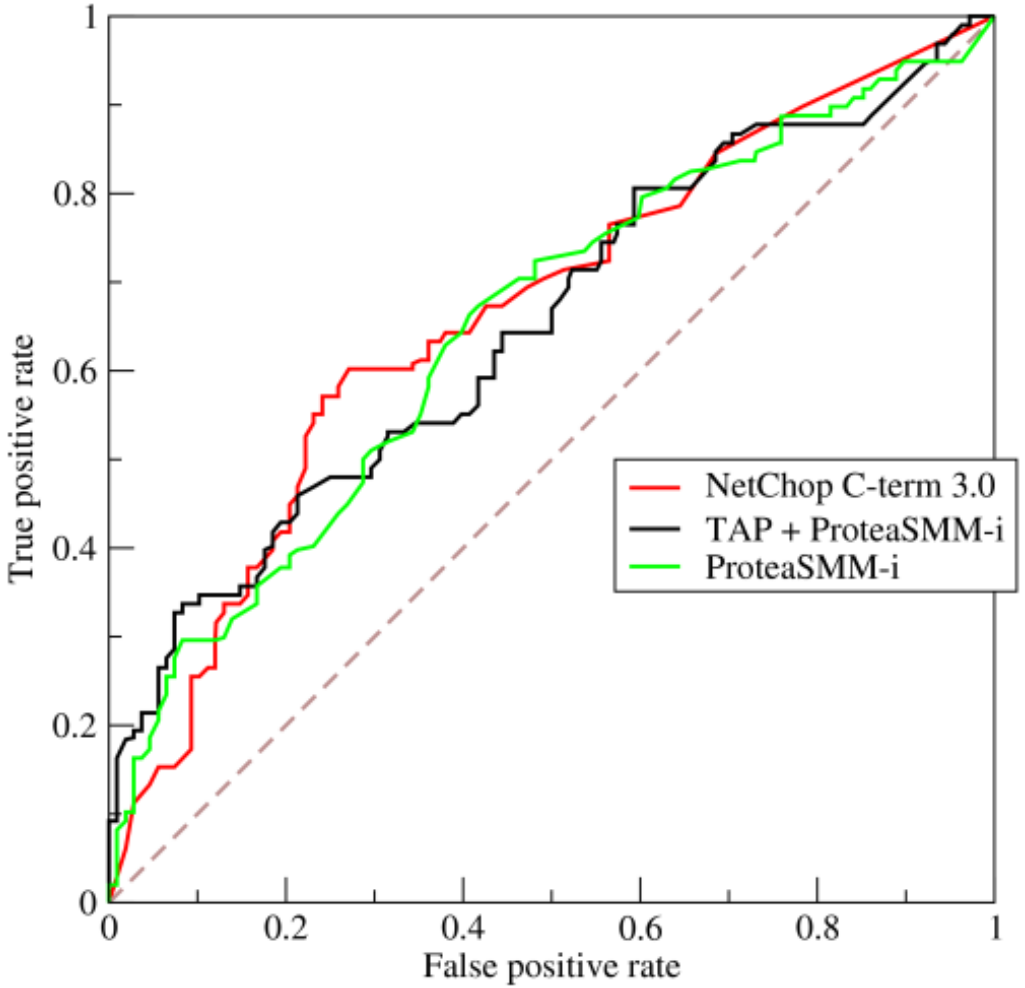

50
The Essential Guide to Effect Sizes
Type I error
(false positive)
Type II error
(false negative)
You're pregnant
You're not pregnant
re 3.1 Type I and Type II errors
levels to .01 or even .001


ROC Space
Random guess
Pefect Classification
C´
B
A
C
Better
Worse
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

ROC

- Your predictions sometimes have natural “order” – from your “best bet” to gradually ”weaker” predictions
- How many you call depends on you
- At each “point” your recall may increase or be “false positive”.
- Model is “better” if it outperforms others across a wide range of “cutoffs”.

ROC curve – Receiver Operator Characteristic, AUC – Area Under Curve



“Training” classifiers on data

- Thus, a “good” classifier is the one which has good Accuracy/Precision/Recall.
  - Or better ROC curve, higher ROC AUC score

- Hence, machine learning boils down to **finding a function** that **optimizes these parameters** for **given data**.

“Training” classifiers on data

- Thus, a “good” classifier is the one which has good Accuracy/Precision/Recall.

- Hence, machine learning boils down to **finding a function** that **optimizes these parameters** for **given data**.

- Yet, there’s a catch

"Training" classifiers on data

- **We want our algorithm to perform well on "unseen" data!**
  - **This makes algorithms and theory way more complicated.**
  - **This makes validation somewhat more complicated.**

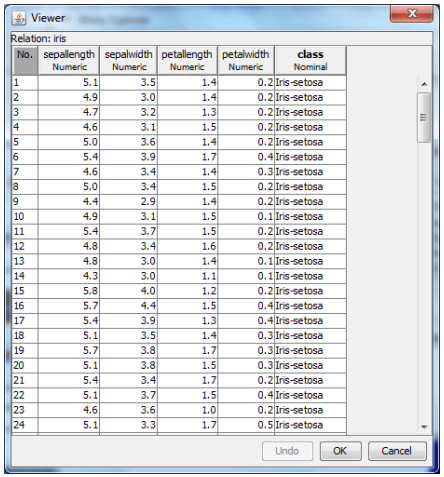
Proper validation

- You **may not test your algorithm** on the same data that you used to train it!

Proper validation

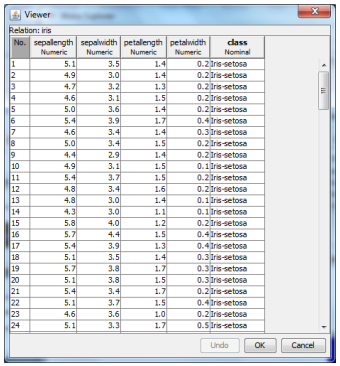
- You **may not test your algorithm** on the same data that you used to train it!

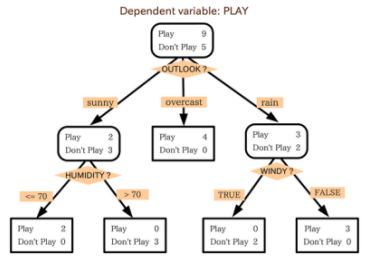
Proper validation :: Holdout



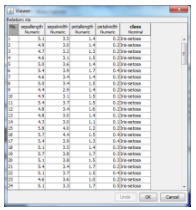
Split

Training set





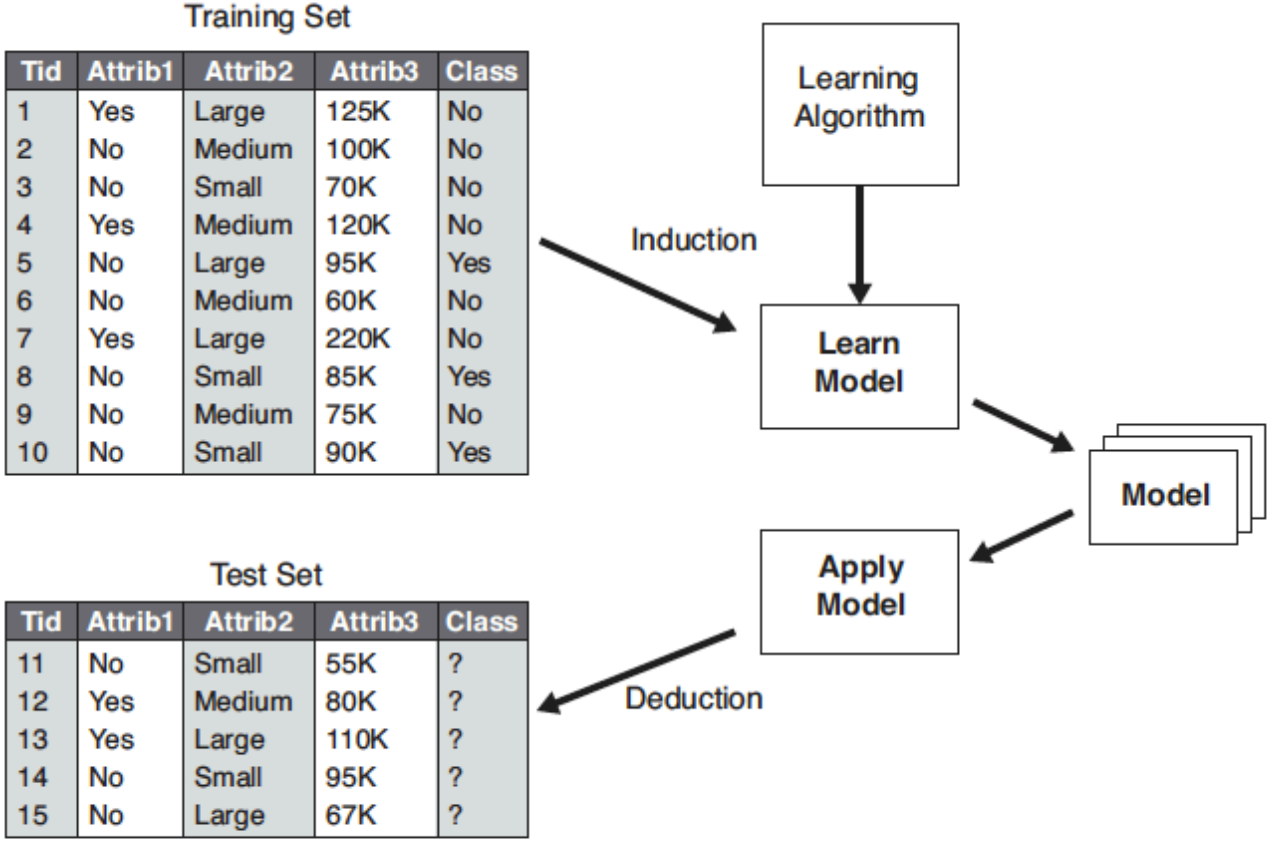
Testing set



Validation

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes



Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Figure 4.3. General approach for building a classification model.

Proper validation

- What are the "sufficient" sizes for the test/training sets and why?
- What if the data is scarce?
  - Cross-validation
  - **K-fold cross-validation**
  - **Leave-one-out cross-validation**
  - Bootstrap .632+

The Netflix Prize Data

- Netflix released three datasets
  - 480,189 *users* (anonymous)
  - 17,770 *movies*
  - *ratings* on integer scale 1 to 5

- *Training set*: 99,072,112 < *user, movie* > pairs with *ratings*
- *Probe set*: 1,408,395 < *user, movie* > pairs with *ratings*
- *Qualifying set* of 2,817,131 < *user*, *movie* > pairs with *no ratings*

Jeff Howbert

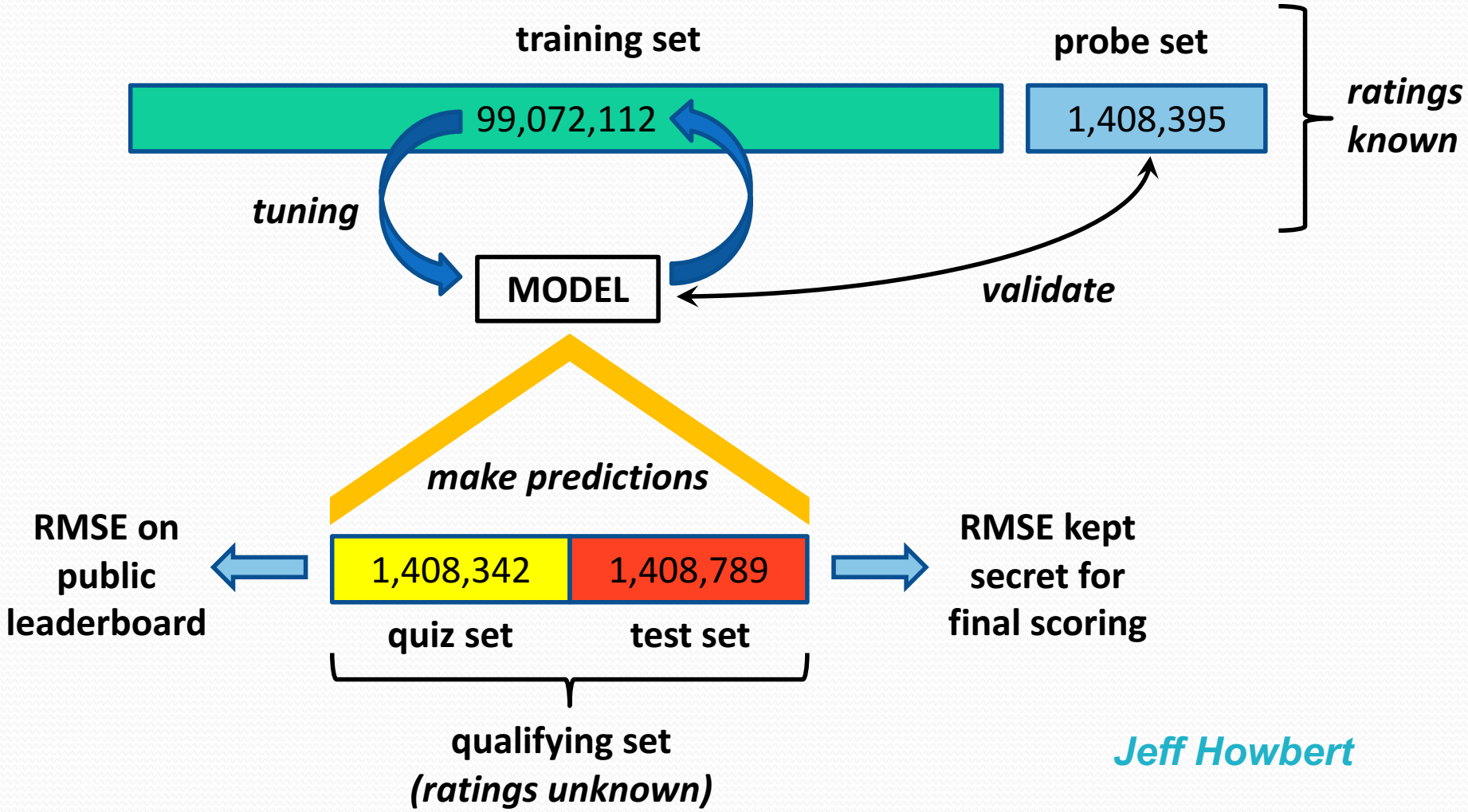
Why the Netflix Prize Was Hard

- Massive dataset
- Very sparse – matrix only 1.2% occupied
- Extreme variation in number of ratings per user
- Statistical properties of qualifying and probe sets different from training set

	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6	movie 7	movie 8	movie 9	movie 10	…	movie 17770
user 1			1		2							3
user 2		2		3	3			4				
user 3							5	3		4		
user 4	2				3			2				2
user 5		4				5			3			4
user 6			2									
user 7			2					4	2	3		
user 8	3	4				4						
user 9									3			
user 10			1		2							2
…												
user 480189		4			3			3				

Jeff Howbert

Model Building and Submission Process



Jeff Howbert

Netflix Prize

Home Rules Leaderboard Register Update Submit Download

Leaderboard

Display top 100 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	-	--	--
	Grand Prize - RMSE <= 0.8563			
1	BellKor in BigChaos	0.8604	9.56	2008-12-03 16:46:15
	Progress Prize - RMSE <= 0.8625			
2	BigChaos	0.8626	9.33	2008-12-04 19:18:27
3	BellKor	0.8630	9.29	2008-12-04 19:25:59
4	PragmaticTheory	0.8638	9.21	2008-11-28 11:46:23
5	Gravity	0.8654	9.04	2008-11-27 21:18:37
6	My Brain and His Chain	0.8668	8.89	2008-09-30 02:19:47
7	Just a guy in a garage	0.8672	8.85	2008-12-07 06:51:12
8	When Gravity and Dinosaurs Unite	0.8675	8.82	2008-10-05 14:16:53
9	Opera Solutions	0.8676	8.81	2008-12-02 22:08:45
10	acmehill	0.8677	8.80	2008-12-05 08:01:00
11	scientist	0.8677	8.80	2008-12-02 01:10:13
12	Ces	0.8711	8.44	2008-08-25 05:00:23
13	Dace	0.8711	8.44	2008-12-07 03:46:04
	Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
14	KorBell	0.8712	8.43	2007-10-01 23:25:23
15	basho	0.8714	8.41	2008-05-21 22:06:00
16	pengpengzhou	0.8714	8.41	2008-11-05 01:11:13
17	blednotik	0.8717	8.38	2008-11-26 00:12:12

Jeff Howbert

Intermediate summary

- **Supervised learning** = predicting f(x) **well**.
- For classification, "well" = high accuracy/precision/recall **on unseen data**.
- To achieve that, most training algorithms will try to optimize their accuracy/precision/recall on **training data**.
- We can then **validate** how good they are on **test data**.

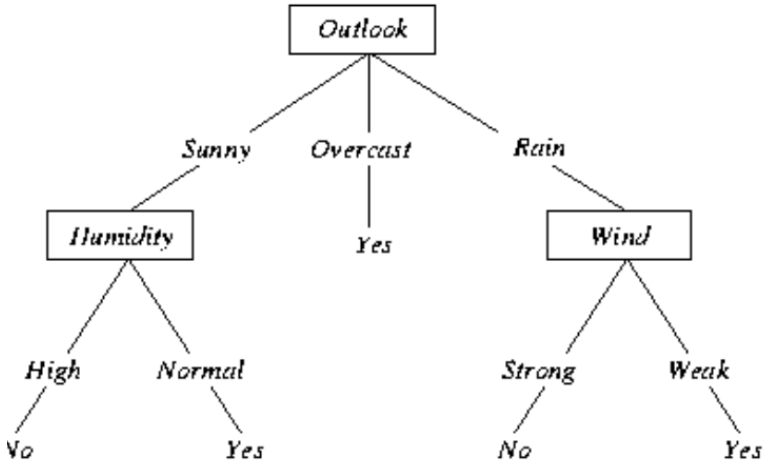
Next

- **Three examples of approaches**
  - **Ad-hoc**
    - **Decision tree induction**
  - **Probabilistic modeling**
    - **Naïve Bayes classifier**
  - **Objective function optimization**
    - **Linear least squares regression**

Decision Tree Induction :: ID3

- **Iterative Dichotomizer 3**
  - Simple yet popular decision tree induction algorithm
  - Builds a decision tree top-down, starting at the root.

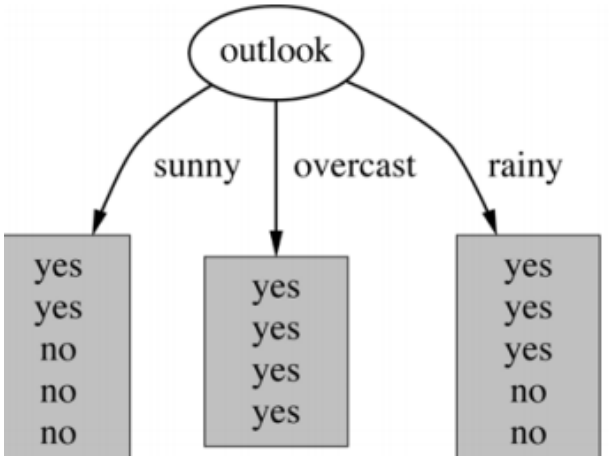
Ross Quinlan

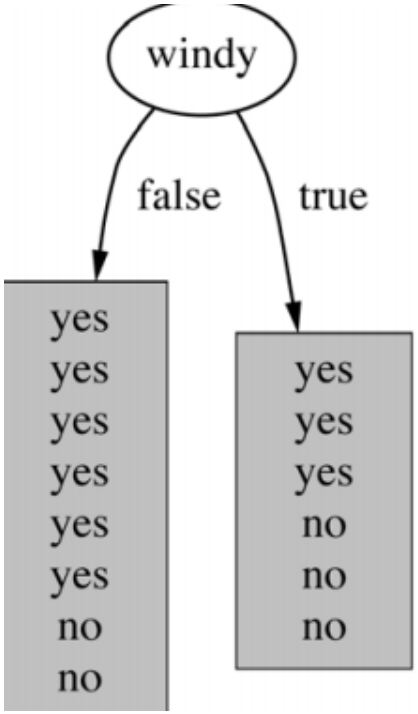


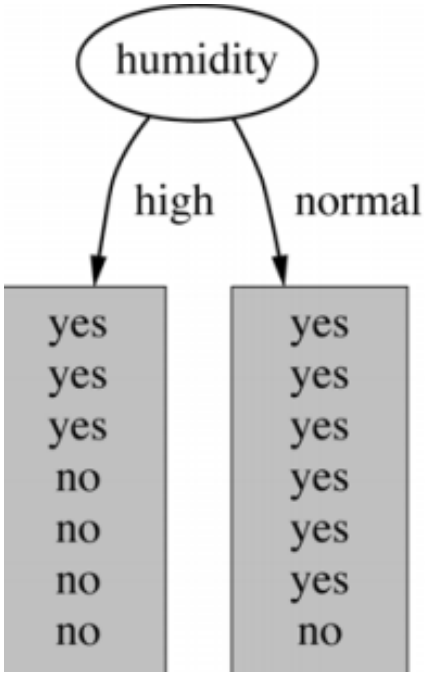
ID3

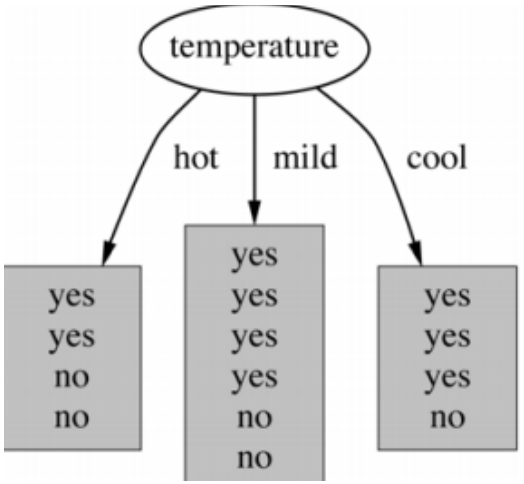
Day	Outlook	Temp	Humidity	Wind	*PlayTennis*
D1	*Sunny*	*Hot*	*High*	*Weak*	*No*
D2	*Sunny*	*Hot*	*High*	*Strong*	*No*
D3	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
D4	*Rain*	*Mild*	*High*	*Weak*	*Yes*
D5	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
D6	*Rain*	*Cool*	*Normal*	*Strong*	*No*
D7	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
D8	*Sunny*	*Mild*	*High*	*Weak*	*No*
D9	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
D10	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
D11	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
D12	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
D13	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
D14	*Rain*	*Mild*	*High*	*Strong*	*No*

ID3 :: First split









Which split is the most informative?

Entropy

- Entropy measures the "informativeness" of a probability distribution.

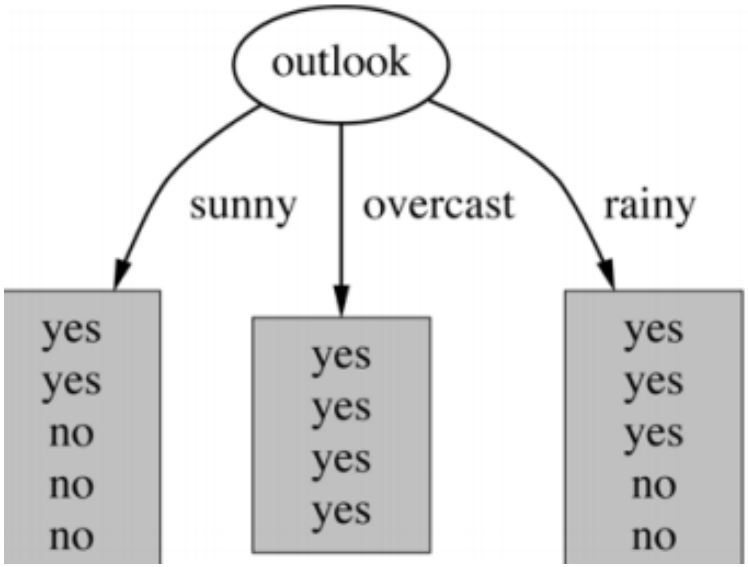
$$H(p) = -\sum_i p_i \log_2 p_i$$

- A split is informative if it *reduces entropy.*

Information gain of a split

- Before split:
  - p_{no} = 5/14, p_{yes} = 9/14, H(p) = **0.94**
- After split on outlook:

H=0.97 **H=0** **H=0.97**



$$0.97\frac{5}{14}+0\frac{4}{14}+0.97\frac{5}{14}$$

=0.69

Information gain
= 0.94-0.69 = 0.25

Entropy, Gini, Misclassification

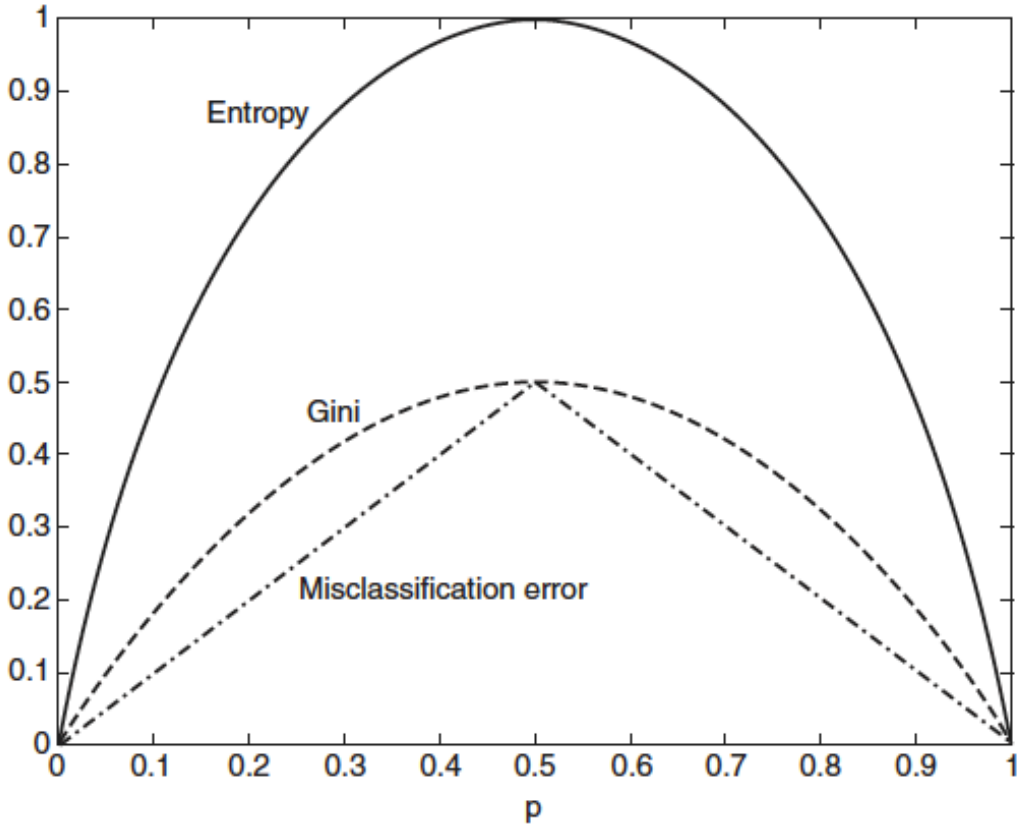


Figure 4.13. Comparison among the impurity measures for binary classification problems.

ID3

1. Start with a single node
2. Find the attribute with the largest information gain
3. Split the node according to this attribute
4. Repeat recursively on subnodes

C4.5

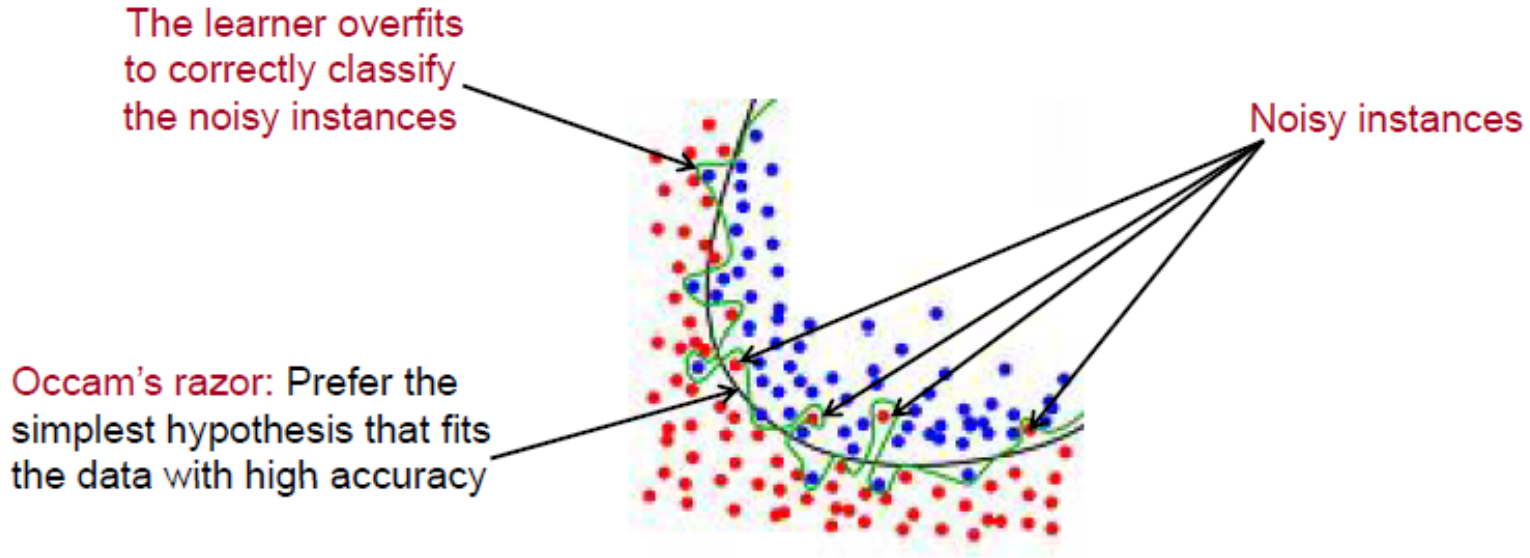
- C4.5 is an extension of ID3
  - Supports continuous attributes
  - Supports missing values
  - Supports pruning
- There is also a C5.0
  - A commercial version with additional bells & whistles

Decision trees - discussion

- The goods:
  - Easy & efficient
  - Interpretable and pretty
- The bads
  - Rather ad-hoc – optimal trees NP-hard to train
  - **Can overfit unless properly pruned**
  - Not the best model for all classification tasks
- Improvements:
  - **Random Forest** – multiple trees, majority voting

Why Does my Method Overfit ?

- **In domains with noise or uncertainty** the system may try to decrease the training error by completely fitting all the training examples



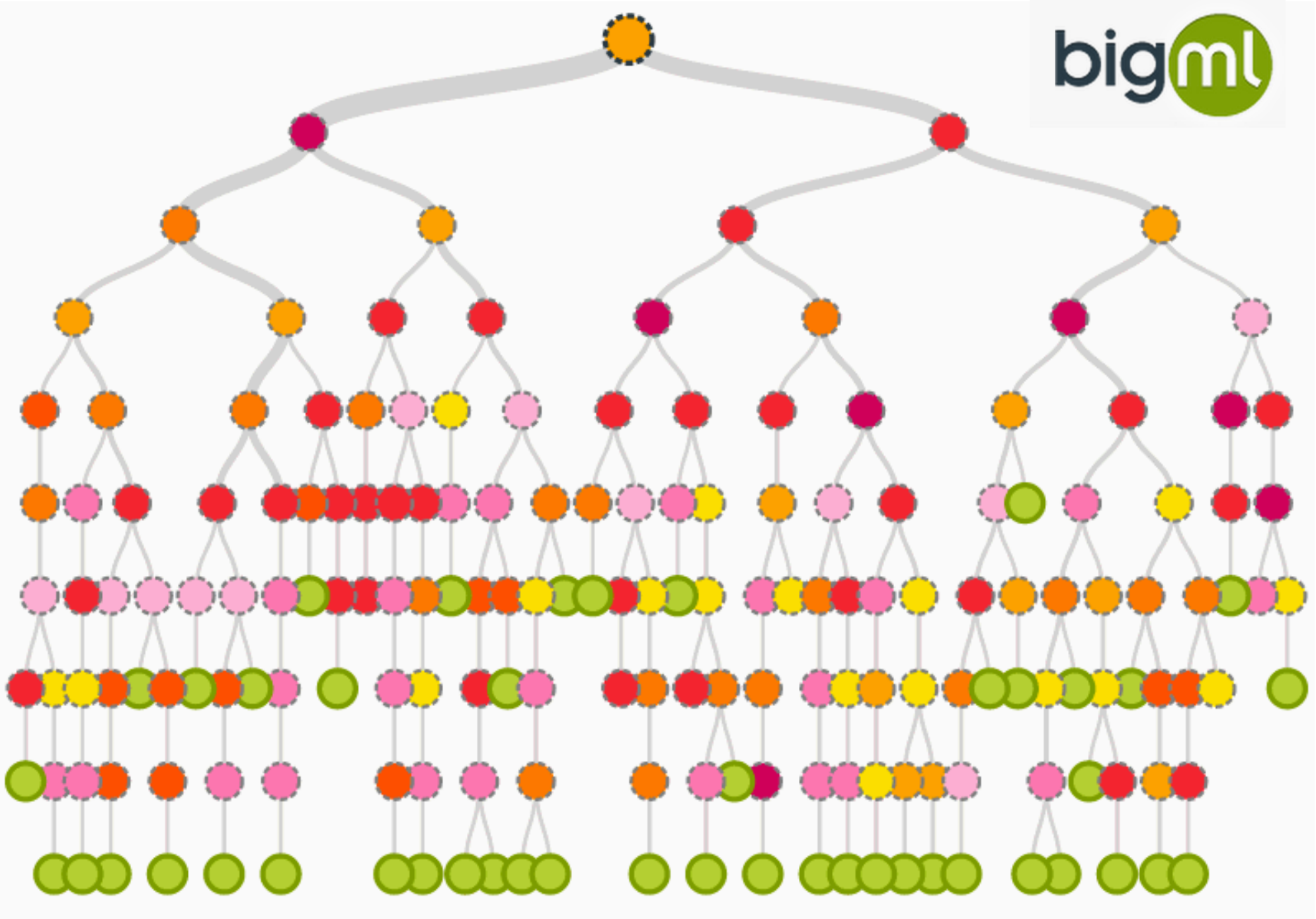
Slide from: Pavan J Joshi

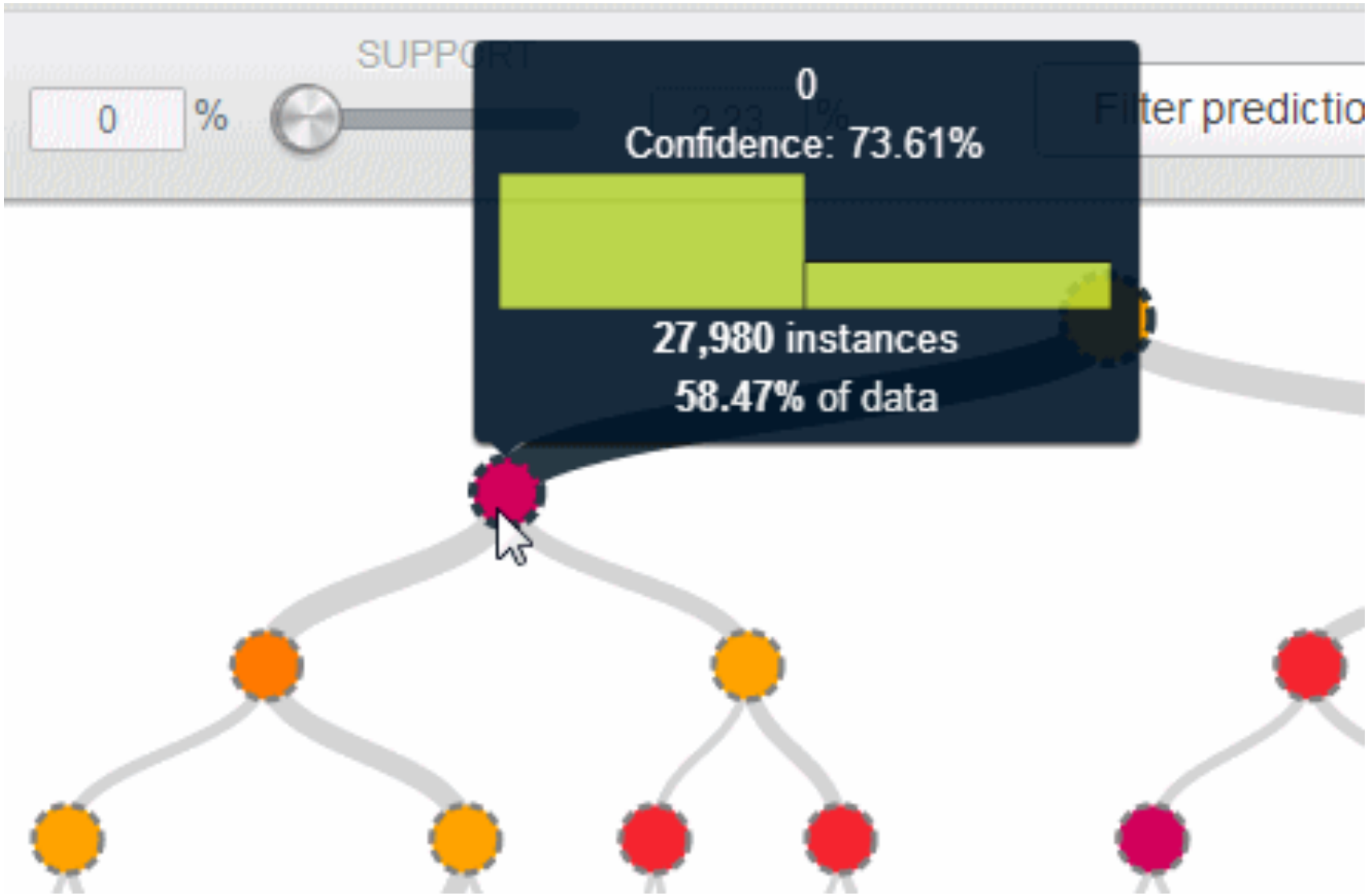
Overfitting

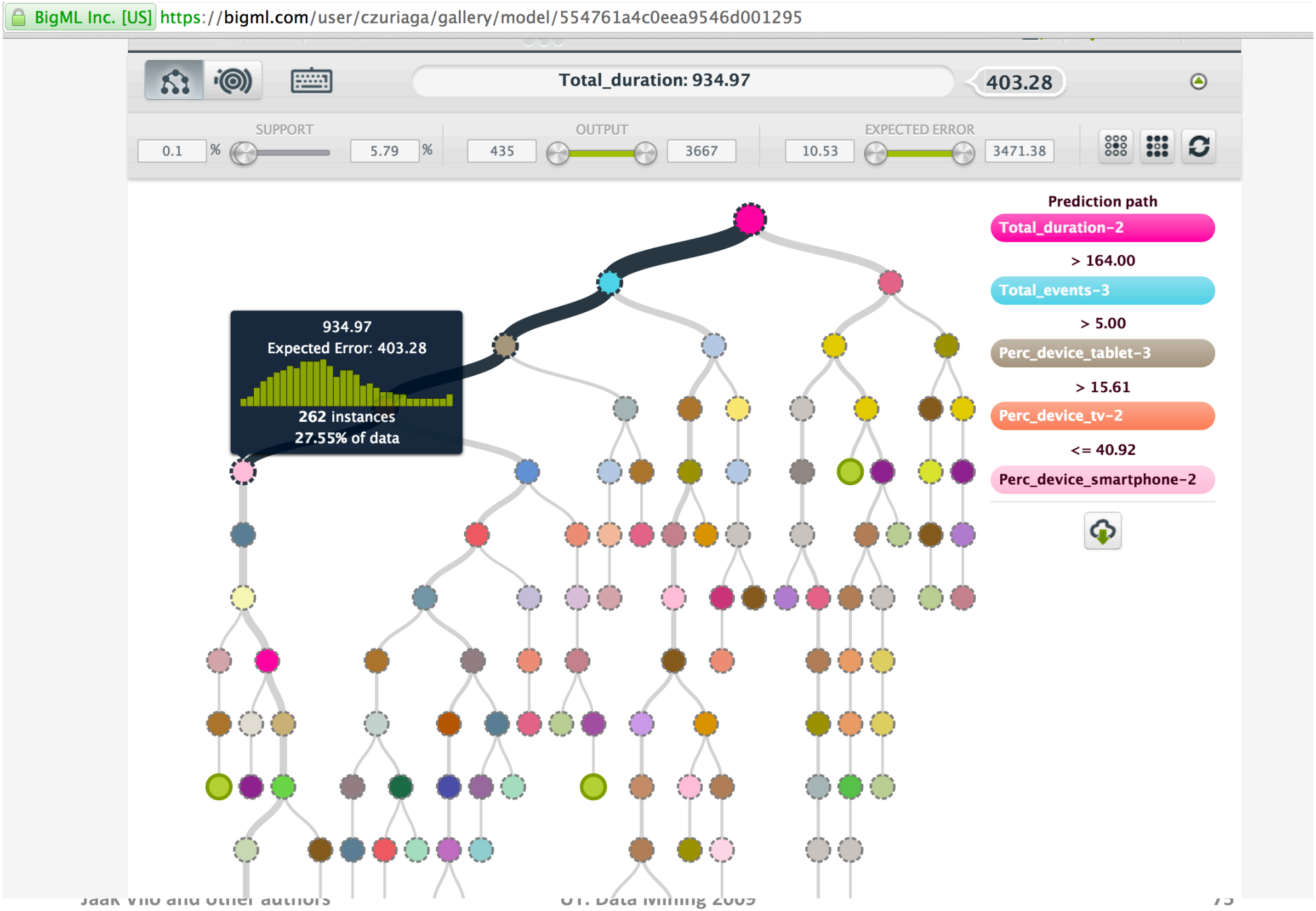
- The longer you train, the worse it becomes…
- Always test against data that was not used in training

Pruning

- Delete subtrees if they do not improve predictions
- Top/down
- Bottom/up
- …
- MDL – Minimum Description Length Principle
  - MDL - Minimise length of **"Model + Exceptions"**


bigml

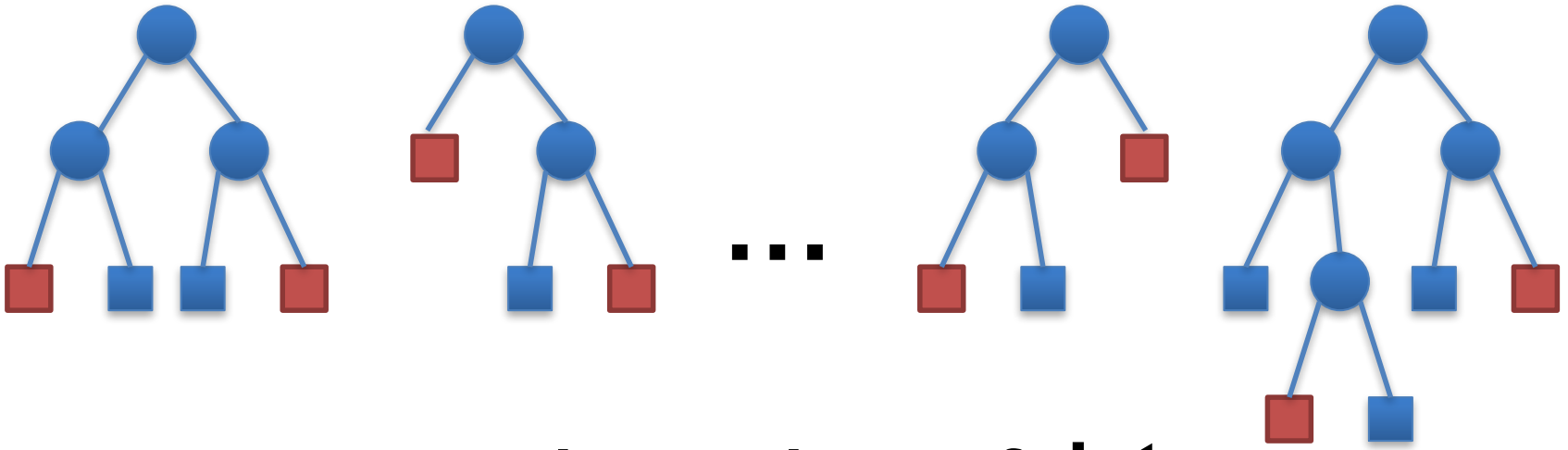

SUPPORT
0
%
0
Confidence: 73.61%
27,980 instances
58.47% of data
Filter predictio


BigML Inc. [US] https://bigml.com/user/czuriaga/gallery/model/554761a4c0eea9546d001295
Total_duration: 934.97
403.28
SUPPORT
0.1
%
5.79
%
OUTPUT
435
3667
EXPECTED ERROR
10.53
3471.38
934.97
Expected Error: 403.28
262 instances
27.55% of data
Prediction path
Total_duration-2
> 164.00
Total_events-3
> 5.00
Perc_device_tablet-3
> 15.61
Perc_device_tv-2
<= 40.92
Perc_device_smartphone-2

Why do we trust in one tree?

- Tree is different depending on data used
- Which attributes are most important?
- **Randomly** select a subset of data, randomly subselect a subset of attributes …

Random Forest



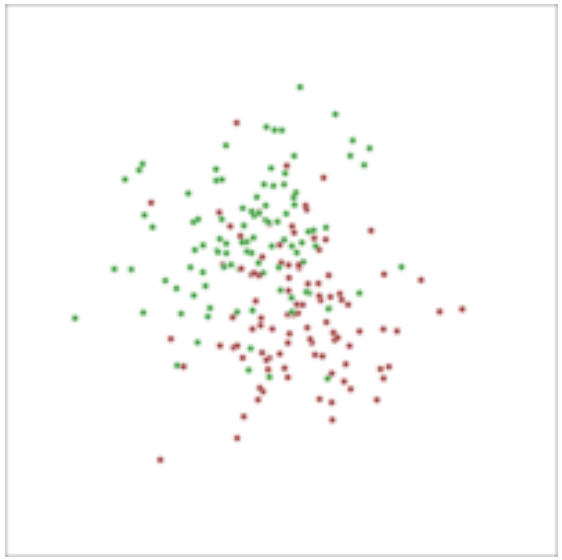
Each trained on a random subset of data

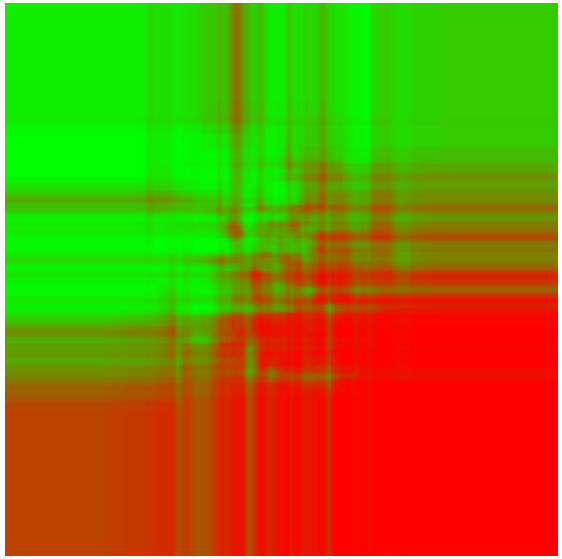
Each trained on a random sub-set of attributes

Every tree provides some level of accuracy.

Majority voting!

Data (2 Gaussians), RF, Logistic regression





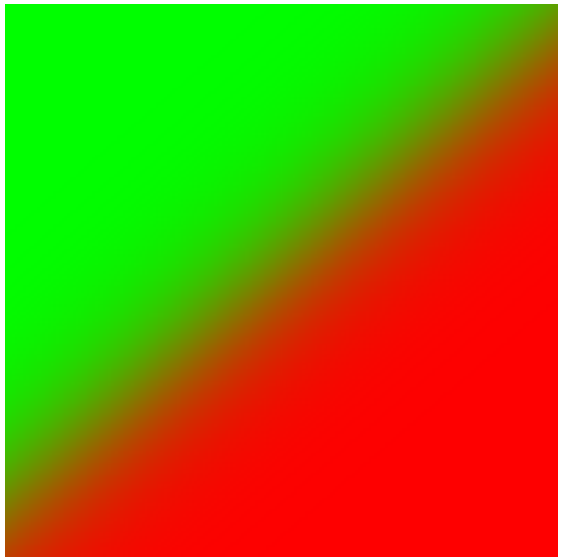


Figure 1.

Resolution: **standard** / high

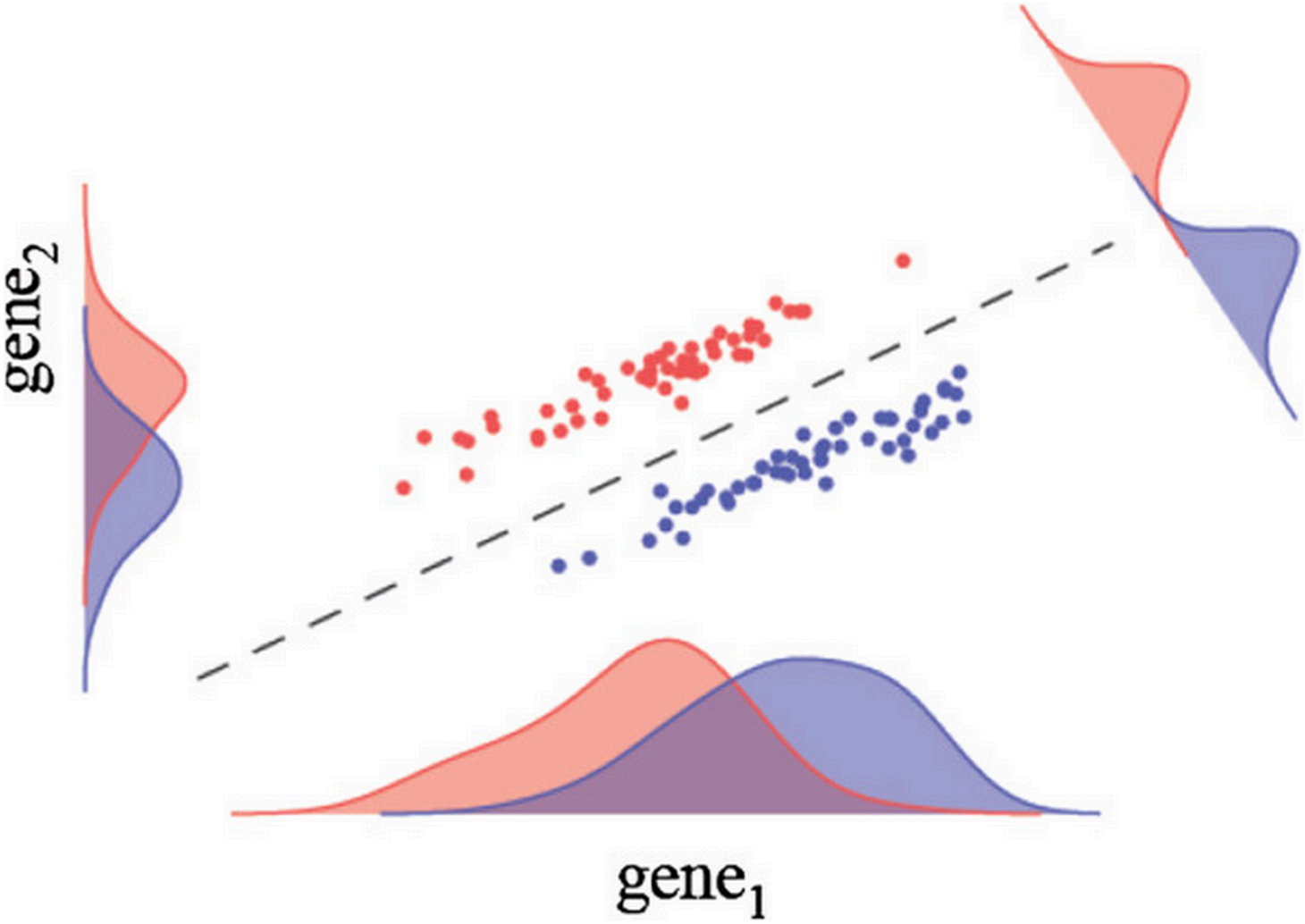


Illustration of a case where there is no marginal differential expression of individual genes, however in the multivariate setting the differential expression becomes clear. Projecting the data onto the appropriate direction in this case leads to a clear separation between the classes.

Clark *et al. BMC Bioinformatics* 2014 **15**:79 doi:10.1186/1471-2105-15-79

References

[BY10] Y. Ben-Haim and E. Yom-Tov. A streaming parallel decision tree algorithm. J. of Machine Learning Research, 11:849–872, 2010.

[B96] L. Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.

[B01] L. Breiman. Random forests. Machine Learning, 45(1):5–32, 2001.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119-139, 1997.

[F01a] Yoav Freund. An adaptive version of the boost by majority algorithm. Machine Learning, 43(3):293--318, 2001.

[F+03] Yoav Freund, Raj Iyer, Robert Schapire, and Yoram Singer. An Efficient Boosting Algorithm for Combining Preferences. Journal of Machine Learning Research 4: 933-969, 2003.

[F01b] Friedman, J. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 25(5):1189-1232, 2001.

[F+00] Jerome Friedman, Trevor Hastie and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. Annals of Statistics 28(2):337-407, 2000.

[H98] Ho, T. K. The random subspace method for constructing decision forests. IEEE PAMI, 20(8):832–844, 1998.

[P+10] D. Y. Pavlov, A. Gorodilov, and C. A. Brunk. BagBoo: a scalable hybrid bagging-the-boosting model. CIKM-2010.

[S+07] Daria Sorokina, Rich Caruana, Mirek Riedewald. Additive Groves of Regression Trees. ECML-2007.

[SUML11-Ch2] Biswanath Panda, Joshua S. Herbach, Sugato Basu, and Roberto J. Bayardo. MapReduce and its Application to Massively Parallel Learning of Decision Tree Ensembles. In “Scaling Up Machine Learning”, Cambridge U. Press, 2011.

[SUML11-Ch8] Krysta M. Svore and Christopher J.C. Burges. Large-scale Learning to Rank using Boosted Decision Trees. In “Scaling Up Machine Learning”, Cambridge U. Press, 2011.

[SUML11-Ch9] Ramesh Natarajan and Edwin Pednault. The Transform Regression Algorithm. In “Scaling Up Machine Learning”, Cambridge U. Press, 2011.

[T+11] Stephen Tyree, Kilian Q. Weinberger, Kunal Agrawal. Parallel Boosted Regression Trees for Web Search Ranking. WWW-2011.

[Y+09] Jerry Ye, Jyh-Herng Chow, Jiang Chen, Zhaohui Zheng. Stochastic Gradient Boosted Distributed Decision Trees. CIKM-2009.

[Z+08] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. NIPS 2008.

Naïve Bayes Classifier

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

The Tennis Dataset

Day	Outlook	Temp	Humidity	Wind	*PlayTennis*
D1	*Sunny*	*Hot*	*High*	*Weak*	*No*
D2	*Sunny*	*Hot*	*High*	*Strong*	*No*
D3	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
D4	*Rain*	*Mild*	*High*	*Weak*	*Yes*
D5	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
D6	*Rain*	*Cool*	*Normal*	*Strong*	*No*
D7	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
D8	*Sunny*	*Mild*	*High*	*Weak*	*No*
D9	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
D10	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
D11	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
D12	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
D13	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
D14	*Rain*	*Mild*	*High*	*Strong*	*No*

Shall we play tennis today?

PlayTennis
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No

Shall we play tennis today?

PlayTennis
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
Yes
No

- Probabilistic model:

P(Yes) = 9/14 = 0.64

P(No) = 5/14 = 0.36

➔ **Yes**

It's windy today. Tennis, anyone?

Wind	PlayTennis
Weak	*No*
Strong	*No*
Weak	*Yes*
Weak	*Yes*
Weak	*Yes*
Strong	*No*
Strong	*Yes*
Weak	*No*
Weak	*Yes*
Weak	*Yes*
Strong	*Yes*
Strong	*Yes*
Weak	*Yes*
Strong	*No*

It's windy today. Tennis, anyone?

- Probabilistic model:

Wind	PlayTennis
Weak	*No*
Strong	*No*
Weak	*Yes*
Weak	*Yes*
Weak	*Yes*
Strong	*No*
Strong	*Yes*
Weak	*No*
Weak	*Yes*
Weak	*Yes*
Strong	*Yes*
Strong	*Yes*
Weak	*Yes*
Strong	*No*

P(Weak) = 8/14

P(Strong) = 6/14

P(Yes | Weak) = 6/8

P(No | Weak) = 2/8

P(Yes | Strong) = 3/6

P(No | Strong) = 3/6

More attributes

- Probabilistic model:

Humidity	Wind	PlayTennis
High	*Weak*	*No*
High	*Strong*	*No*
High	*Weak*	*Yes*
High	*Weak*	*Yes*
Normal	*Weak*	*Yes*
Normal	*Strong*	*No*
Normal	*Strong*	*Yes*
High	*Weak*	*No*
Normal	*Weak*	*Yes*
Normal	*Weak*	*Yes*
Normal	*Strong*	*Yes*
High	*Strong*	*Yes*
Normal	*Weak*	*Yes*
High	*Strong*	*No*

P(High,Weak) = 4/14

P(Yes | High,Weak) = 2/4

P(No | High,Weak) = 2/4

P(High,Strong) = 3/14

P(Yes | High,Strong) = 1/3

P(No | High,Strong) = 2/3

…

The Bayes Classifier

In general:

1. **Estimate from data:**

$$P(\text{Class} \mid X_1, X_2, X_3, \ldots)$$

2. **For a given instance** $(X_1, X_2, X_3 \ldots)$ **predict class whose conditional probability is greater:**

$$P(C_1 \mid X_1, X_2, X_3, \ldots) > P(C_2 \mid X_1, X_2, X_3, \ldots)$$

➔ predict C_1

The Bayes Classifier

In general:

1. $P(C_1 \mid \mathbf{X}) > P(C_2 \mid \mathbf{X})$

   ➔ predict C_1

The Bayes Classifier

In general:

1. $P(C_1 \mid \mathbf{X}) > P(C_2 \mid \mathbf{X})$

➔ predict C_1

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

The Bayes Classifier

In general:

1. $P(C_1 \mid \mathbf{X}) > P(C_2 \mid \mathbf{X})$
   ➔ predict C_1
2. $P(\mathbf{X} \mid C_1)P(C_1)/P(\mathbf{X}) > P(\mathbf{X} \mid C_2)P(C_2)/P(\mathbf{X})$
   ➔ predict C_1

The Bayes Classifier

In general:

1. $P(C_1 \mid \mathbf{X}) > P(C_2 \mid \mathbf{X})$
   ➔ predict C_1
2. $P(\mathbf{X} \mid C_1)P(C_1)/P(\mathbf{X}) > P(\mathbf{X} \mid C_2)P(C_2)/P(\mathbf{X})$
   ➔ predict C_1
3. $P(\mathbf{X} \mid C_1)P(C_1) > P(\mathbf{X} \mid C_2)P(C_2)$
   ➔ predict C_1

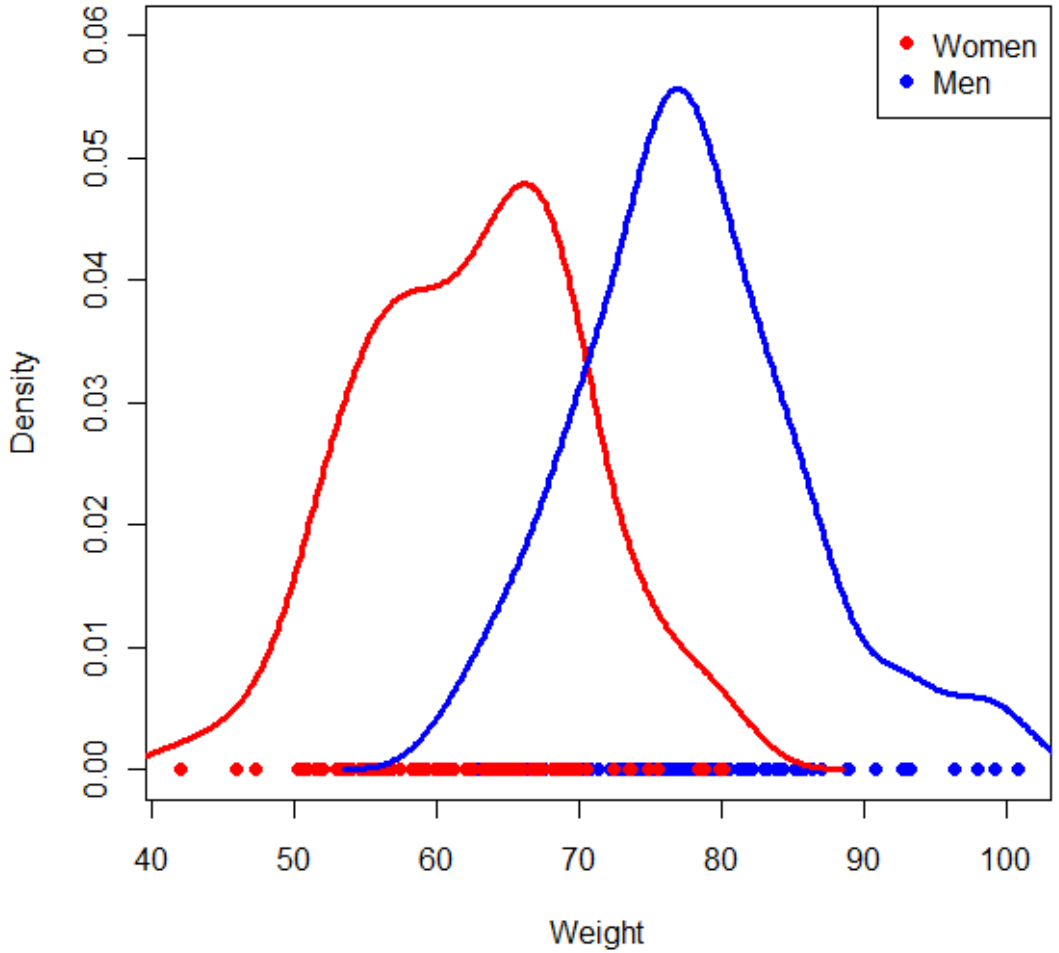
The Bayes Classifier

In general:

1. $P(C_1 \mid \mathbf{X}) > P(C_2 \mid \mathbf{X})$
   ➔ predict C_1
2. $P(\mathbf{X} \mid C_1)P(C_1)/P(\mathbf{X}) > P(\mathbf{X} \mid C_2)P(C_2)/P(\mathbf{X})$
   ➔ predict C_1
3. $P(\mathbf{X} \mid C_1)P(C_1) > P(\mathbf{X} \mid C_2)P(C_2)$
   ➔ predict C_1
4. $P(\mathbf{X} \mid C_1)/P(\mathbf{X} \mid C_2) > P(C_2)/P(C_1)$
   ➔ predict C_1

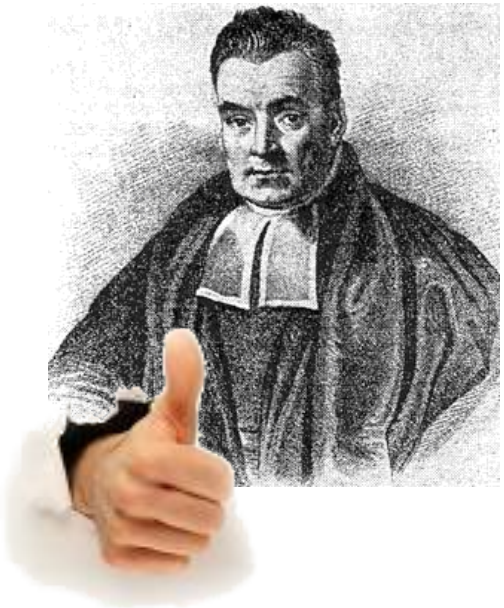
The Bayes Classifier

3. $P(\mathbf{X} \mid C_1)P(C_1) > P(\mathbf{X} \mid C_2)P(C_2)$ ➔ predict C_1



The Bayes Classifier

If **the true** underlying distribution is known, the Bayes classifier is **optimal** (i.e. it achieves minimal error probability).



Problem

- **We need exponential amount of data**

Humidity	Wind	PlayTennis
High	*Weak*	*No*
High	*Strong*	*No*
High	*Weak*	*Yes*
High	*Weak*	*Yes*
Normal	*Weak*	*Yes*
Normal	*Strong*	*No*
Normal	*Strong*	*Yes*
High	*Weak*	*No*
Normal	*Weak*	*Yes*
Normal	*Weak*	*Yes*
Normal	*Strong*	*Yes*
High	*Strong*	*Yes*
Normal	*Weak*	*Yes*
High	*Strong*	*No*

P(High,Weak) = 4/14

P(Yes | High,Weak) = 2/4

P(No | High,Weak) = 2/4

P(High,Strong) = 3/14

P(Yes | High,Strong) = 1/3

P(No | High,Strong) = 2/3

…

The Naïve Bayes Classifie

To scale beyond 2-3 attributes, use a trick:

***Assume* that attributes of each class are independent:**

$$P(X_1,X_2,X_3 \mid \text{Class}) = P(X_1|\text{Class})P(X_2|\text{Class})P(X_3|\text{Class})\ldots$$

The Naïve Bayes Classifie

$P(\mathbf{X} \mid C_1)/P(\mathbf{X} \mid C_2) > P(C_2)/P(C_1)$

➔ predict C_1

becomes

$$\prod_i \frac{P(X_i|C_1)}{P(X_i|C_2)} > \frac{P(C_2)}{P(C_1)} \Rightarrow \text{Predict } C_1$$

Naïve Bayes Classifier

1. **Training:**

   For each value *v* of each attribute *i*, compute

$$m(i,v) = \frac{P(X_i = v|C_1)}{P(X_i = v|C_2)}$$

2. **Classification:**

   For a given instance (x_1, x_2, …) compute

$$\text{score} = \prod_i m(i,x_i)$$

   If $\text{score} > \frac{P(C_2)}{P(C_1)}$ classify as C_1 , else C_2

Naïve Bayes Classifier

1. **Training:**

   For each value *v* of each attribute *i*, compute

$$l(i,v) = \log \frac{P(X_i = v|C_1)}{P(X_i = v|C_2)}$$

2. **Classification:**

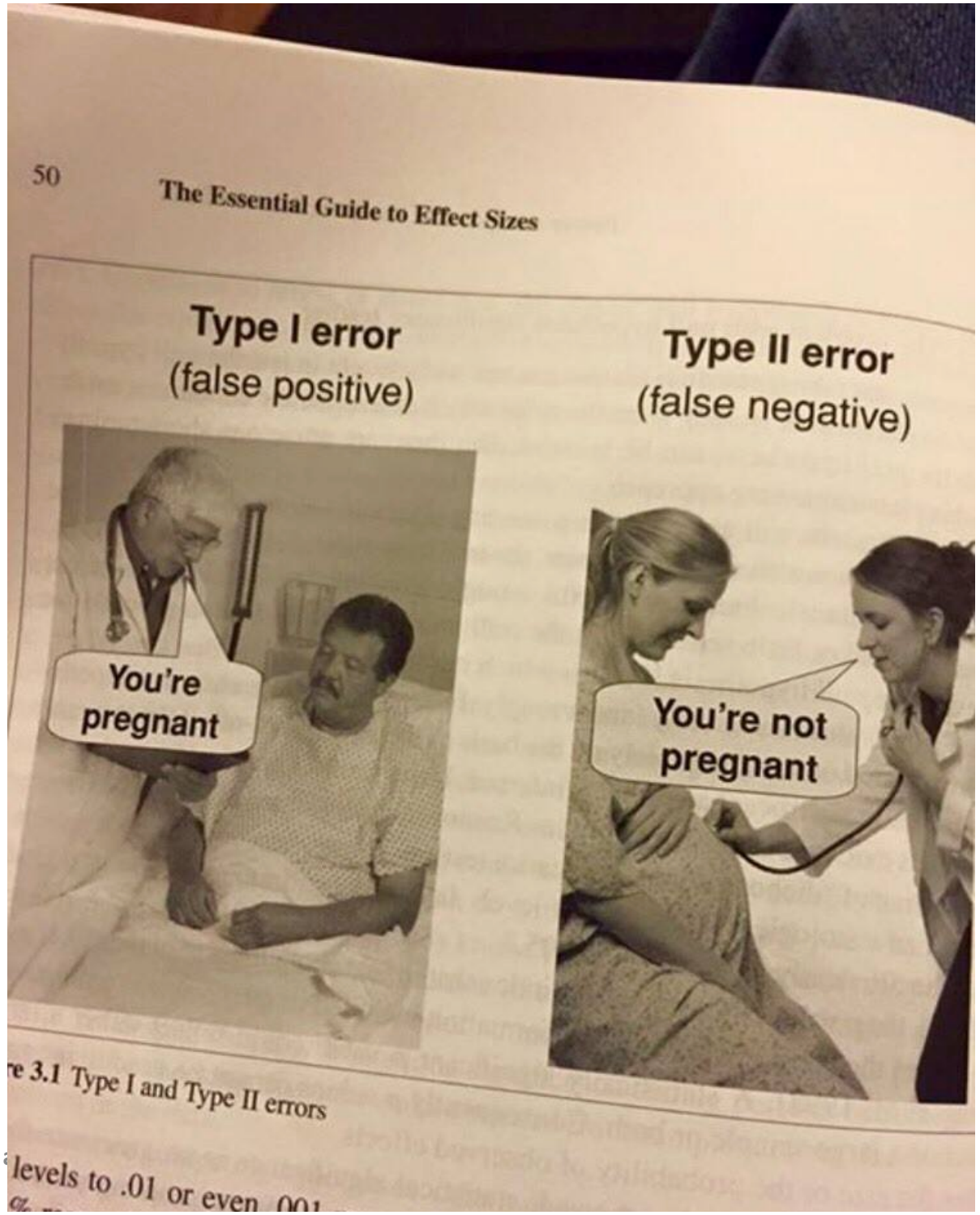
   For a given instance (x_1, x_2, …) compute

$$\mathrm{lscore} = \sum_i l(i,x_i)$$

   If $\mathrm{lscore} > \log \frac{P(C_2)}{P(C_1)}$ classify as C_1 , else C_2

Naïve Bayes Classifier

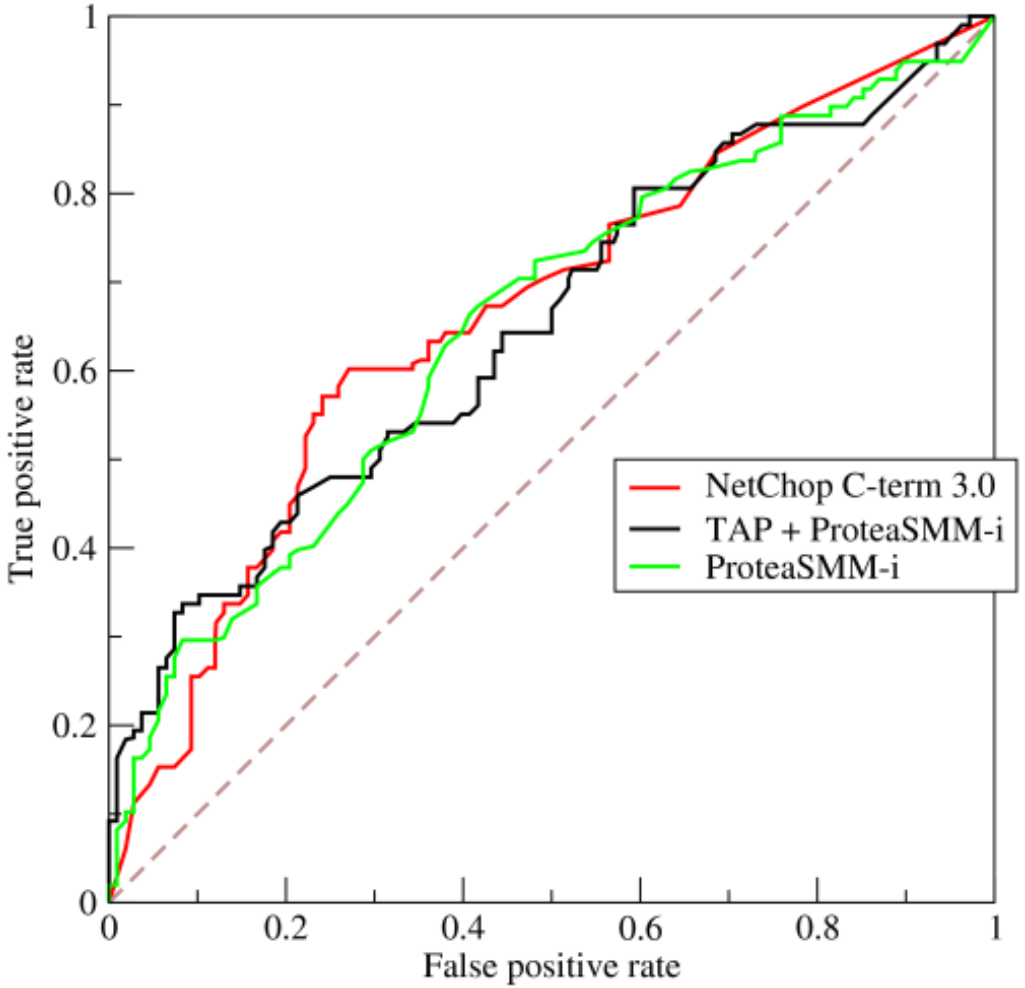
- Extendable to continuous attributes via kernel density estimation.
- The goods:
  - Easy to implement, efficient
  - Won't overfit, intepretable
  - Works better than you would expect (e.g. spam filtering)
- The bads
  - "Naïve", linear
  - Usually won't work well for too many classes

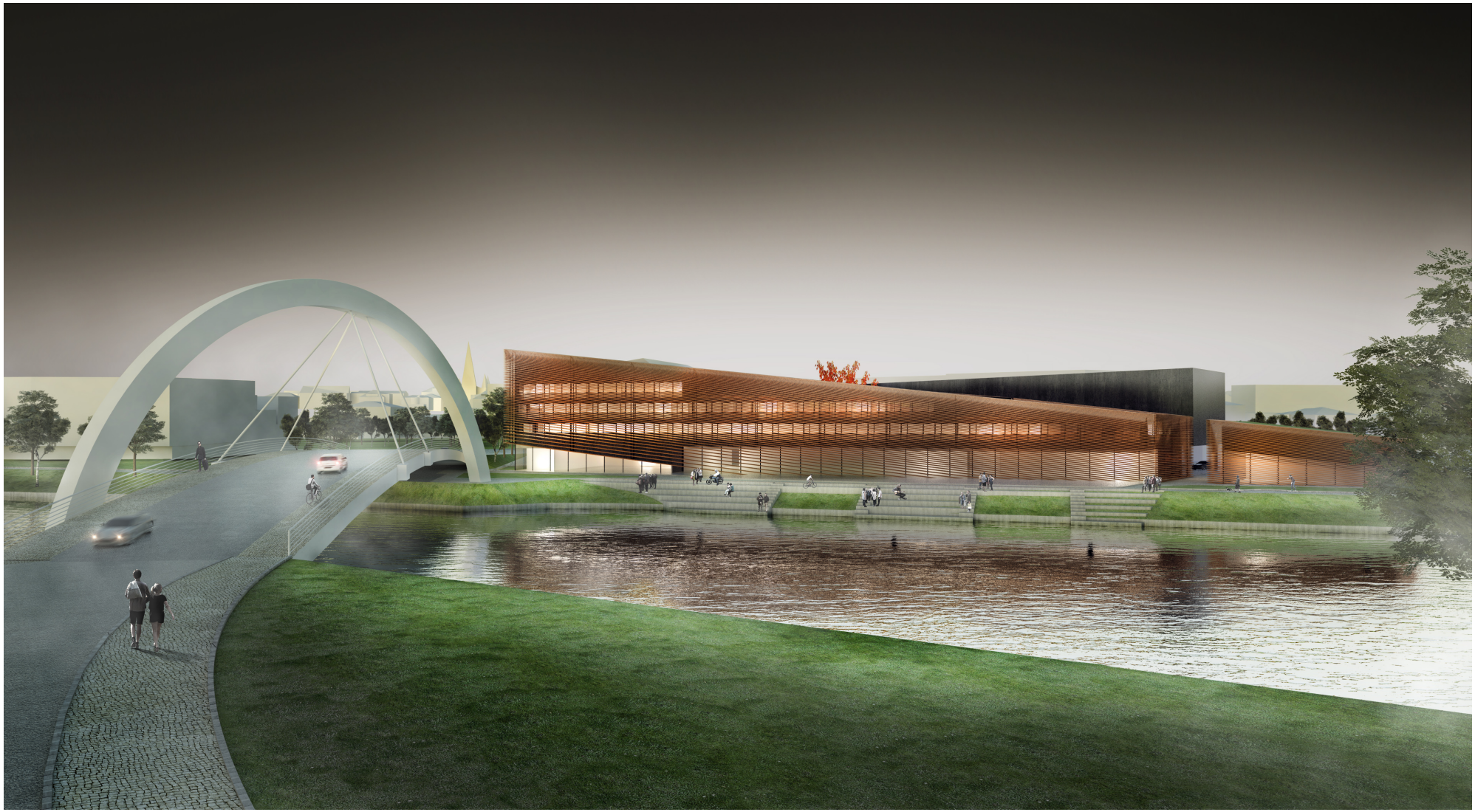


re 3.1 Type I and Type II errors

levels to .01 or even .001

ROC curve – Receiver Operator Characteristic, AUC – Area Under Curve





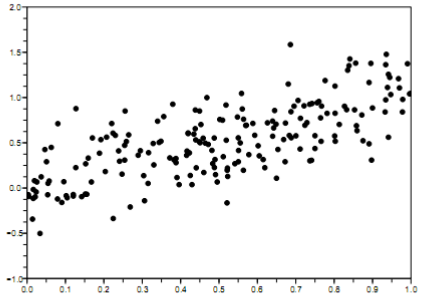
Supervised learning

- **Four examples of approaches**
  - Ad-hoc
    - Decision tree induction
  - Probabilistic modeling
    - Naïve Bayes classifier
  - **Objective function optimization**
    - **Linear least squares regression**
  - **Instance-based methods**
    - **K-nearest neighbors**

Linear regression

- Consider data points of the type

$$(x_i, y_i) \in \mathbb{R}$$



- Let us search for a function of the form

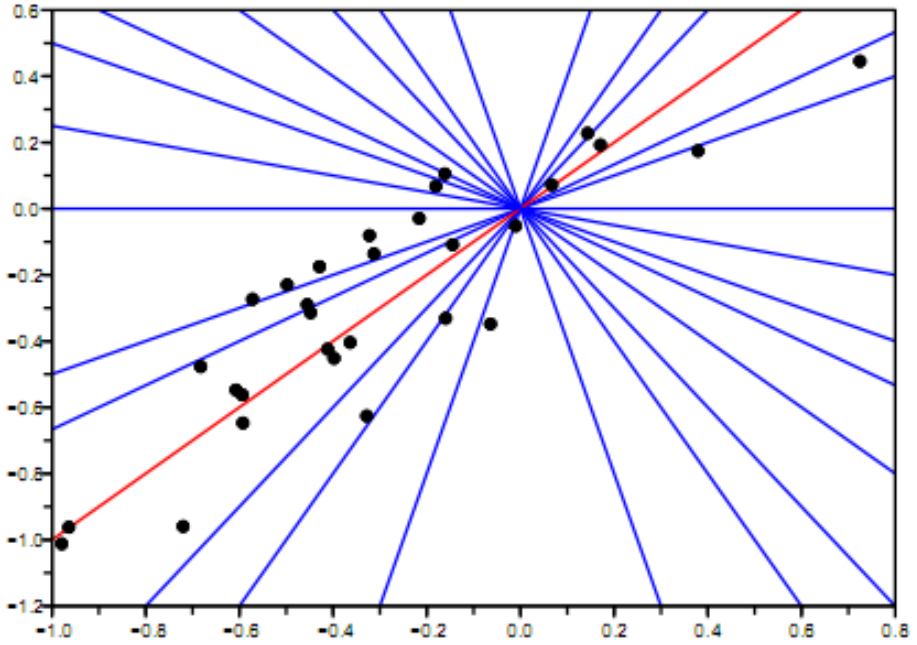
$$f(x) = w\,x$$

that would have **the least possible sum of error squares:**

$$E(w) = \sum_i (f(x_i) - y_i)^2$$

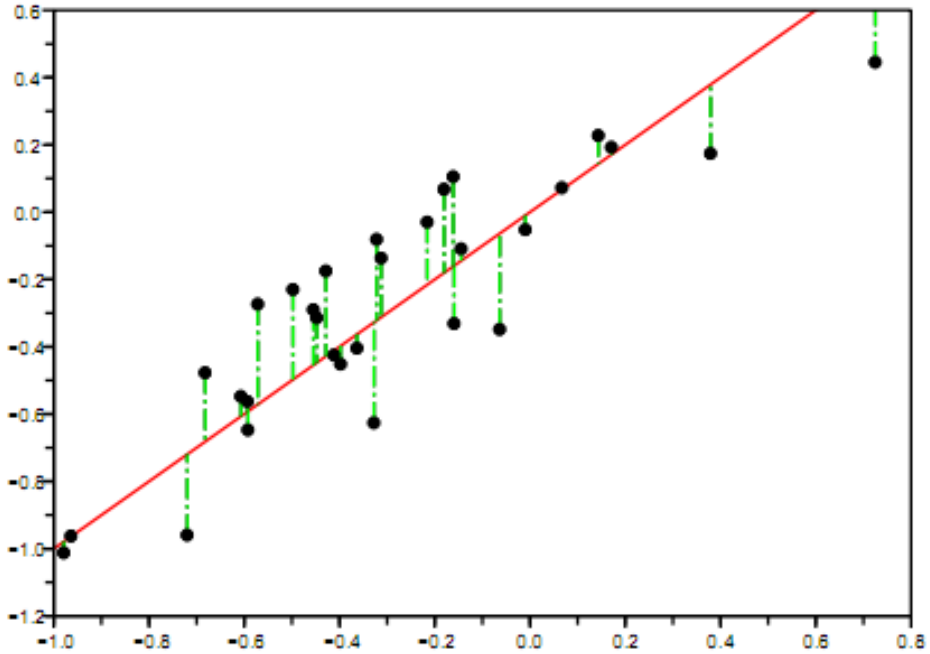
Linear regression

- Set of functions $f(x) = wx$ for various w.



Linear regression

- The function having the least error



Linear regression

- We need to solve: $\arg\min_w E(w)$

  where $$E(w) = \sum_i (w x_i - y_i)^2$$

- Setting the derivative to 0:

  we get $$E'(w) = \sum_i 2(w x_i - y_i) x_i = 0$$

$$w = \sum_i y_i x_i / \sum_i x_i^2$$

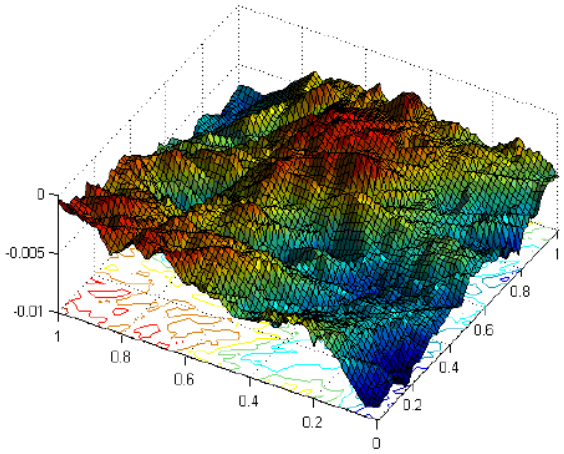
Linear regression

- The idea naturally generalizes to more complex cases (e.g. multivariate regression)
- The goods
  - Simple, easy
  - Intepretable, popular, extendable to many variants
  - Won't overfit
- The bads
  - Linear, i.e. works for a limited set of datasets.
  - Nearly never perfect.

Median Error Regression (quantile)

- Minimize mean error
- More robust: minimize median error
- Requires different optimisation
  - **Differential Evolution**
  - Genetic Programming
  - …

- **search spaces**



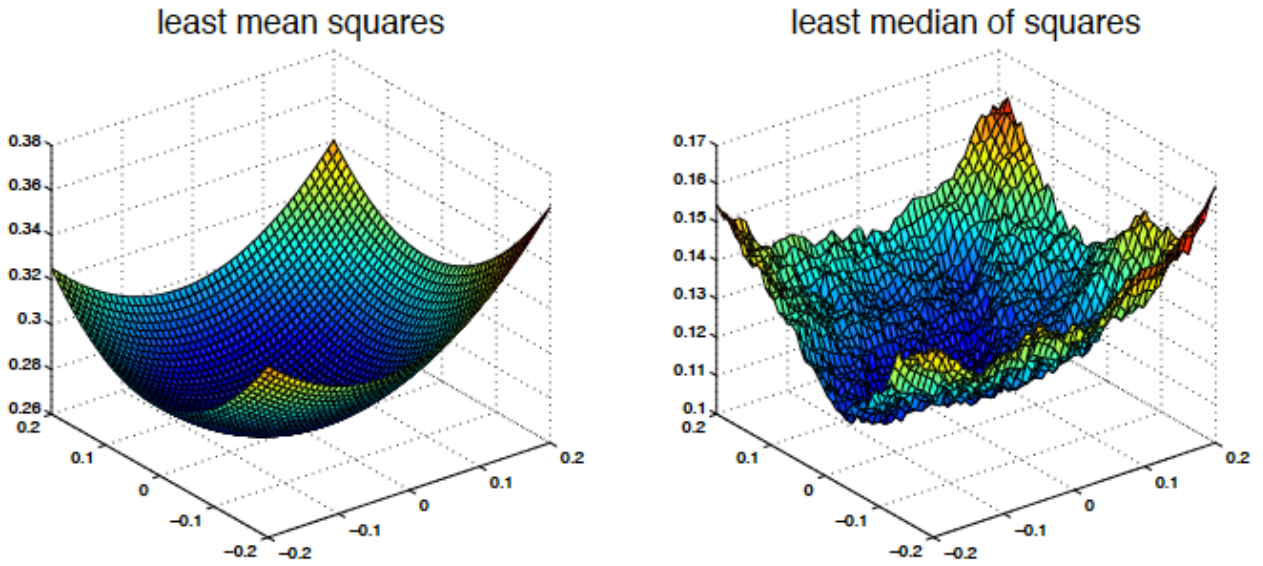
Robust Regression

least quantile of squares

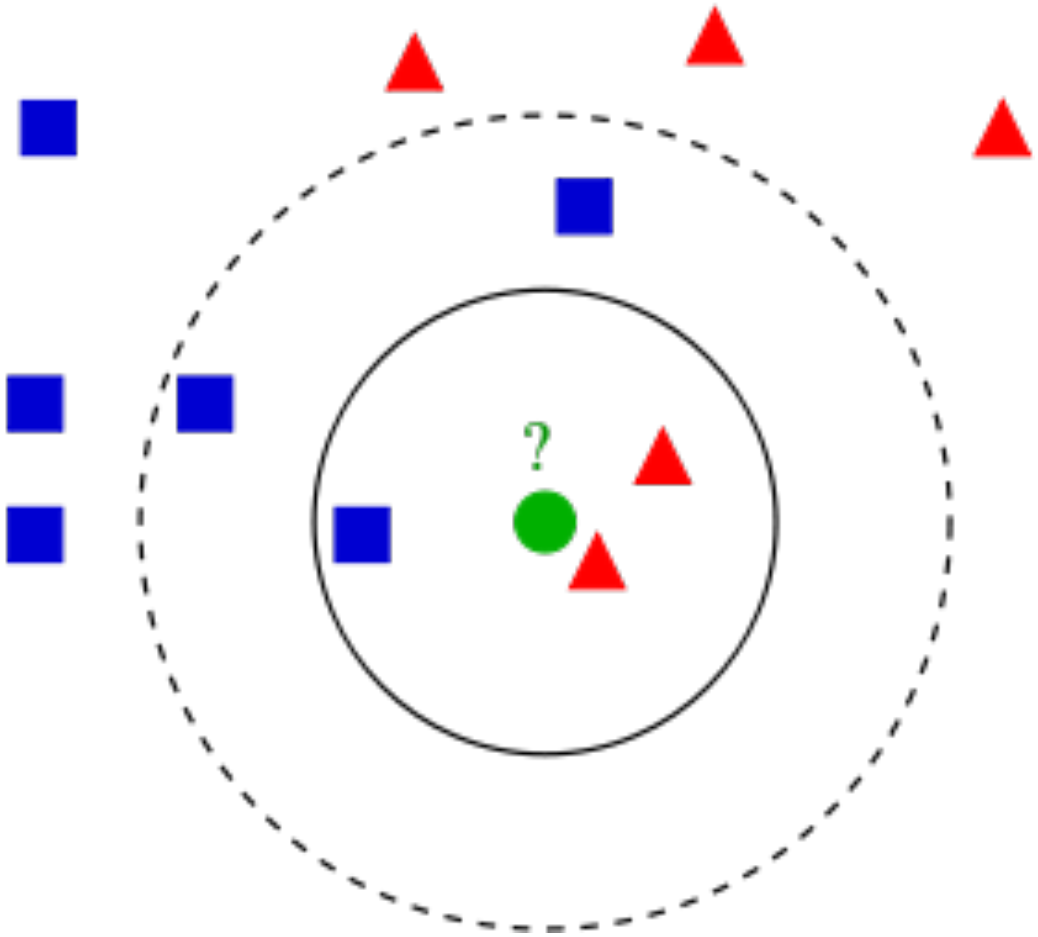
(Gilli, Maringer and Schumann, 2011)

$$\min_{\beta} e^2_{(\alpha N)} \quad \text{where} \quad \mathbf{e} = \mathbf{X}\beta - \mathbf{y}$$

$$e^2_{(j-1)} \leqslant e^2_{(j)} \; j = 2..N$$

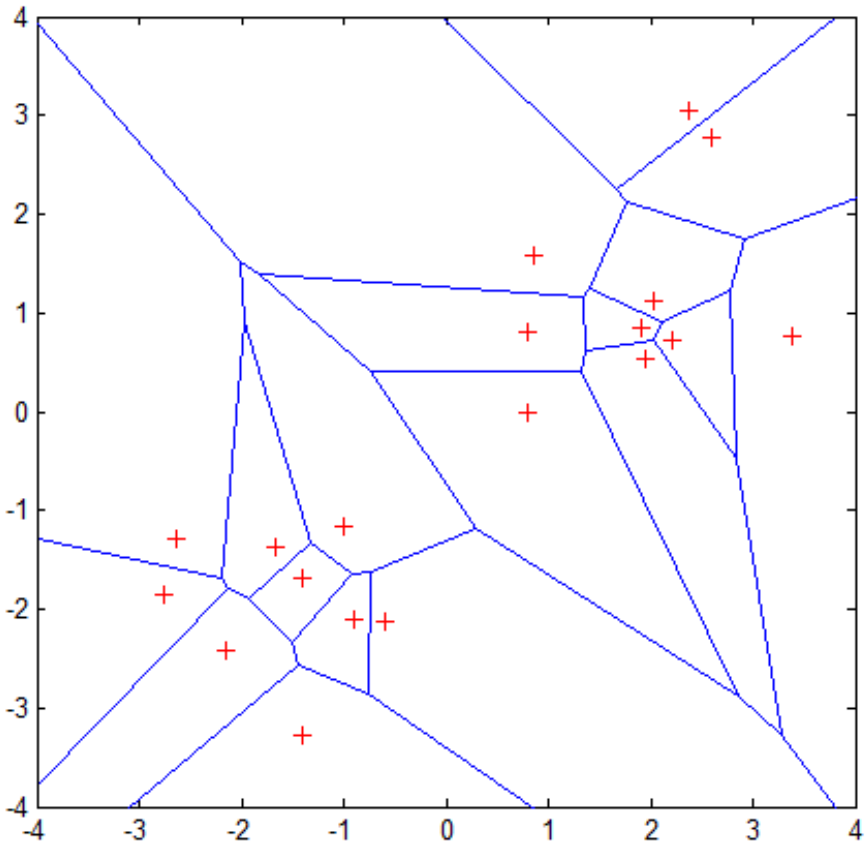


K-Nearest Neighbors



K-Nearest Neighbors

- **One-nearest neighbor**



K-Nearest Neighbors

- **Training:**
  - Store and index all training instances

- **Classification**
  - Given an instance to classify, find *k* instances from the training sample nearest to it.
  - Predict the majority class among these k instances

K-Nearest Neighbors

- **The goods**
  - Trivial concept
  - Easy to implement
  - Asymptotically optimal
  - Allows various kinds of data
- **The bads**
  - Difficult to implement efficiently
  - Not interpretable (no model)
  - On smaller datasets looses to other methods

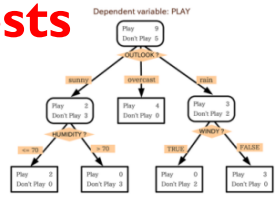
Supervised learning

Ad-hoc

Decision trees, forests

Rule induction, ILP

Fuzzy reasoning



…

$$H(p) = -\sum_i p_i \log_2 p_i$$

Probabilistic models

Naïve Bayes $\prod_i \frac{P(X_i|C_1)}{P(X_i|C_2)} \gtrless \frac{P(C_2)}{P(C_1)}$

Graphical models

Regression models

Density estimation

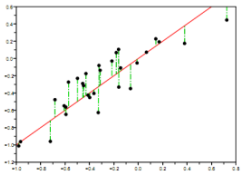
…

Objective optimization

Regression models

Kernel methods, SVM, RBF

Neural networks



…

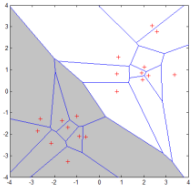
$$\arg\min_w E(w)$$

Instance-based

K-NN, LOWESS

Kernel densities

SVM, RBF

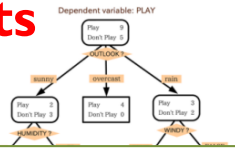


…

Supervised learning

Ad-hoc

Decision trees, forests



Rule induction, ILP

Fuzzy reason

...

$$H(\mathcal{L}$$

Probabilistic models

Naïve Bayes $\prod_i \frac{P(X_i|C_1)}{P(X_i|C_2)} \gtrless \frac{P(C_2)}{P(C_1)}$

Graphical models

odels

ation

Ensemble-learners

Arcing, Boosting,

Bagging, Dagging,

Voting, Stacking

Objective c

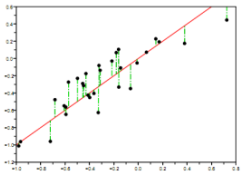
Regression mo

Kernel methods, SVM, RBF

Neural networks

...

$$\underset{w}{\operatorname{argmin}} E(w)$$



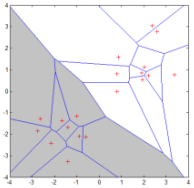
ased

SS

Kernel densities

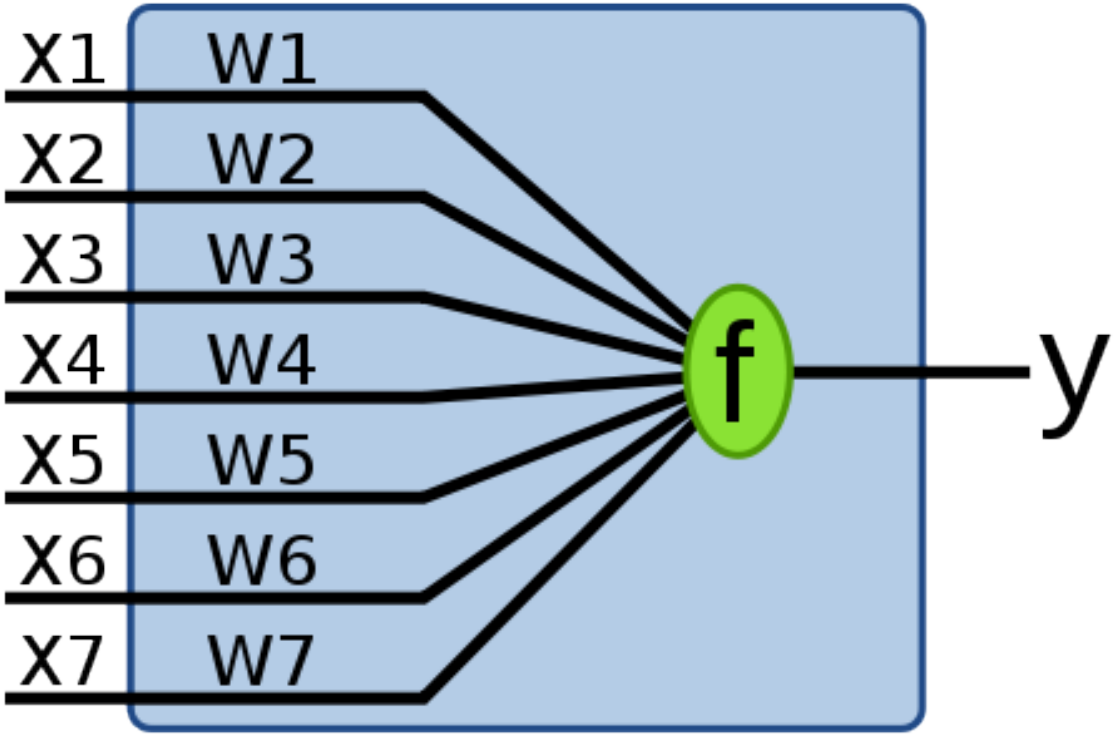
SVM, RBF

...

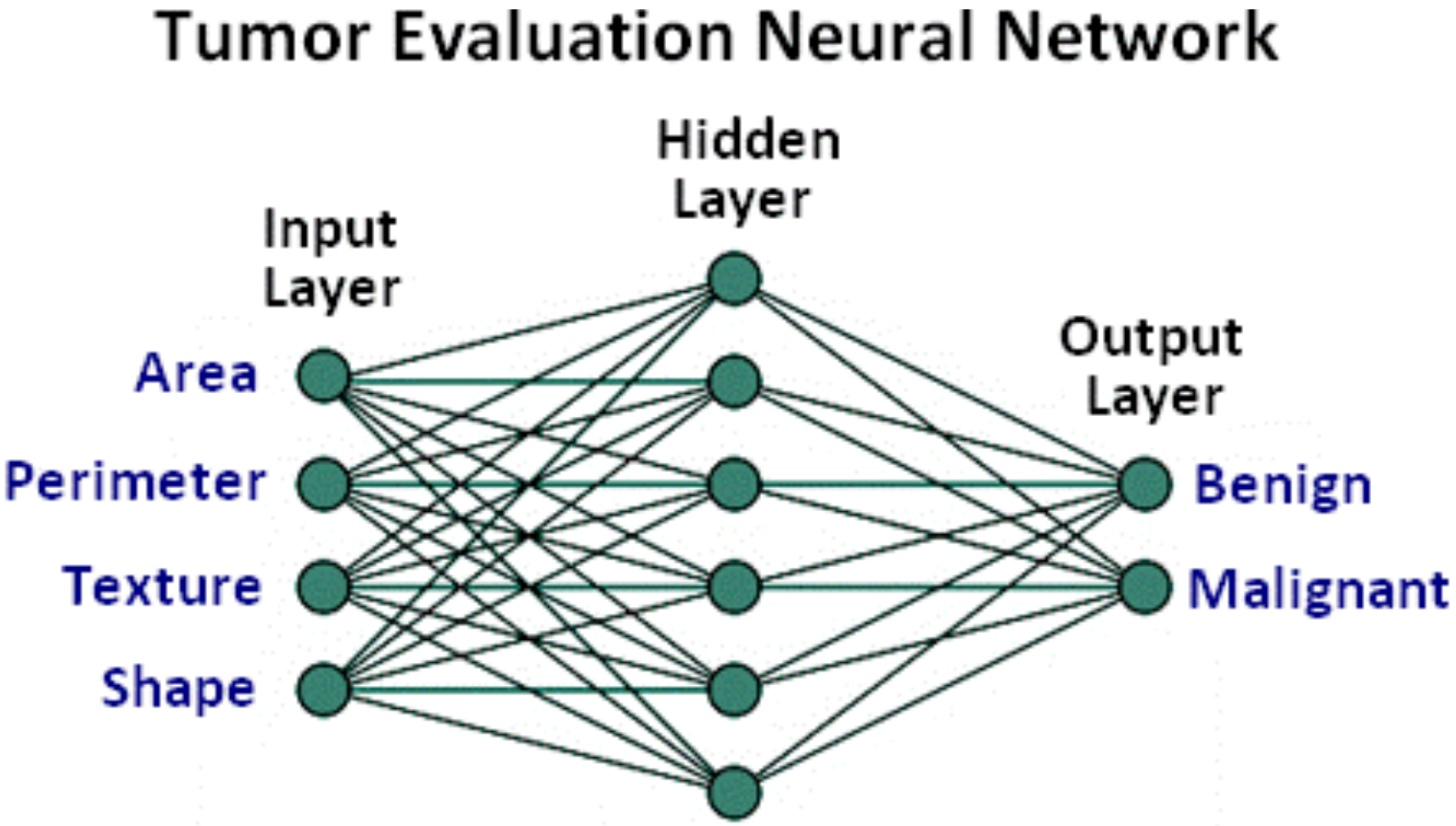


Perceptron

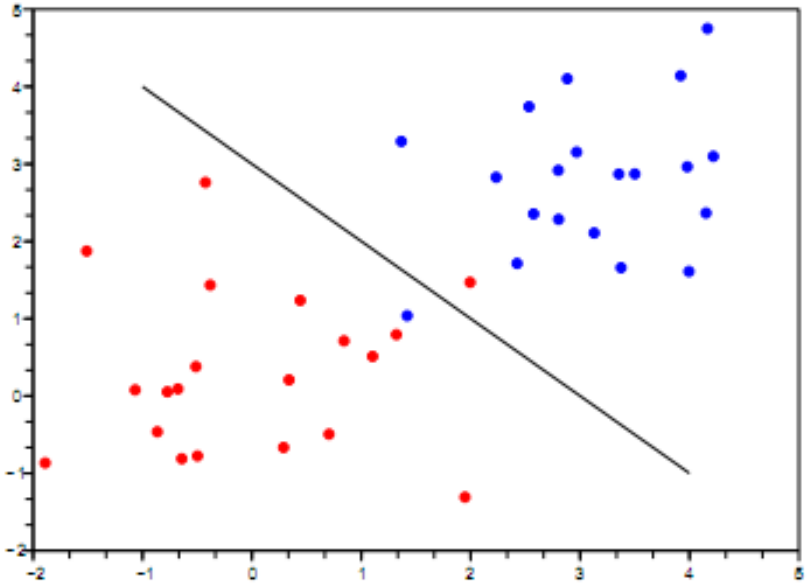
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



Neural Network

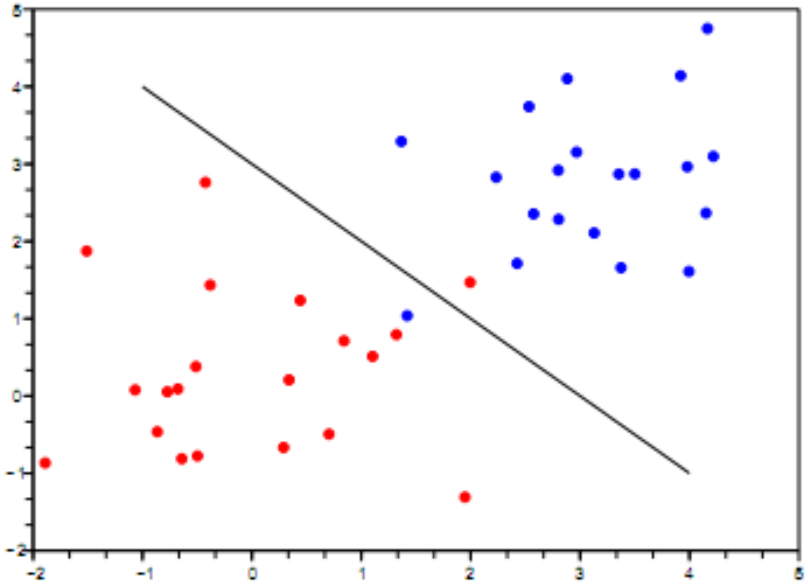


Linear classification



$$f(x_1, x_2) = \mathrm{sign}(x_1 + x_2 - 3)$$

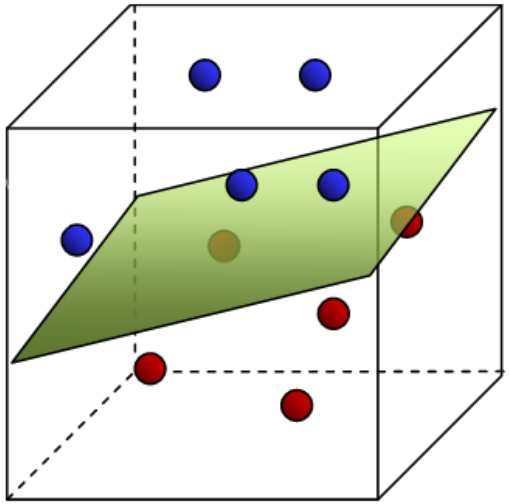
Linear classification



$$f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

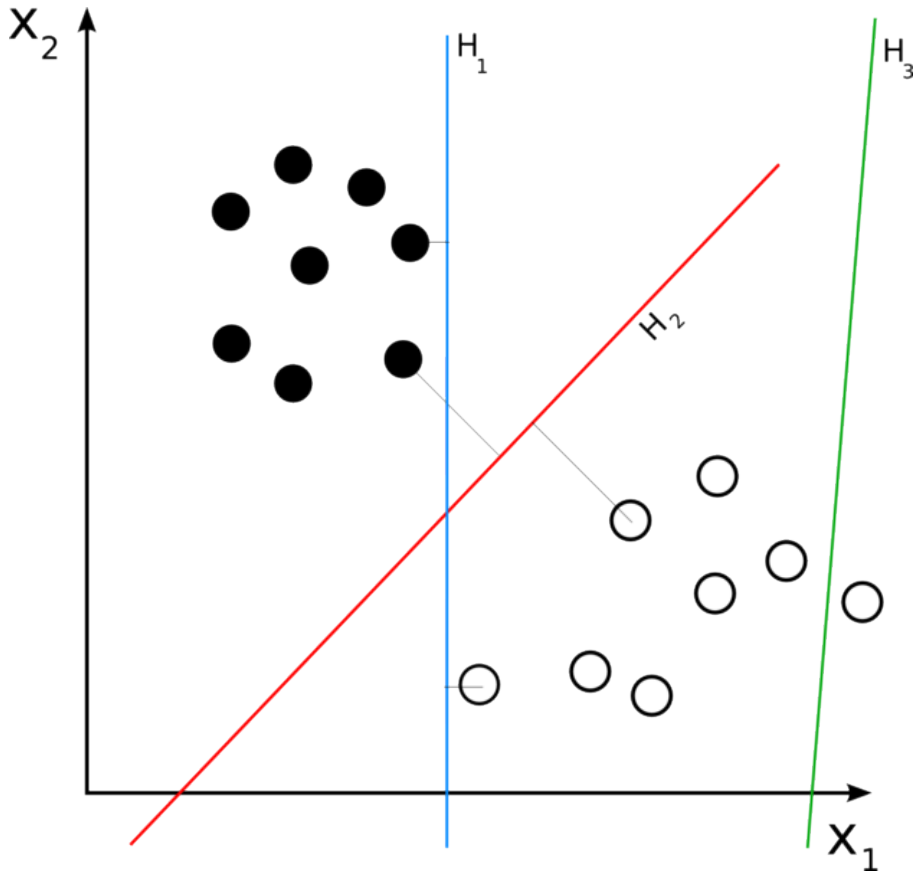
$$\boldsymbol{x} = (x_1, x_2) \quad \boldsymbol{w} = (1,1) \quad b = -3$$

Linear classification



$$f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

Linear classifiers

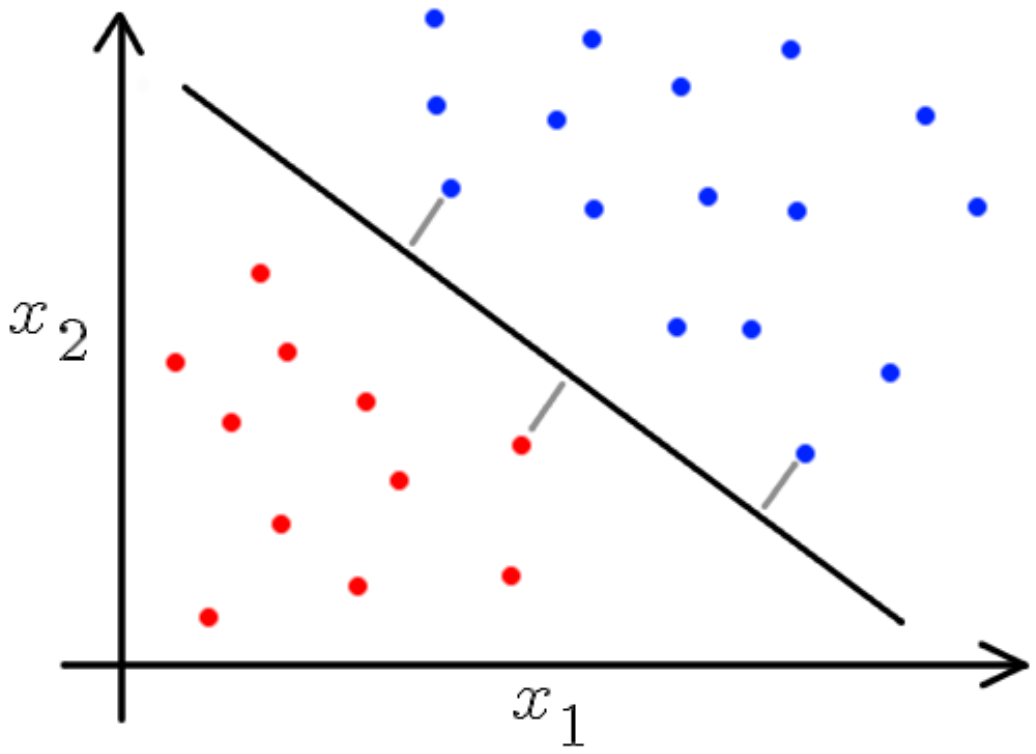


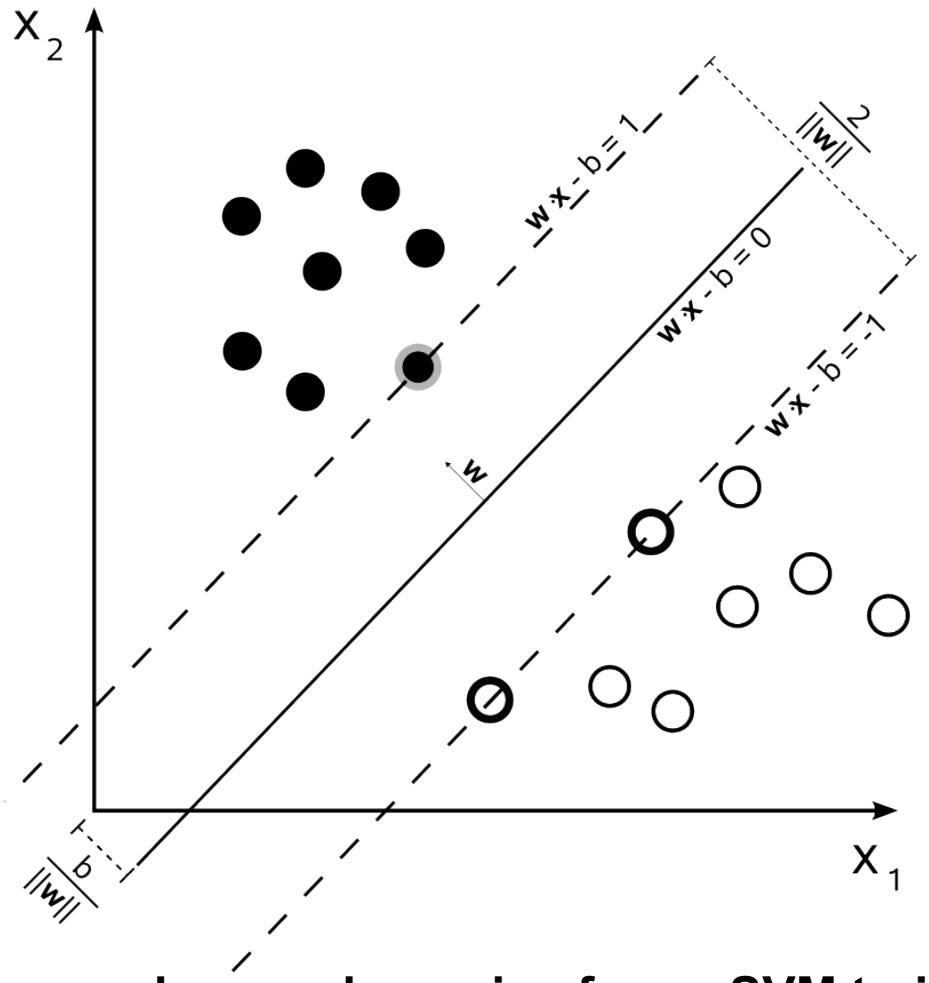
H3 (green) doesn't separate the two classes. H1 (blue) does, with a small margin and H2 (red) with the maximum margin.

Linear classification

- Learning a linear classifier from data:
  - Minimizing **training error**
    - NP-complete
  - Minimizing **sum of error squares**
    - Suboptimal, yet can be easy and fun: e.g. the perceptron.
  - Maximizing the **margin**
    - Doable and well-founded by statistical learning theory

The margin





Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

The margin

- For any point *x*, its distance to the hyperplane:

$$\text{Distance} = \frac{|\boldsymbol{w} \cdot \boldsymbol{x} + b|}{\|\boldsymbol{w}\|}$$

- Assuming all points are classified correctly:

$$\text{Distance} = \frac{y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b)}{\|\boldsymbol{w}\|}$$

- The margin is then:

$$\text{Margin} = \min_i \frac{y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}$$

Maximal margin training

- Find (w, b) such that Margin is maximal, can be shown to be equivalent with:

$$\widehat{\boldsymbol{w}} = \underset{w,b}{\operatorname{argmin}} \|\boldsymbol{w}\|^2$$

subject to

$$\forall i: \; y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$$

- This is doable using efficient optimization algorithms.

Soft margin training

- If data is not linearly separable:

$$\widehat{\boldsymbol{w}} = \underset{w,b}{\operatorname{argmin}} \sum \varepsilon_i + \lambda \|\boldsymbol{w}\|^2$$

subject to

$$\forall i: \; y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \varepsilon_i \qquad \varepsilon_i \geq 0$$

- It is called the Support Vector Machine (SVM).
- It can be generalized to regression tasks.

Soft margin training

- In more general form:

$$\widehat{\boldsymbol{w}} = \underset{w,b}{\operatorname{argmin}} \sum \varepsilon_i + \lambda \|\boldsymbol{w}\|^2$$

where

$$\varepsilon_i = (1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b))^+$$

is the *hinge loss*.

Regularization

- There are many algorithms which essentially look as follows:

  For given data D find a model M, which minimizes

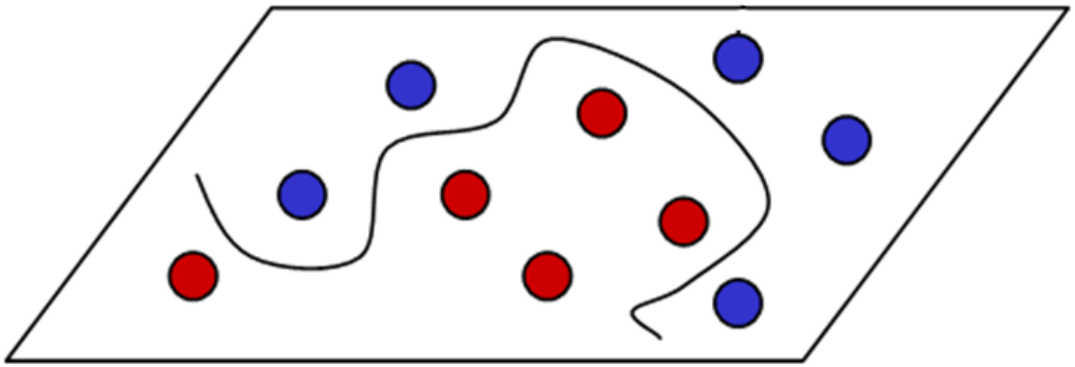
  Error(M,D)+ Complexity(M)

- An SVM is a linear model, which minimizes

  Hinge loss + l_2-norm penalty

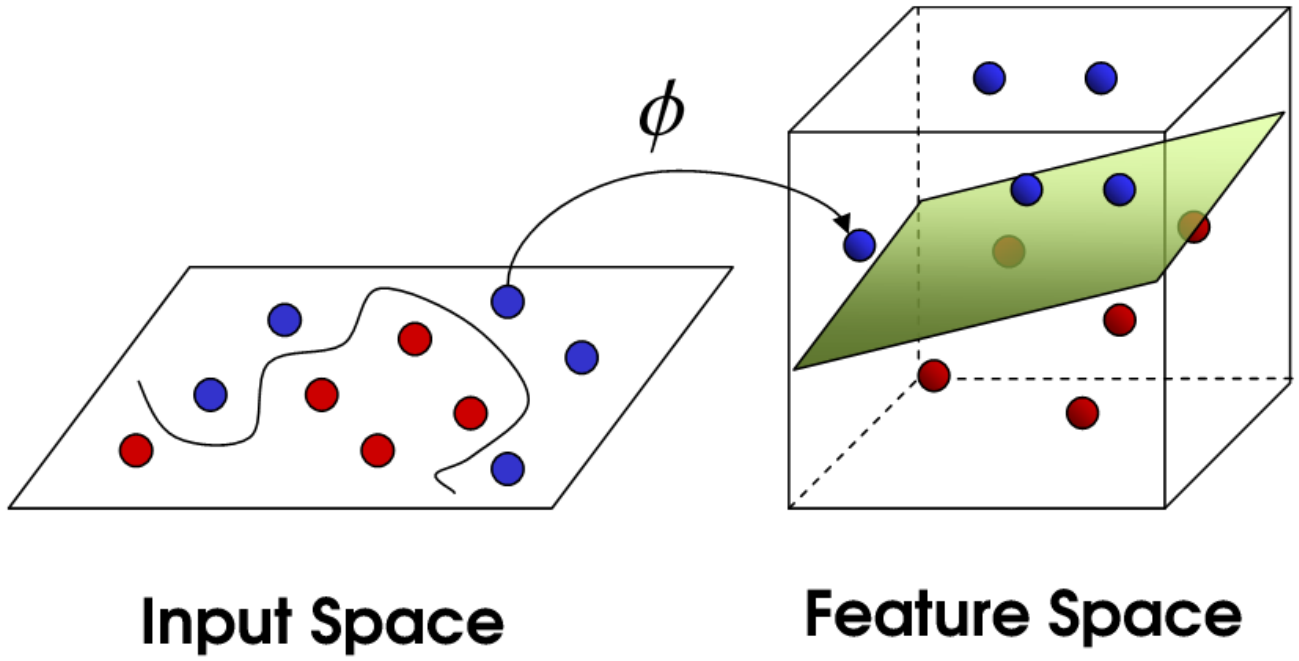
Going nonlinear

- **But a linear classifier is so linear!**

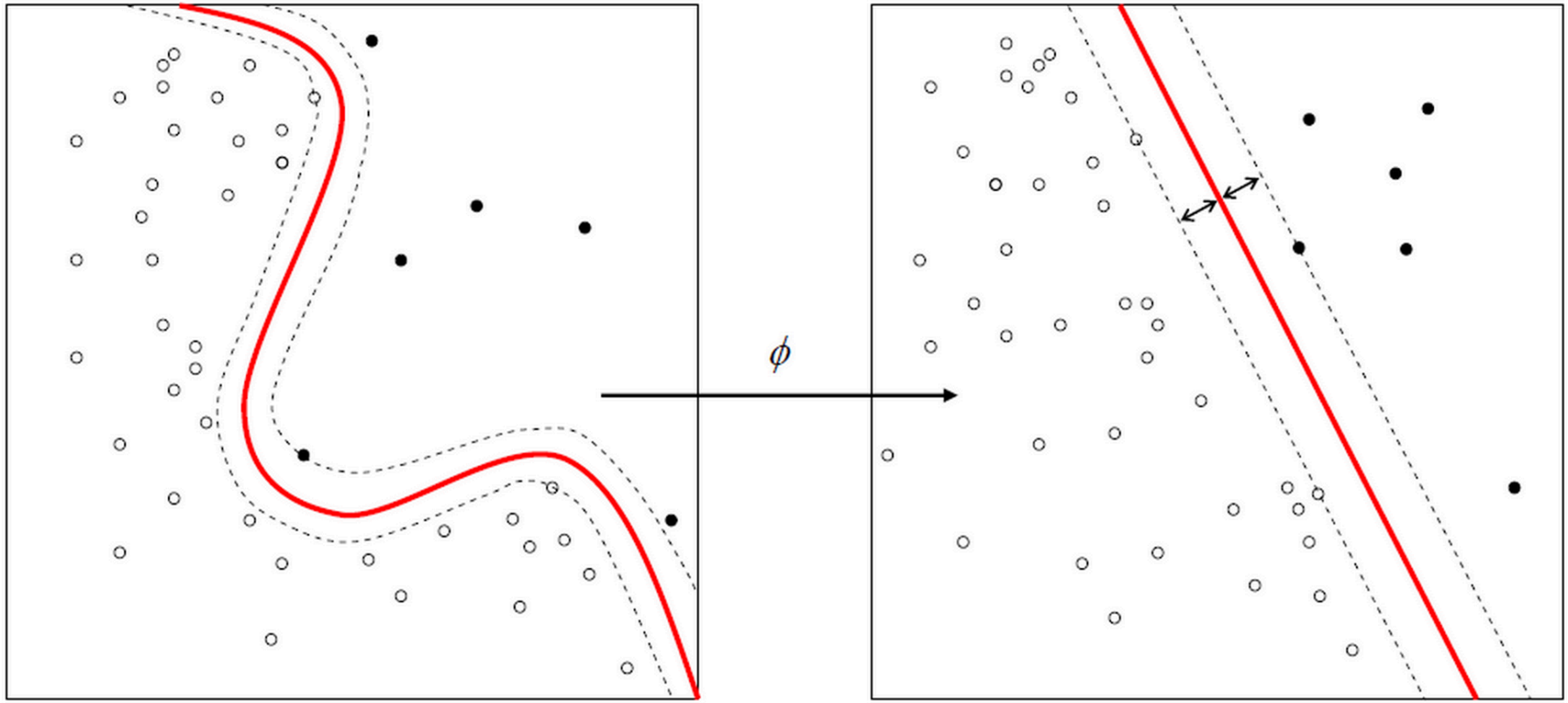


Solution: a nonlinear map

- Instead of classifying points x, we'll classify points $\phi(x)$ in a higher-dimensional space.



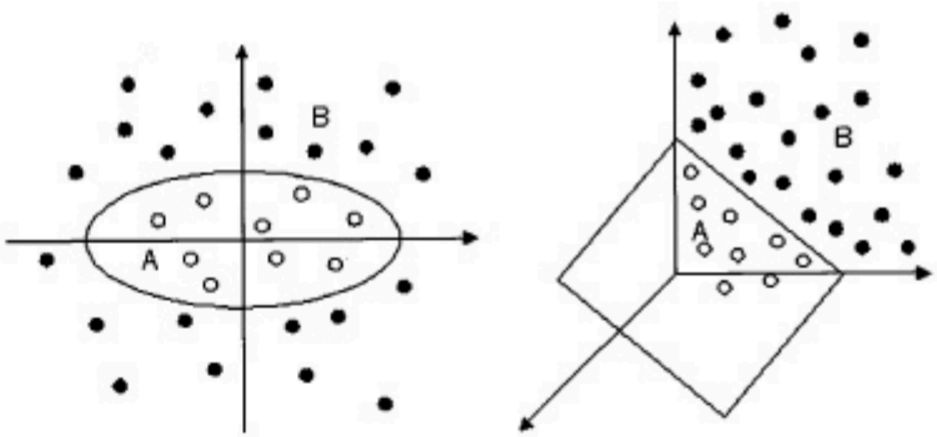
Linear separation in non-linear world



http://news360.com/digestarticle/t-2ryjS4d0io86BLq8d-qA

Mapping Example

- map data points into feature space with some function ϕ
- e.g.:
  - $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$
  - $(x_2, x_2) \rightarrow (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$



- hyperplane $\langle w \cdot z \rangle = 0$, as a function of x:

$$w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 = 0$$

Kernel methods (not density!)

- For nearly *any* linear classifier:

$$f(x) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

- Trained on a dataset

$$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)\}$$

- The resulting vector w can be represented as:

$$\widehat{\boldsymbol{w}} = \sum_i \alpha_i \boldsymbol{x}_i$$

- Which means:

$$f(x) = \mathrm{sign}(\sum_i \alpha_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b)$$

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Kernel methods (not density!)

$$f(x) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

$$f(x) = \mathrm{sign}(\sum_i \alpha_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b)$$

$$f(x) = \mathrm{sign}(\sum_i \alpha_i \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}) + b)$$

$$f(x) = \mathrm{sign}(\sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b)$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x})$$

Kernel methods (not density!)

$$f(x) = \mathrm{sign}\left(\sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$$

- Function K is called a kernel, it measures similarity between objects.
- The computation of $\phi(\boldsymbol{x})$ is unnecessary.
- You can use any type of data.
- Your method is nonlinear.
- Any linear method can be kernelized.
- Kernels can be combined.

Example Kernel Trick

$$\vec{x} = (x_1, x_2)$$
$$\vec{z} = (z_1, z_2)$$
$$K(x, z) = \langle \vec{x} \cdot \vec{z} \rangle^2$$

$$\begin{aligned} K(x,z) &= \langle \vec{x} \cdot \vec{z} \rangle^2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2) \\ &= \left\langle (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \cdot (z_1^2, \sqrt{2} z_1 z_2, z_2^2) \right\rangle \\ &= \langle \phi(\vec{x}) \cdot \phi(\vec{z}) \rangle \end{aligned}$$

mapping function ϕ fused in K

$\rightarrow$ implicit $\phi(\vec{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$

Typical Kernels

- **Polynomial Kernel**

$$K(x,z) = \left(\langle x \cdot z \rangle + \theta\right)^d, \qquad for \;\; d \geq 0$$

- **Radial Basis Function** (Gaussian Kernel)

$$K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}} \qquad \|x\| := \sqrt{\langle x \cdot x \rangle}$$

- **(Sigmoid Kernel)**

$$K(x,z) = tanh(\eta \langle x \cdot z \rangle + \theta$$

- **Inverse multi-quadric**

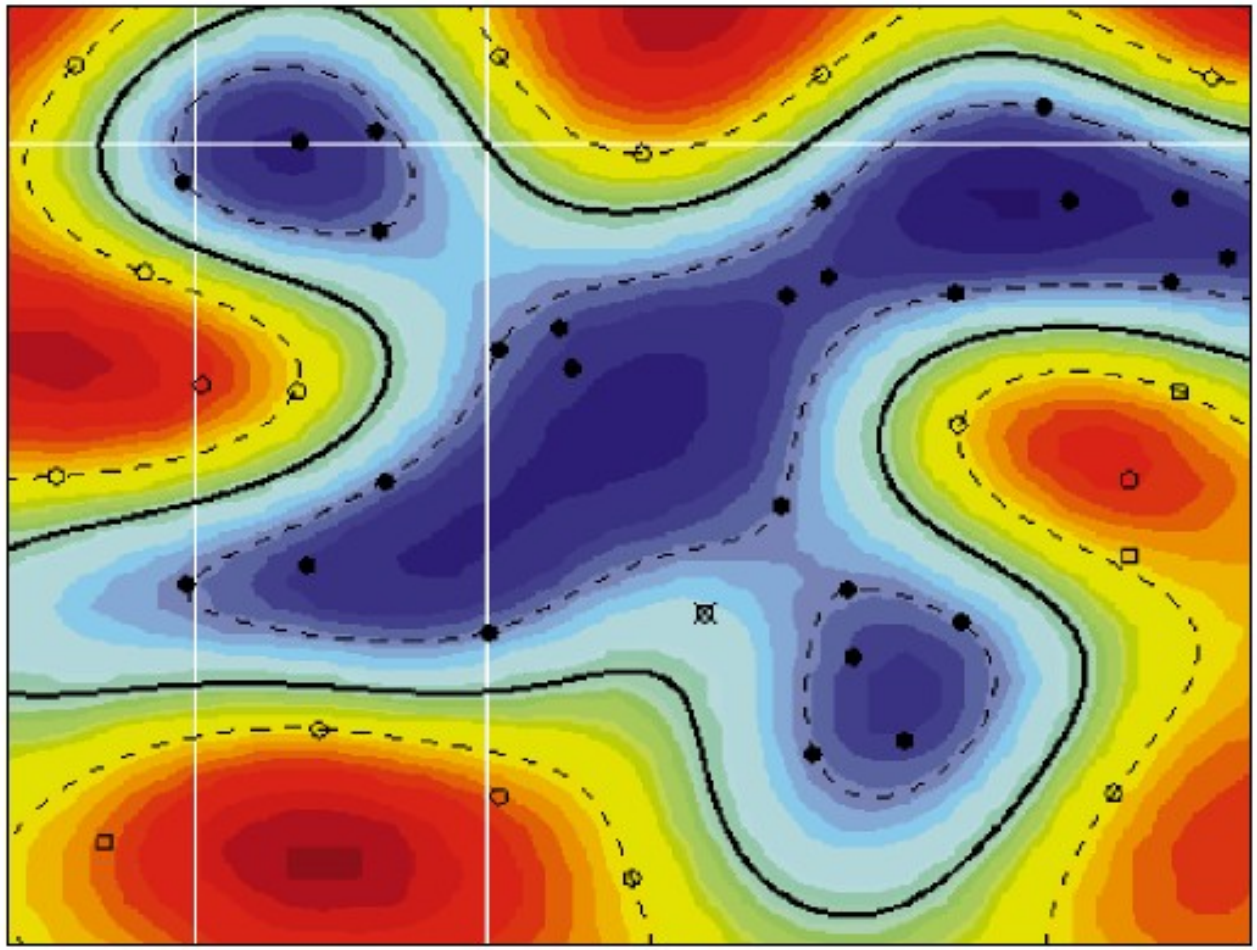
$$K(x,z) = \frac{1}{\sqrt{\|x-z\|^2 2\sigma^2 + c^2}}$$

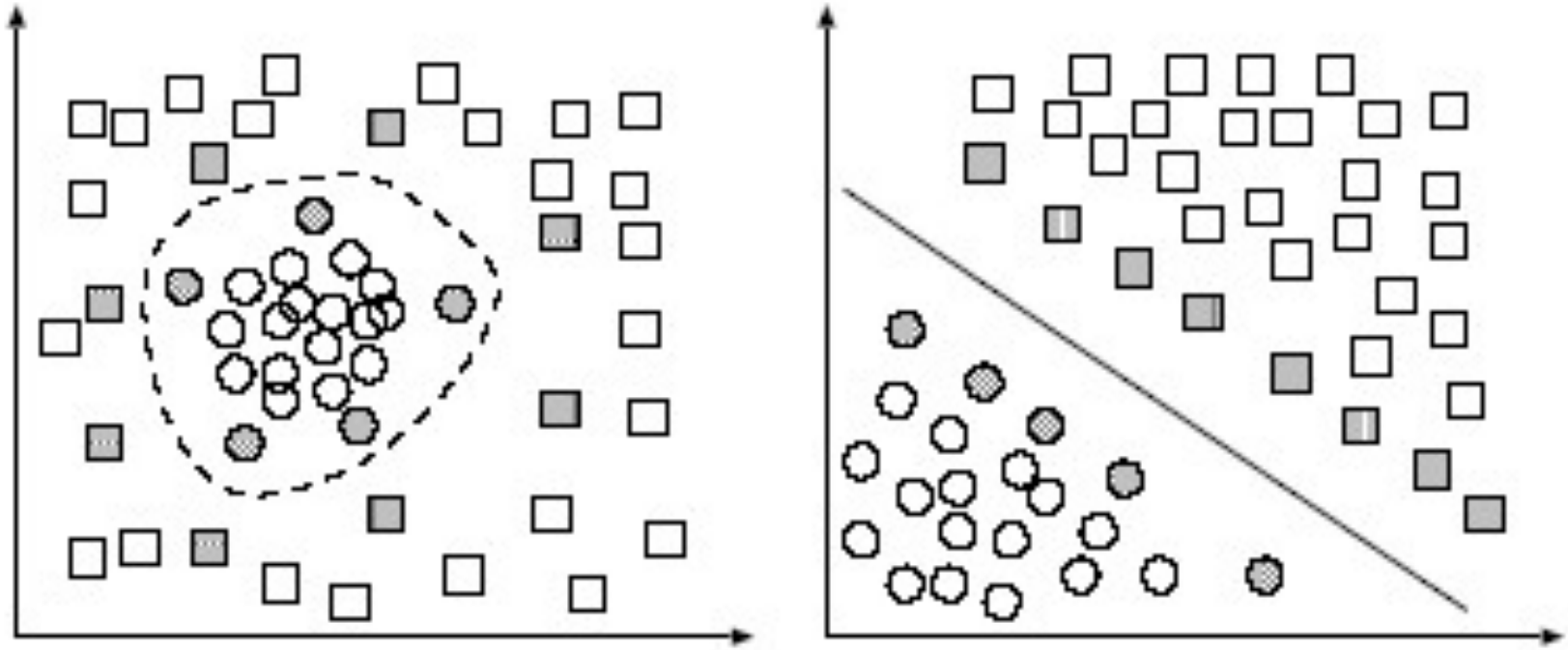
Making Kernels from Kernels

- create complex Kernels by combining simpler ones
- Closure Properties:

$$
\begin{aligned}
K(x,z) &= c \cdot K_1(x,z) \\
K(x,z) &= c + K_1(x,z) \\
K(x,z) &= K_1(x,z) + K_2(x,z) \\
K(x,z) &= K_1(x,z) \cdot K_2(x,z) \\
K(x,z) &= f(x) \cdot f(z)
\end{aligned}
$$

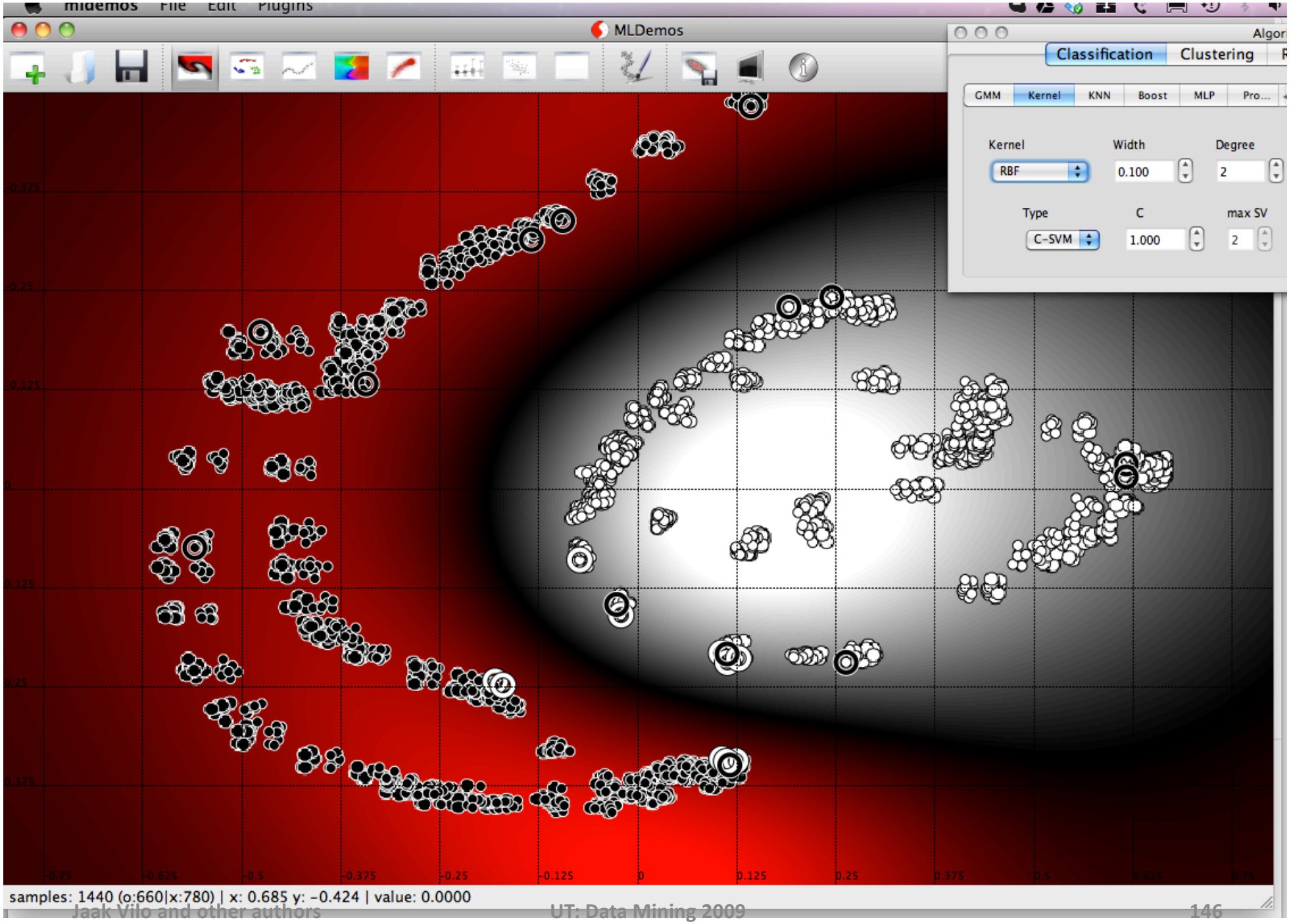
if K_1 and K_2 are kernels, $\forall f : X \to \mathbb{R}$, and $c > 0$





(a) Radial Basis Function (b) RBF mapping

Separable classification with Radial Basis kernel functions in different space. Left: original space. Right: feature space.


MLDemos
Classification
Clustering
GMM
Kernel
KNN
Boost
MLP
Kernel
Width
Degree
RBF
0.100
2
Type
C
max SV
C-SVM
1.000
2
samples: 1440 (o:660|x:780) | x: 0.685 y: -0.424 | value: 0.0000

Summary

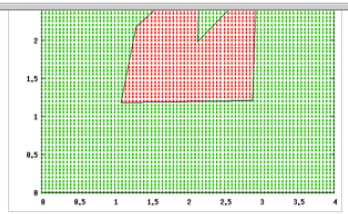
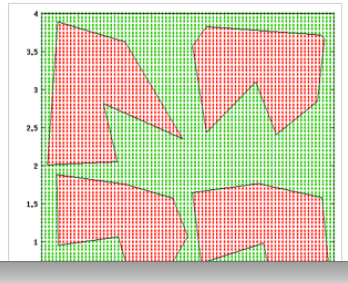
- SVM:
  - A maximal margin linear classifier.
  - A linear model, which minimizes
    Hinge loss + l_2-norm penalty
  - Kernelizable

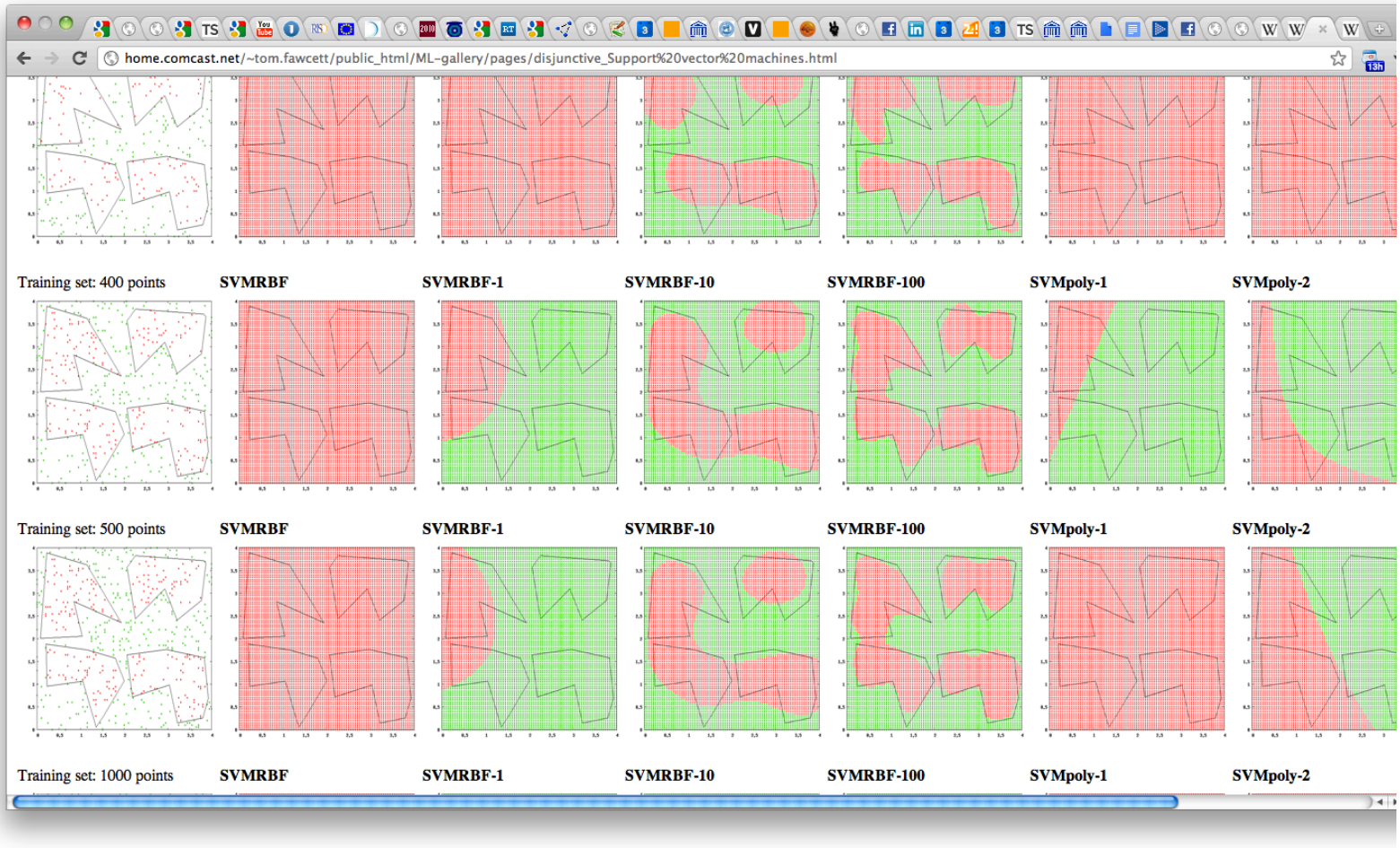
- Kernel methods:
  - An easy and elegant way of "plugging-in"
    - nonlinearity
    - different data types

ML Gallery

- http://home.comcast.net/~tom.fawcett/public_html/ML-gallery/pages/index.html
  - Now disappeared from net …

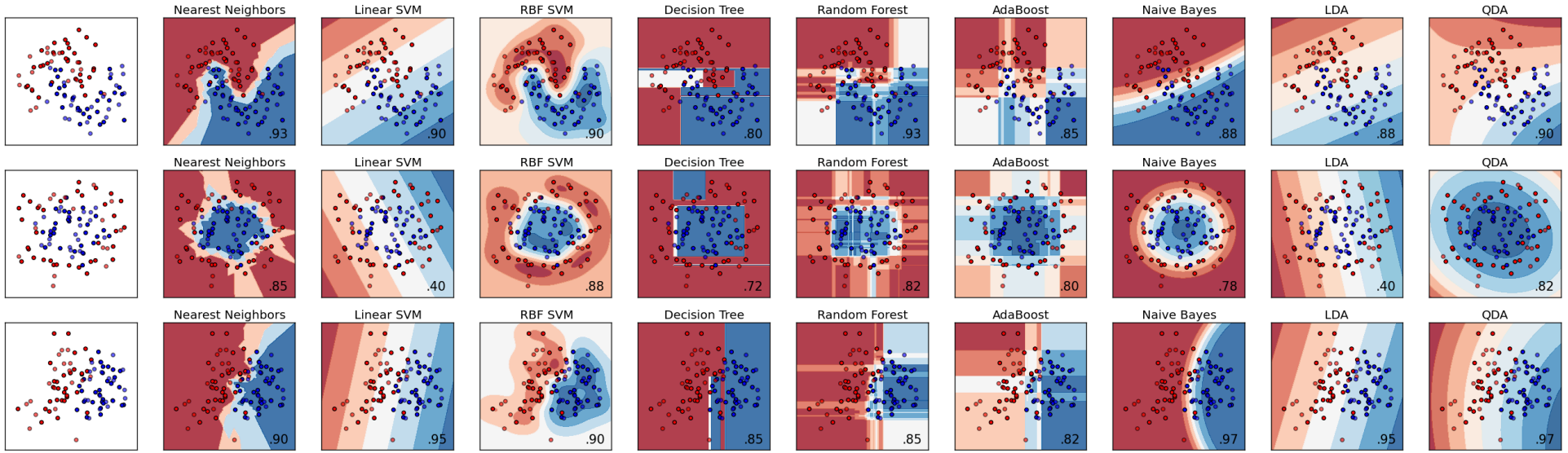
- A Collection of example data sets and illustration of different ML algorithms performance on them

X = [0, 4], Y = [0, 4], resolution 0.05 Concave polygon of 6 points Total of 6561 points (862 class 1, 5699 class 2) Skew: 0.15 / 1		Bayes	Instance-space methods	Logistic regression	Rule and tree	Support vector machines
CLOSEDCONVEX X = [0, 4], Y = [0, 4], resolution 0.05 Convex polygon of 8 points Total of 6561 points (914 class 1, 5647 class 2) Skew: 0.16 / 1		Bayes	Instance-space methods	Logistic regression	Rule and tree	Support vector machines
DISJUNCTIVE X = [0, 4], Y = [0, 4], resolution 0.05 4 disjoint concave polygons Total of 6561 points (2945 class 1, 3616 class 2) Skew: 0.81 / 1		Bayes	Instance-space methods	Logistic regression	Rule and tree	Support vector machines


home.comcast.net/~tom.fawcett/public_html/ML-gallery/pages/disjunctive_Support%20vector%20machines.html
Training set: 400 points
SVMRBF
SVMRBF-1
SVMRBF-10
SVMRBF-100
SVMpoly-1
SVMpoly-2
Training set: 500 points
SVMRBF
SVMRBF-1
SVMRBF-10
SVMRBF-100
SVMpoly-1
SVMpoly-2
Training set: 1000 points
SVMRBF
SVMRBF-1
SVMRBF-10
SVMRBF-100
SVMpoly-1
SVMpoly-2

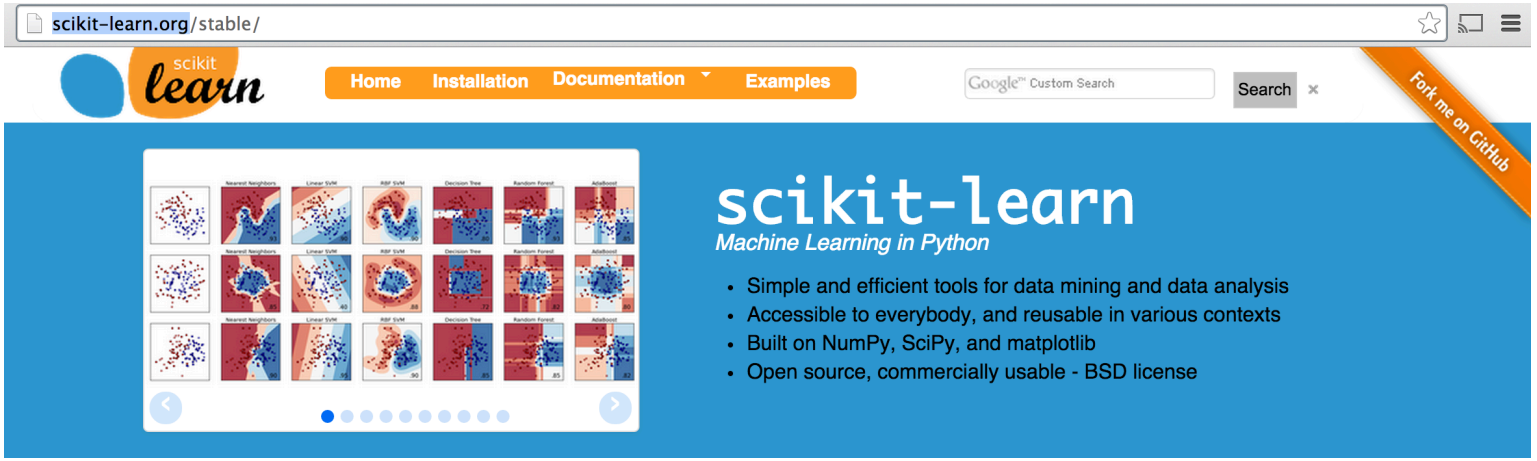

home.comcast.net/~tom.fawcett/public_html/ML-gallery/pages/disjunctive_Rule%20and%20tree.html
dtree
dtree-U
JRip
JRip-P
pyriel-1
pyriel-2
pyriel-3
ADTree
RandomTree
RandomForest
scett@gmail.com
Main index page

Scikit learn



scikit-learn

http://scikit-learn.org/



A Must software

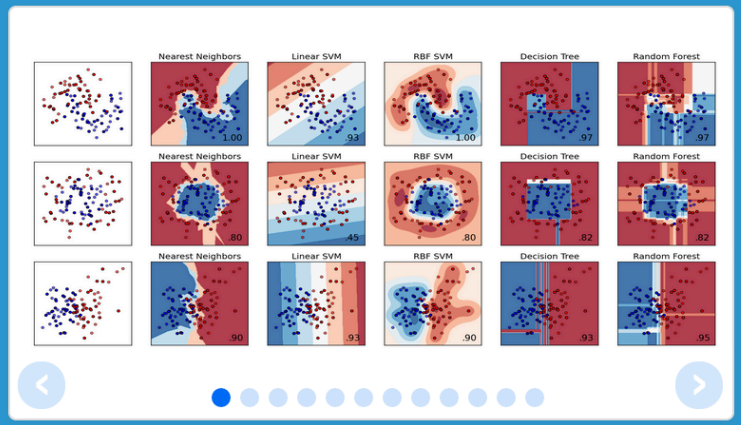
- WEKA
- Mldemos
- R
- …

scikit-learn.org/stable/

scikit learn

Home | Installation | Documentation | Examples

Google™ Custom Search | Search



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which set of categories a new observation belong to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM*, *nearest neighbors*, *random forest*, ... — *Examples*

Regression

Predicting a continuous value for a new example.

Applications: Drug response, Stock prices.
Algorithms: *SVR*, *ridge regression*, *Lasso*, ... — *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means*, *spectral clustering*, *mean-shift*, ... — *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA*, *Isomap*, *non-negative matrix factorization*. — *Examples*

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search*, *cross validation*, *metrics*. — *Examples*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing*, *feature extraction*. — *Examples*

pyml.sourceforge.net

home | tutorial | svm howto | download | developer »

PyML – machine learning in Python

PyML is an interactive object oriented framework for machine learning written in Python. PyML focuses on SVMs and other kernel methods. It is supported on Linux and Mac OS X.

News

- Added wrapper for liblinear in version 0.7.8. This allows extremely fast training of linear SVMs.

Features

- Classifiers: support vector machines, nearest neighbor, ridge regression
- Multi-class methods (one-against-rest and one-against-one)
- Feature selection (filter methods, RFE)
- Model selection
- Preprocessing and normalization
- Syntax for combining classifiers
- Classifier testing (cross-validation, error rates, ROC curves)
- Various kernels for biological sequences (several variants of the spectrum kernel, and the weighted-degree kernel).

SOURCEFORGE.NET

home | tutorial | svm howto | download | developer »

mlpy
machine learning py

Home
Download
Documentation
mlpy at SF

minepy project

News

mlpy 3.5.0 released (2012-03-12).

mlpy 3.4.0 released (2012-01-09).

mlpy 3.3.0 released (2011-12-19). From this version, mlpy for Windows is compiled with Visual Studio Express 2008 in order to avoid runtime errors.

mlpy 3.2.1 released (2011-12-9). From this version mlpy is available both for **Python >=2.6** and **Python 3.X**.

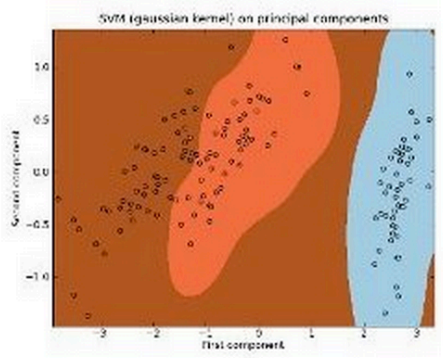
mlpy - Machine Learning Python

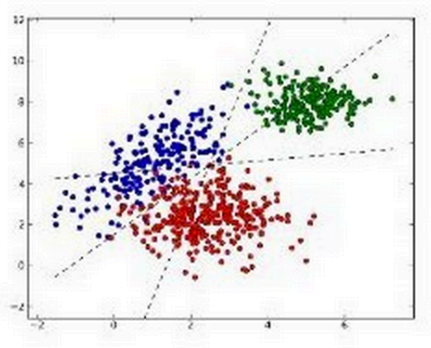
mlpy is a Python module for **Machine Learning** built on top of NumPy/SciPy and the GNU Scientific Libraries.

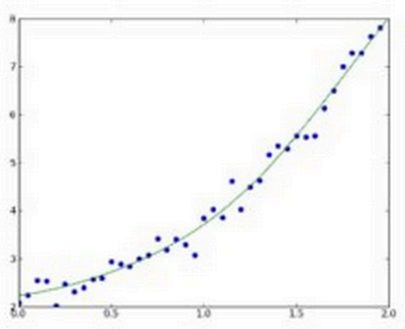
mlpy provides a wide range of state-of-the-art machine learning methods for **supervised** and **unsupervised** problems and it is aimed at finding a reasonable compromise among modularity, maintainability, reproducibility, usability and efficiency. mlpy is **multiplatform**, it works with **Python 2** and **3** and it is **Open Source**, distributed under the GNU General Public License version 3.

If you use mlpy, please cite:

D. Albanese, R. Visintainer, S. Merler, S. Riccadonna, G. Jurman, C. Furlanello. *mlpy: Machine Learning Python*, 2012. arXiv:1202.6548 [bib]







mlpy was used in the following applications.

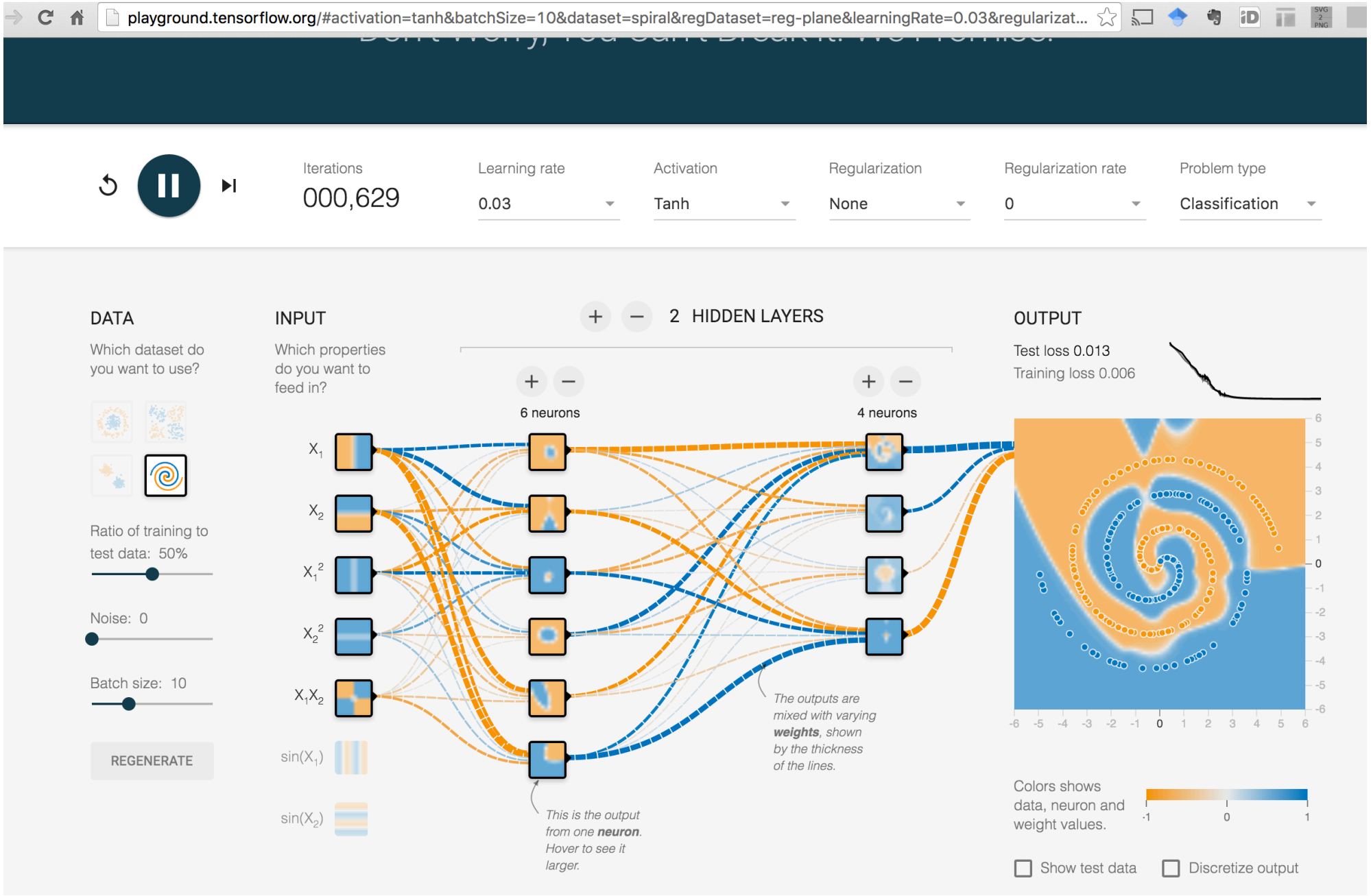
Features

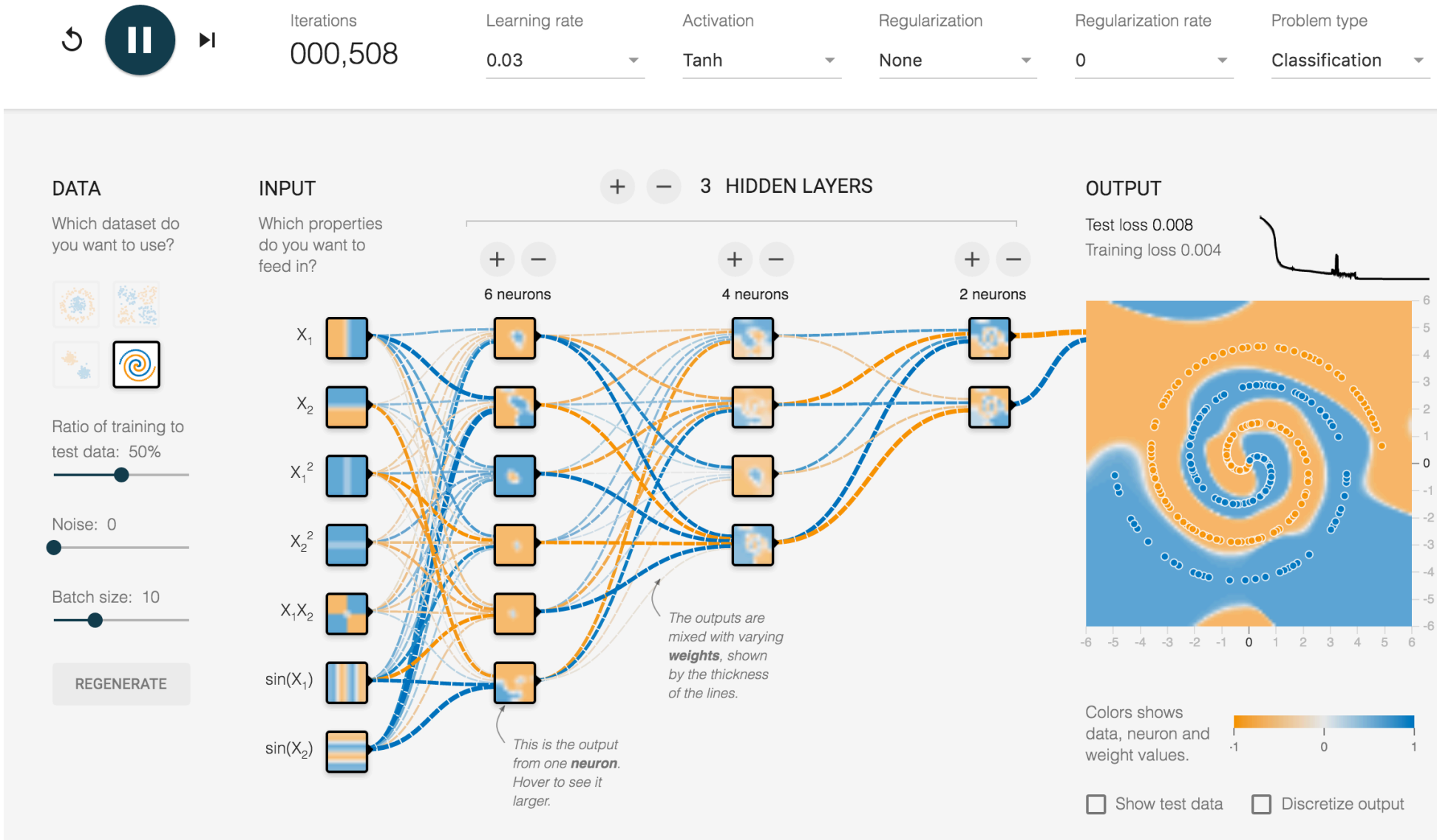
Regression: Least Squares, Ridge Regression, Last Angle Regression, Elastic Net, Kernel Ridge Regression, Support Vector Machines (SVR), Partial Least Squares (PLS)

Classification: Linear Discriminant Analysis (LDA), Basic Perceptron, Elastic Net, Logistic Regression, (Kernel) Support Vector Machines (SVM), Diagonal Linear Discriminant Analysis (DLDA), Golub Classifier, Parzen-based, (kernel) Fisher Discriminant Classifier k-Nearest-Neighbor, Iterative RELIEF, Classification Tree, Maximum Likelihood Classifier

Clustering: Hierarchical Clustering, Memory-saving Hierarchical Clustering, k-means

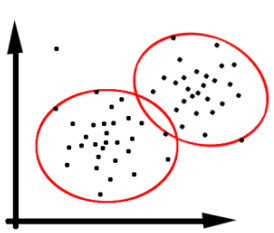
http://playground.tensorflow.org/

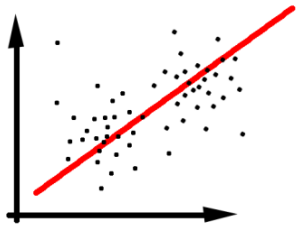


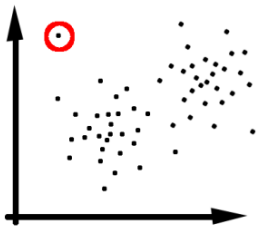

Iterations
000,508
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
INPUT
Which properties do you want to feed in?
3 HIDDEN LAYERS
6 neurons
4 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.008
Training loss 0.004
Colors shows data, neuron and weight values.
Show test data
Discretize output

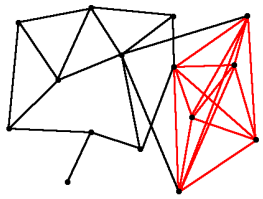
In previous episodes

Approaches to data analysis









```
AATAACGGCCCGATGAGGAAACGAACGGTCGCACT
AAGATGAGACATGTCCCGAAAGGTGCATAAGTTAT
GGACGAAAAACTTTCTTCGCCCTTTGATGTGCCCC
AGCGCGGGATGAGGATCAGCCCCCGCATTAGTTCA
ATATGCGAGCTTTCGCGCTCGGAAAGGGCAATAAA
GCGACGGCCCCGATGAGGGGTGTTACTAGATTGGA
TGGGTGGTTCAGATCTCGGCTTACCCCCTTTATCA
ACCCTGCTACAGACTCGTTGAGAATGCTACGGATC
```

- **The general principle is the same, though:**
  1. Define a set of patterns of interest
  2. Define a measure of goodness for the patterns
  3. Find the best pattern in the data

The "No Free Lunch" Theorem

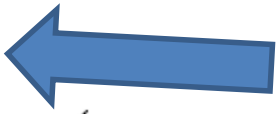
Learning purely from data is, in general, impossible

X	Y	Output
0	0	False
0	1	True
1	0	True
1	1	?

Statistical Learning Theory

- What is learning and how to analyze it?
  - There are various ways to answer this question. We'll just consider the most popular one.

- Let $(x, y) \sim D$ be the distribution of data.
- We observe an i.i.d. sample:



Inductive bias

$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

- We produce a classifier:

$$g_S(x)$$

The "No Free Lunch" Theorem

Learning purely from data is, in general, impossible

- Is it good or bad?
  –
- What should we do to enable learning?
  –

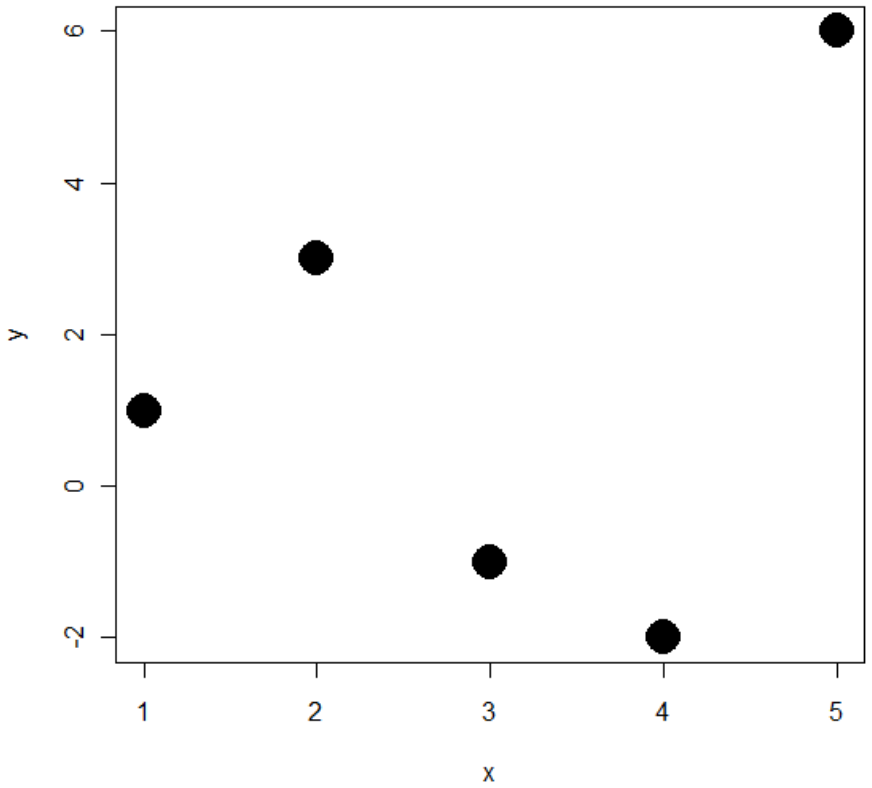
The "No Free Lunch" Theorem

Learning purely from data is, in general, impossible

- Is it good or bad?
  - Good for cryptographers, bad for data miners
- What should we do to enable learning?
  - Introduce **assumptions about data** ("inductive bias"):
    1. **How does existing data relate to the future data?**
    2. **What is the system we are learning?**

The "No Free Lunch" Theorem

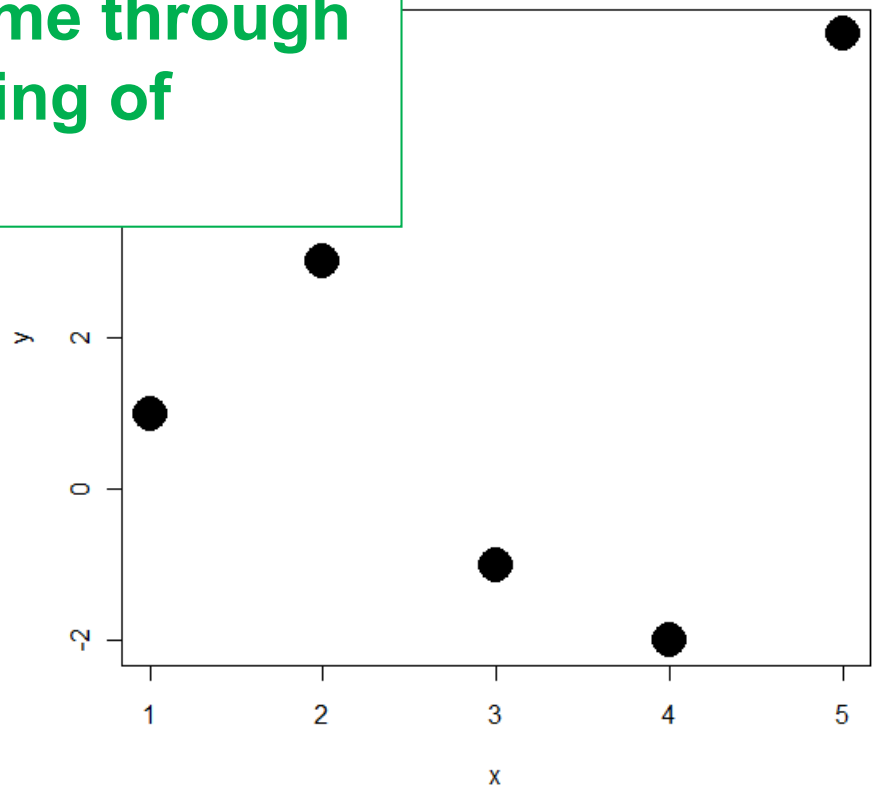
Learning purely from data is, in general, impossible



The "No Free Lunch" Theorem

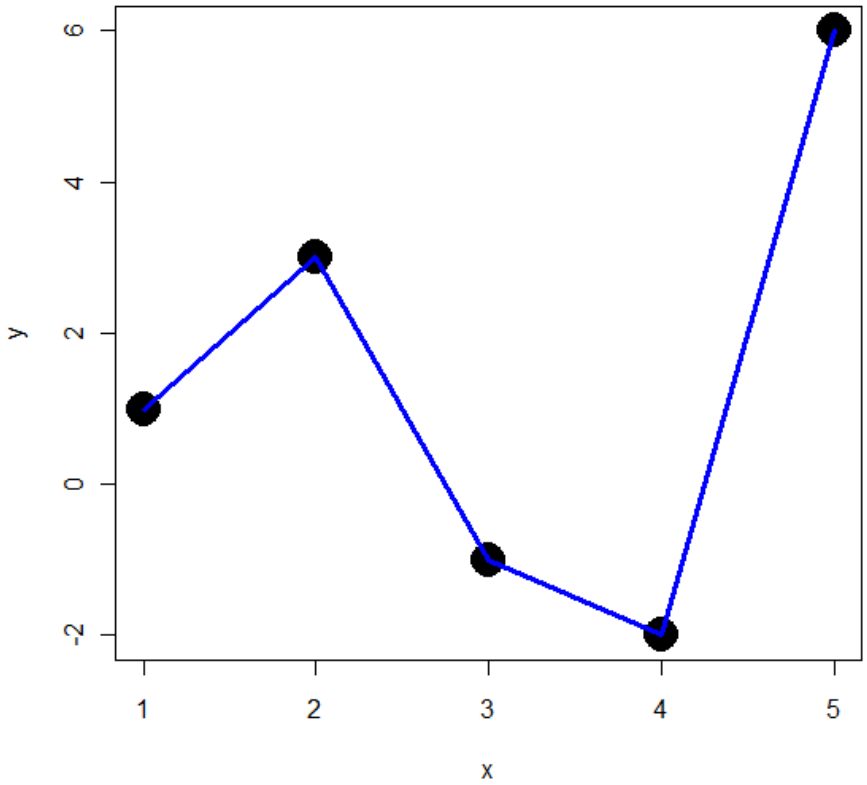
Learning purely from data is, in general, impossible

Rule 1: Generalization will only come through understanding of similarity!



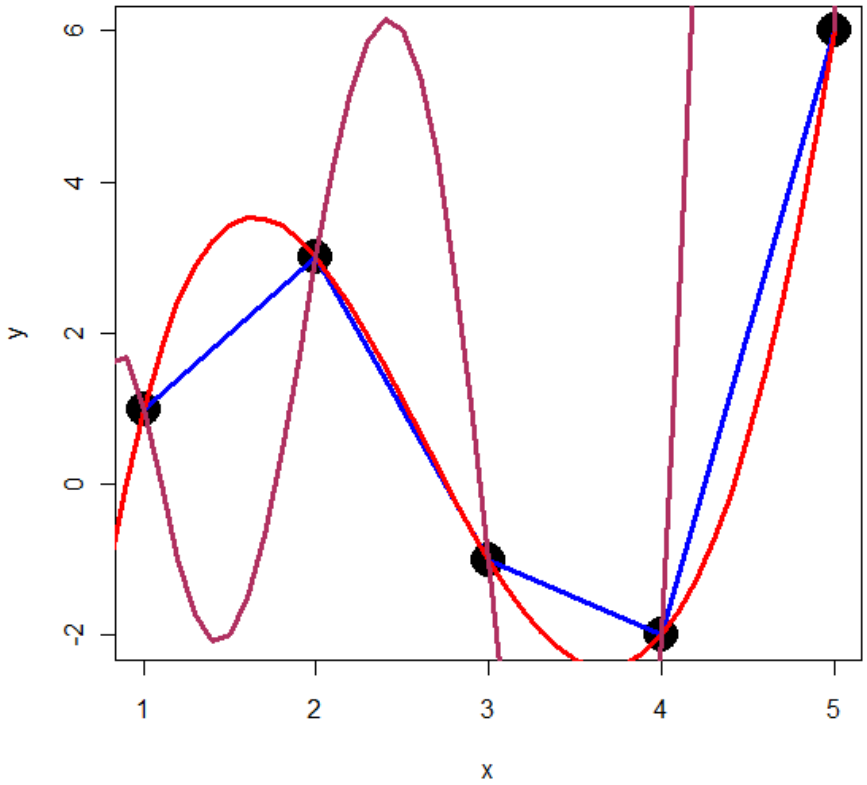
The "No Free Lunch" Theorem

Learning purely from data is, in general, impossible



The "No Free Lunch" Theorem

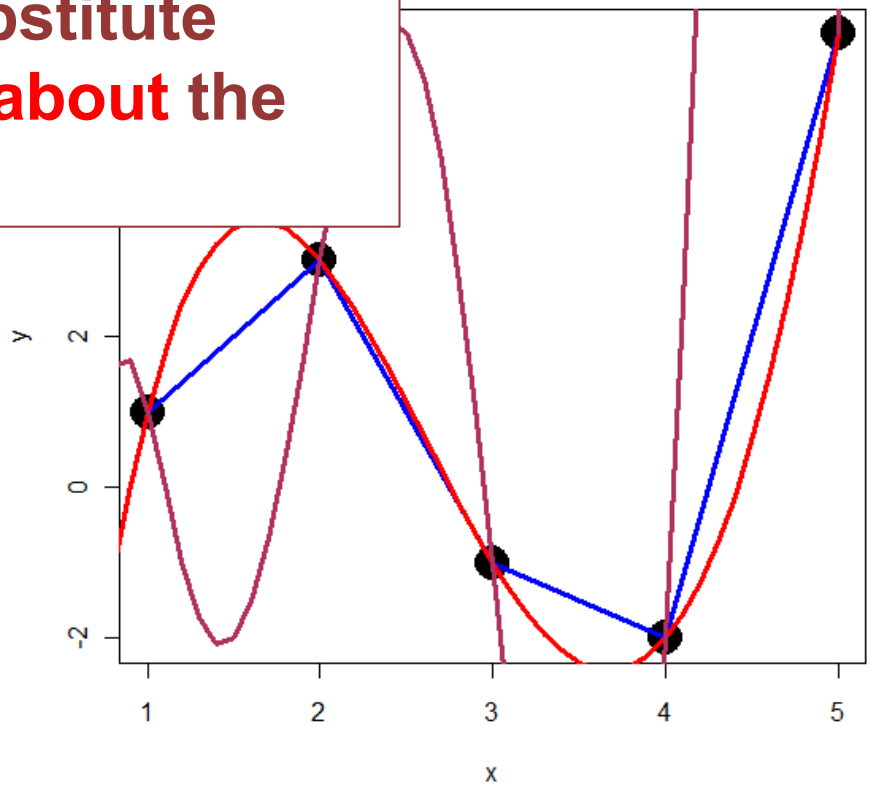
Learning purely from data is, in general, impossible



The "No Free Lunch" Theorem

Learning purely from data is, in general, impossible

Rule 2: Data can only partially substitute knowledge about the system!

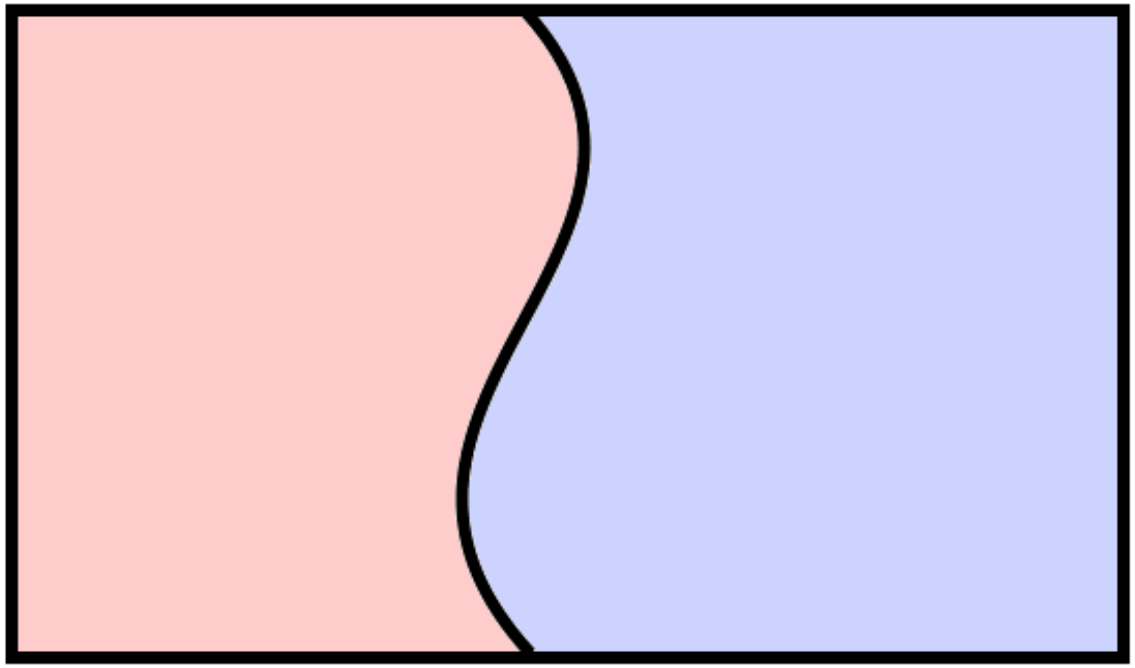


Statistical Learning Theory

- What is learning and how to analyze it?
  - There are various ways to answer this question. We'll just consider the most popular one.

Statistical Learning Theory

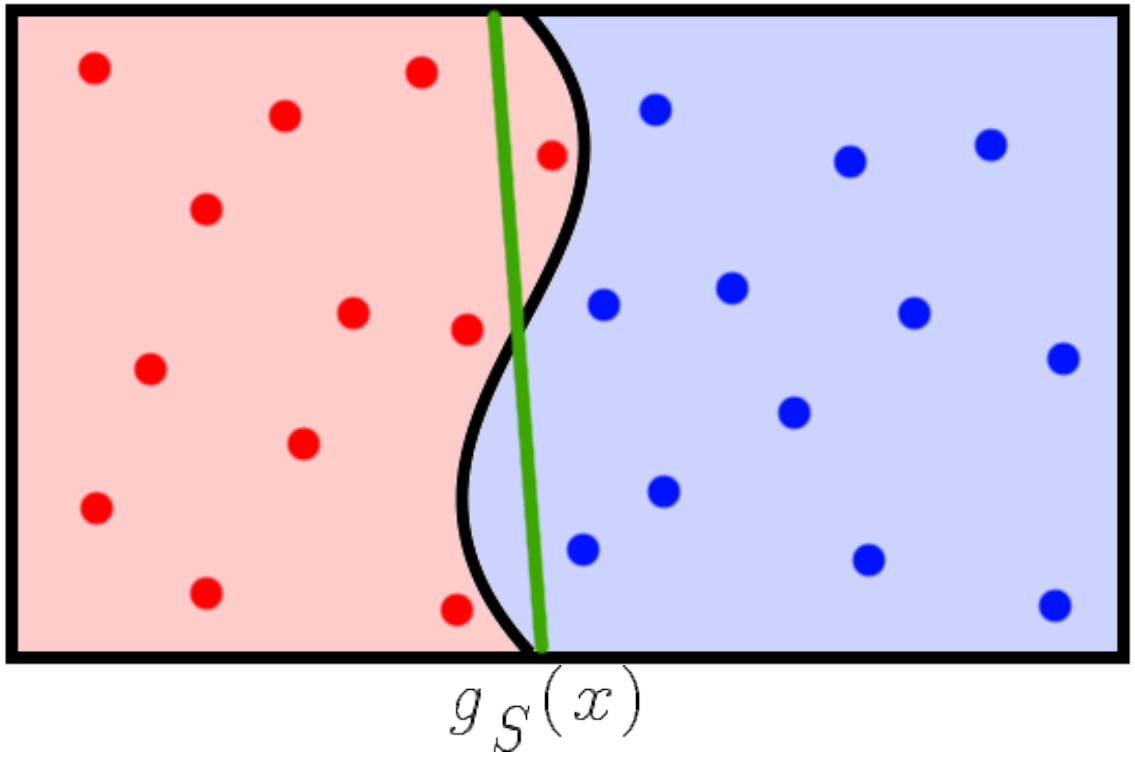
$$(x, y) \sim D$$



Distribution of data (x – coords, y – color)

Statistical Learning Theory

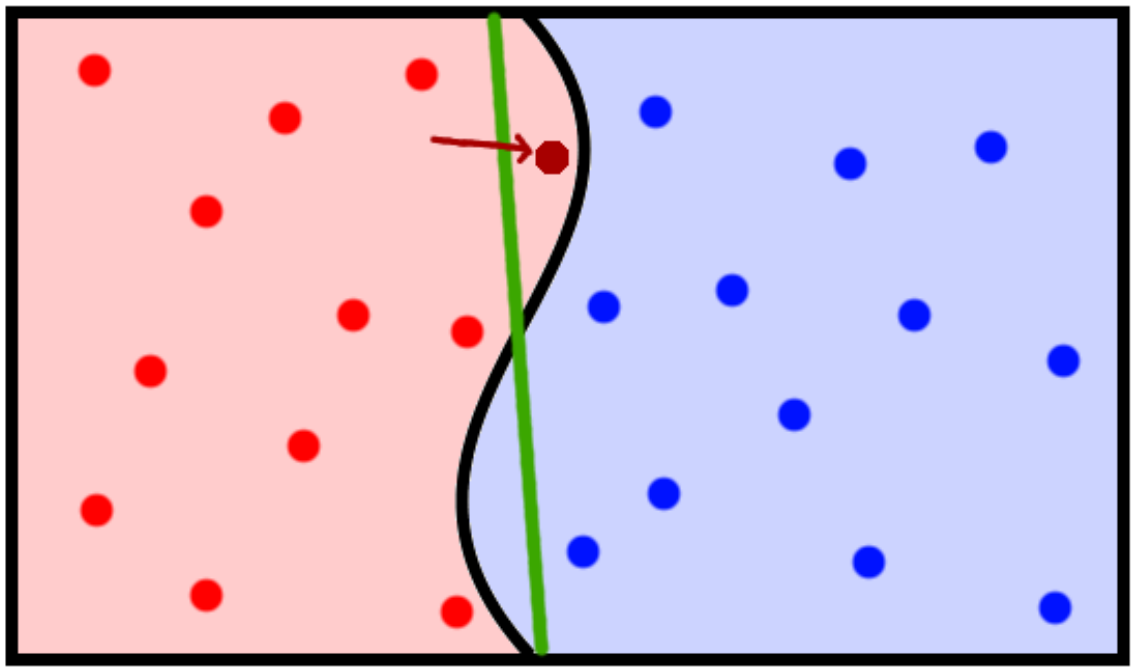
$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$



Sample S and trained classifier g

Statistical Learning Theory

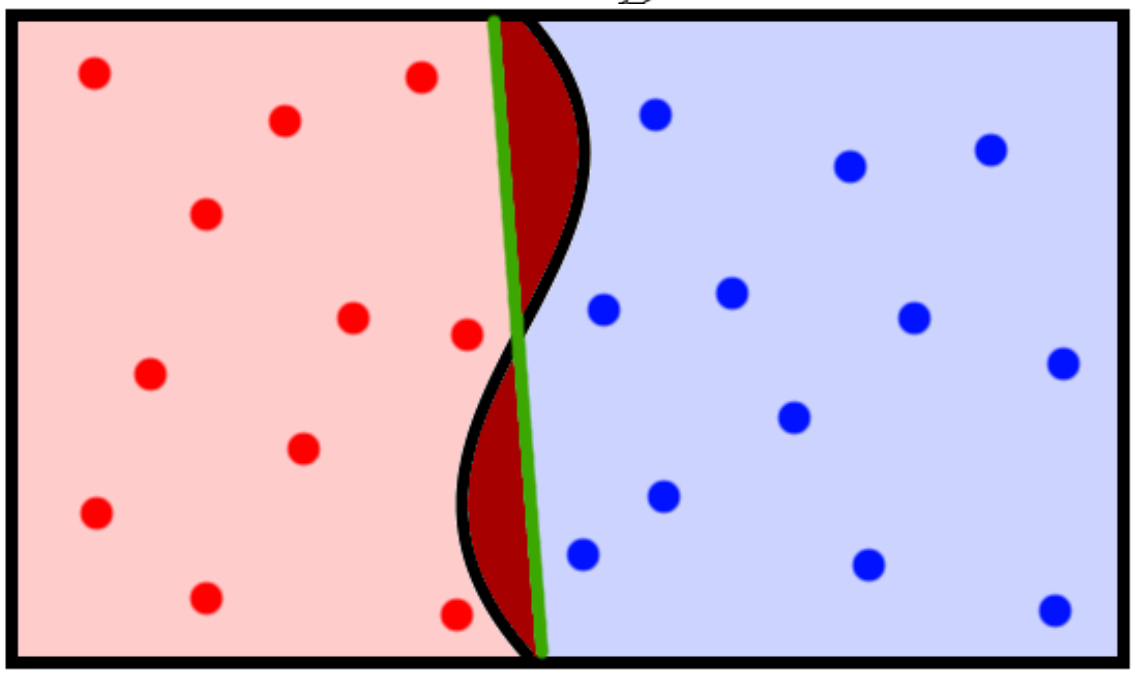
$$\text{training error} = \frac{1}{n} \sum_i [g_S(x_i) \neq y_i]$$



Training error

Statistical Learning Theory

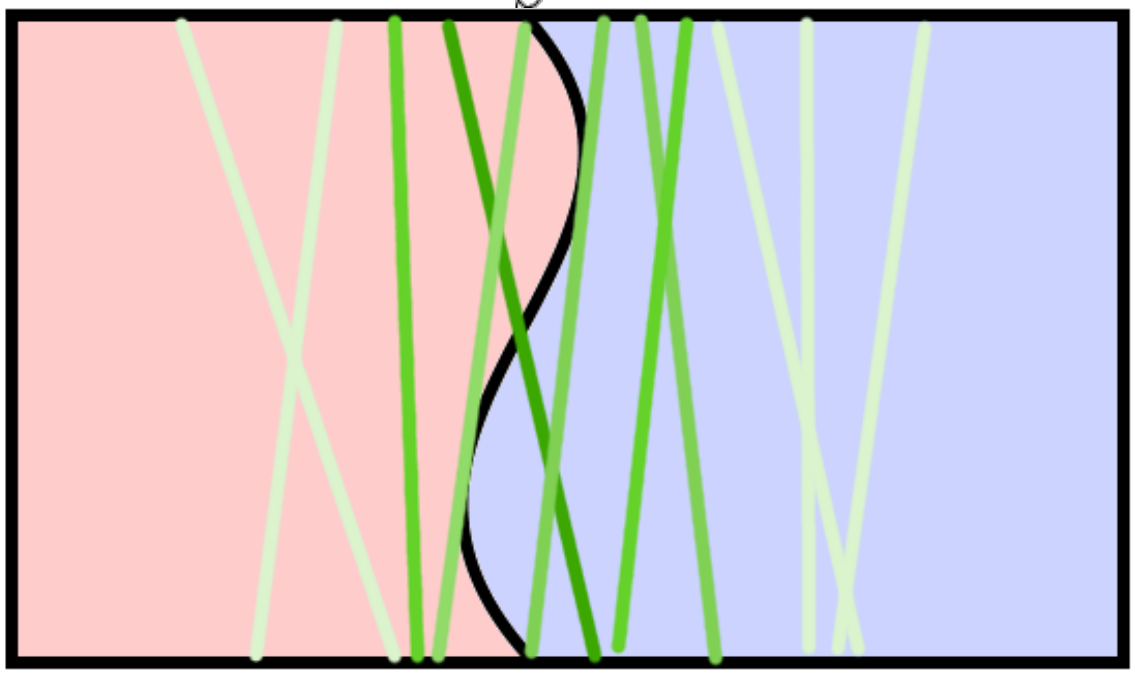
$$\text{error} = \Pr(g_S(x) \neq y) = \int_D [g_S(x) \neq y] f(x,y) dx dy$$



Generalization error

Statistical Learning Theory

$$E(\text{error}) = \int_S \text{error}(g_S) dS$$



Expected generalization error

Statistical Learning Theory

- Questions:
  - What is the generalization error of our classifier?
    - How to estimate it?
    - How to find a classifier with low generalization error?
  - What is the expected generalization error of our method?
    - How to estimate it?
    - What methods have low expected generalization error?
    - What methods are asymptotically optimal (consistent)?
  - When is learning computationally tractable?

Statistical Learning Theory

- Some answers: for linear classifiers
  - Finding a linear classifier with a small training error is a good idea

$$E(\text{error}) - \text{optimum} \leq 16\sqrt{\frac{(d+1)\ln n + 4}{2n}}$$

(however, finding such a classifier is NP-complete, hence an alternative method must be used)

Statistical Learning Theory

- Some answers: in general
  - Small training error → small generalization error.
    - But only if you search a limited space of classifiers.
    - The more data you have, the larger space of classifiers you can afford.

Overfitting

- Why limited space?
  - Suppose your hypothesis space is just one classifier:

    f(x) = if [x > 3] then 1 else 0
  - You pick first five training instances:

    (1 → 0), (2 → 0), (4 → 1), (6 → 1), (-1 → 0)
  - How surprised are you? How can you interpret it?

Overfitting

- Why limited space?
  - Suppose your hypothesis space is just one classifier:

    f(x) = if [x > 3] then 1 else 0
  - You pick first five training instances:

    (1 → 0), (2 → 0), (4 → 1), (6 → 1), (-1 → 0)
  - How surprised are you? How can you interpret it?
  - What if you had 100000 classifiers to start with and one of them matched? Would you be surprised?

Overfitting

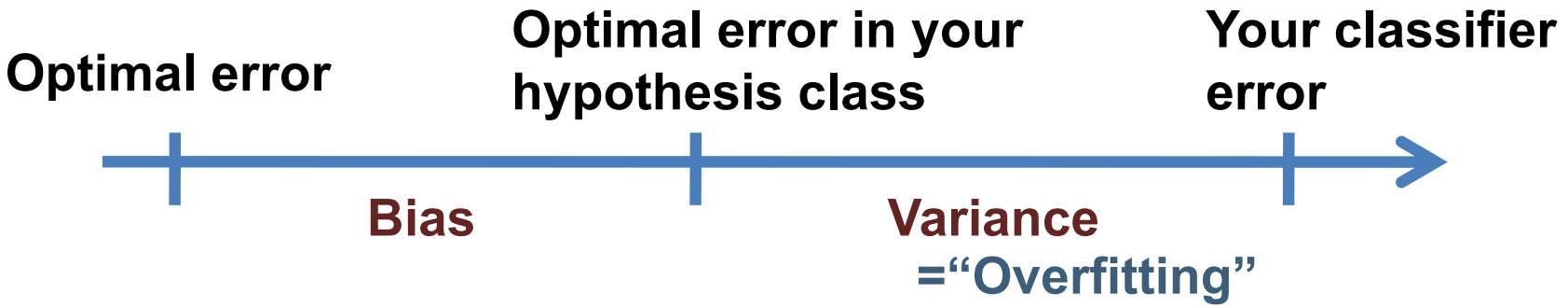
- Why limited space?
  - Suppose your hypothesis space is just one classifier:
    f(x) = if [x > 3] then 1 else 0
  - You pick first five training instances:
    (1 → 0), (2 → 0), (4 → 1), (6 → 1), (-1 → 0)
  - How surprised are you? How can you interpret it?
  - What if you had 100000 classifiers to start with and one of them matched? Would you be surprised?

Large hypothesis space ➔ Small training error becomes a matter of chance ➔ Overfitting

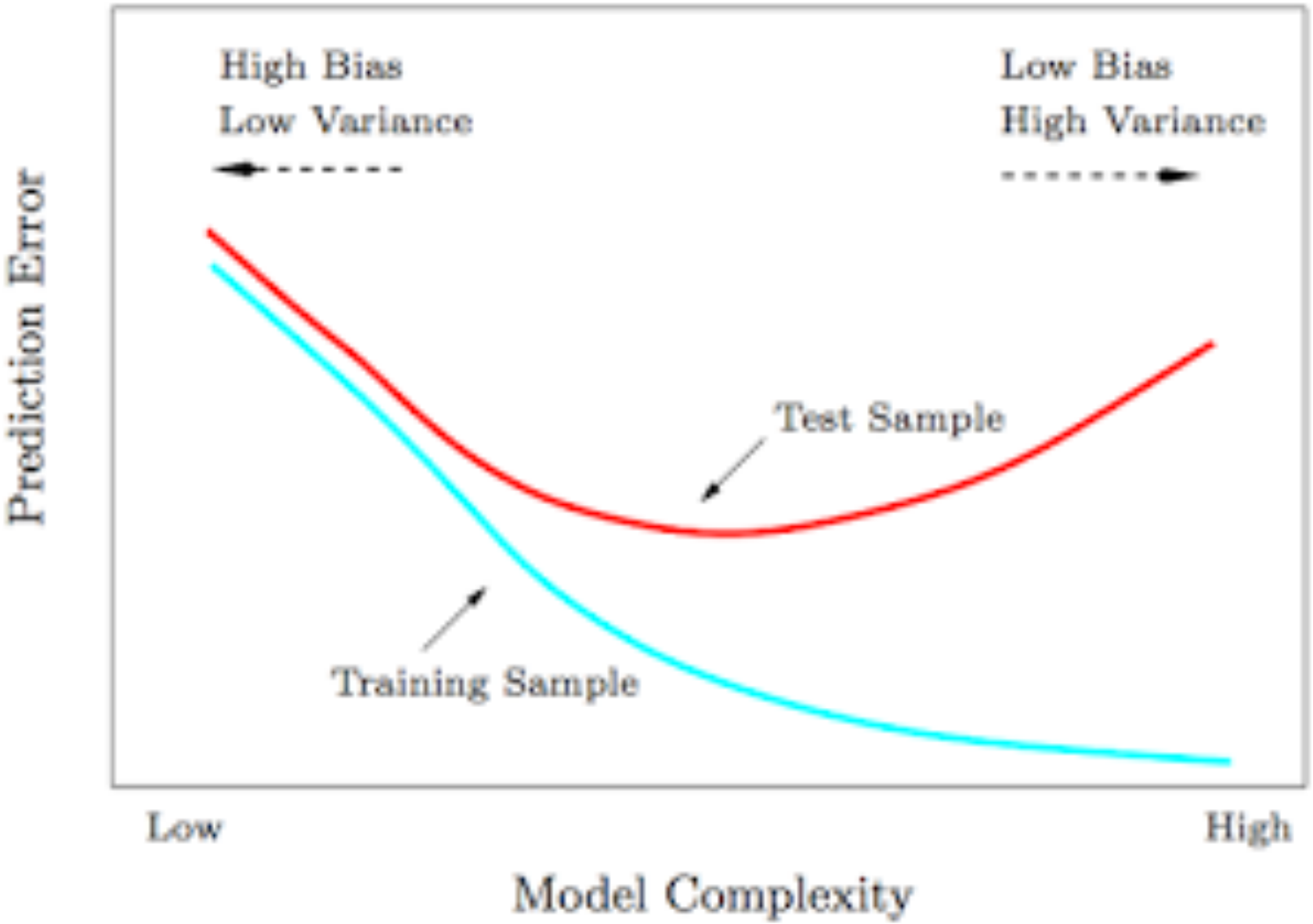
Bias-variance dilemma

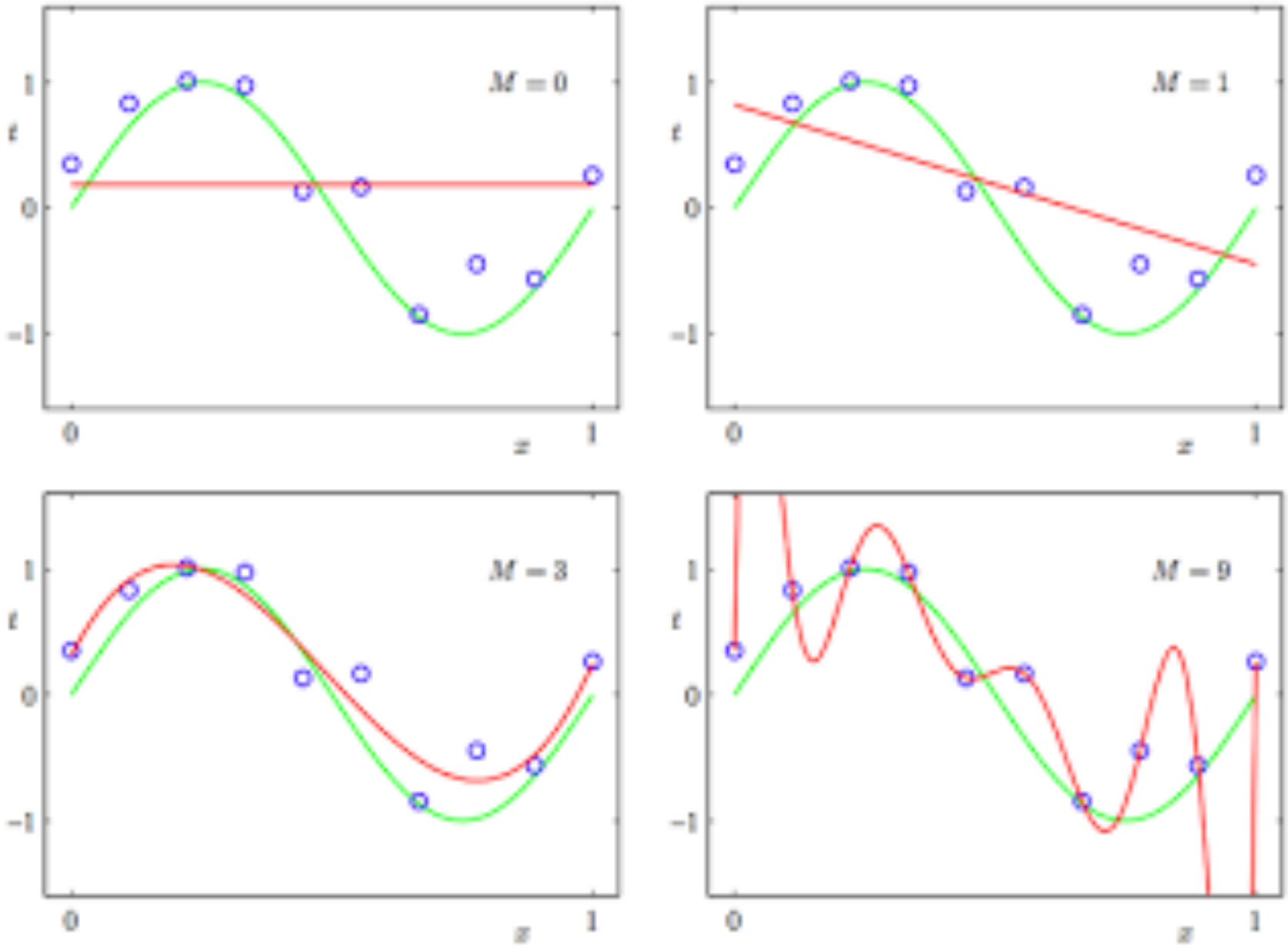
- So what if the data is scarce?

  - No free lunch
  - Bias-variance tradeoff:



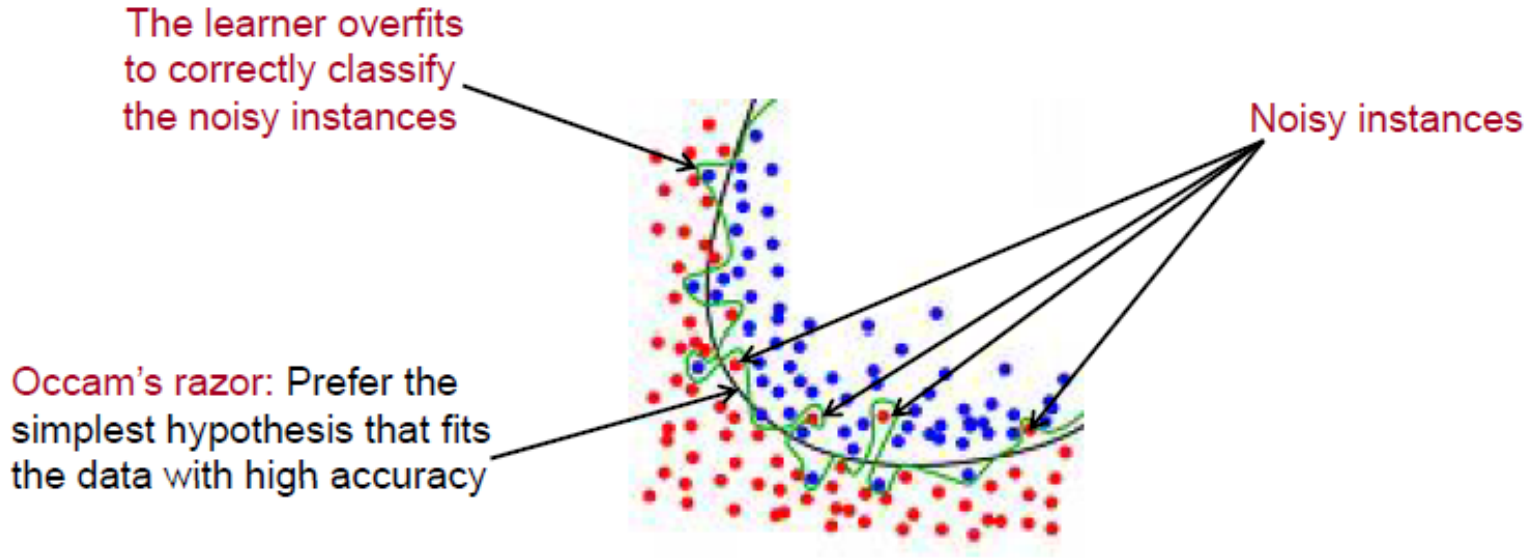
  - The only way out is to **introduce a strong yet "correct"** bias (or, well, to get **more data**).


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test Sample
Training Sample
Low
High
Model Complexity


M = 0
M = 1
M = 3
M = 9
t
x
0
1
−1

Why Does my Method Overfit ?

- **In domains with noise or uncertainty** the system may try to decrease the training error by completely fitting all the training examples



Slide from: Pavan J Joshi

Summary

- Learning can be approached formally
- Learning is feasible in many cases
  - But you pay with data or prior knowle

Summary

- Learning can be approached formally
- Learning is feasible in many cases
  - But you pay with data or prior knowle

- You have to be careful with complex models
  - Beware overfitting
  - If data is scarce – use simple models: they are not optimal, but at least you can fit them from data!

Using complex models with scarce data is like throwing data away.

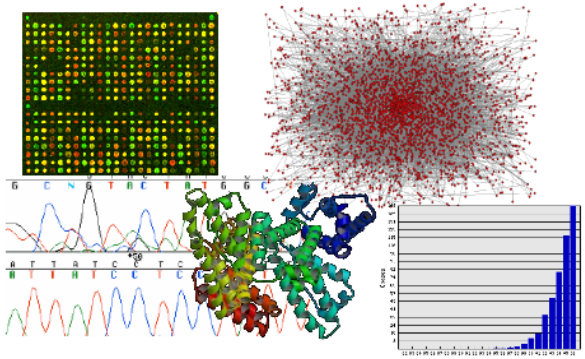
Next

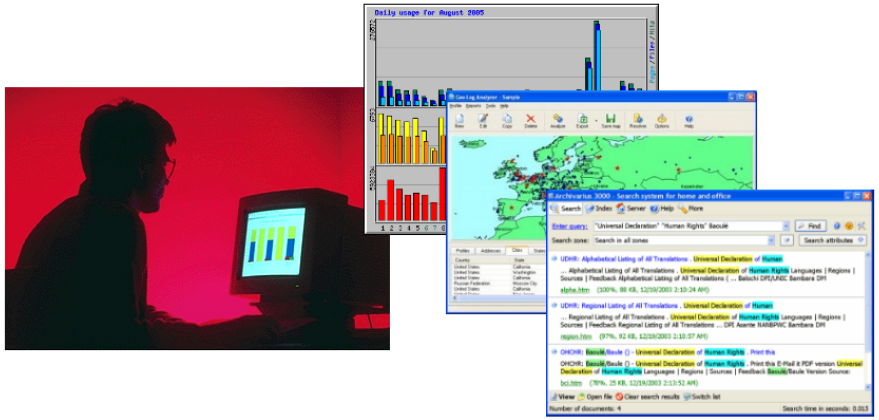
- **"Machine learning"**
  - Terminology, foundations, general framework.
- **Supervised machine learning**
  - Basic ideas, algorithms & toy examples.
- **Statistical challenges**
  - Learning theory, bias-variance, consistency…
- **State of the art techniques**
  - SVM, kernel methods, graphical models, latent variable models, boosting, bagging, LASSO, on-line learning, deep learning, reinforcement learning, …

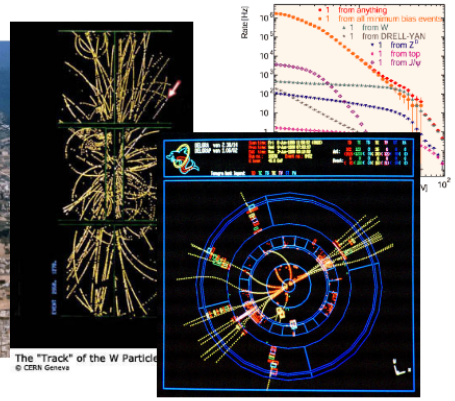
Summary summary

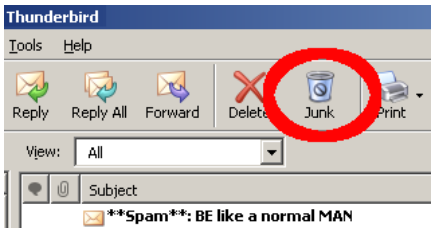
- **"Machine learning"**
  - Terminology, foundations, general framework.
- **Supervised machine learning**
  - Basic ideas, algorithms & toy examples.
- **Statistical challenges**
  - Learning theory, bias-variance, consistency,…
- **State of the art techniques**
  - SVM, kernel methods.

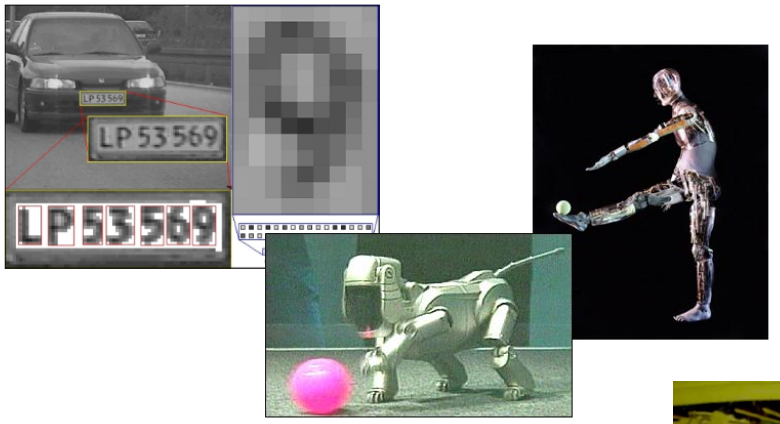
Machine learning is important











Questions?