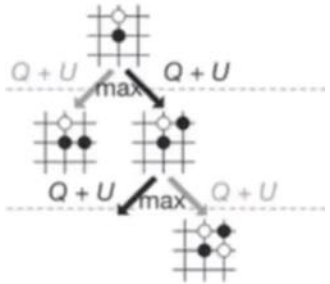
Einführung in die Künstliche Intelligenz

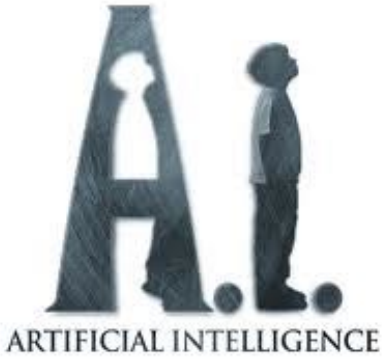
Paul Lukowicz,

Teil 3: Statistische KI und Lernen ?

Themen der Vorlesung

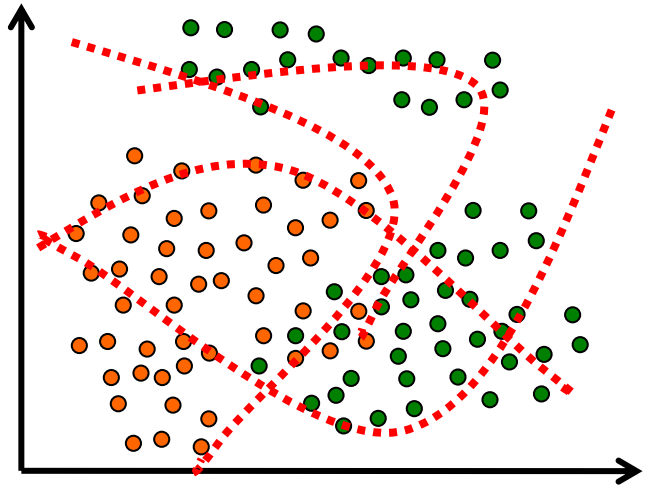
highly efficient complex (1) search and (2) knowledge representation





(3) learning: statistical analysis and optimization

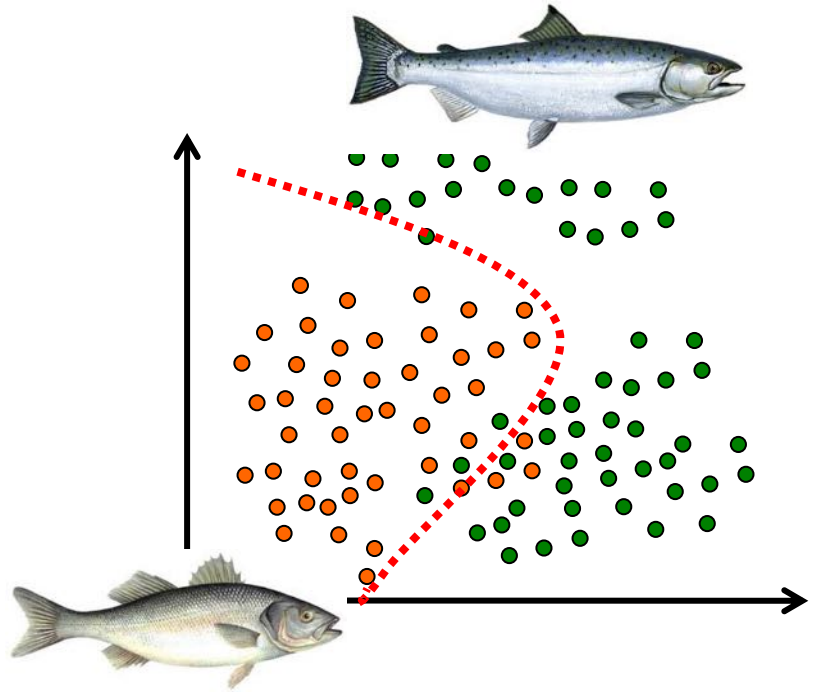
How it is done

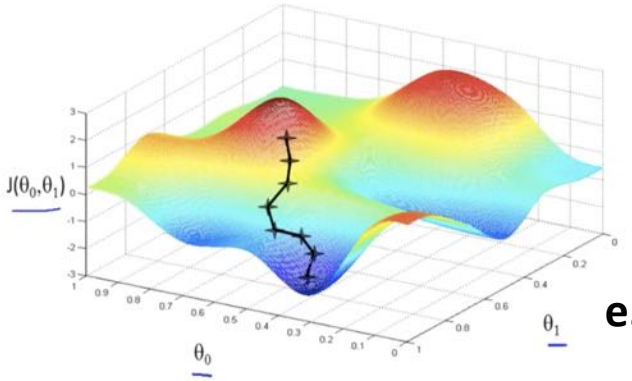


1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

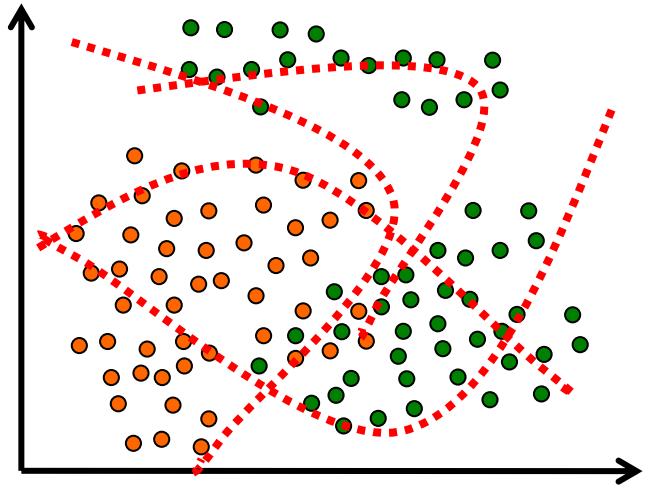
3. use an appropriate optimization techniques to find parameter values that minimize the error function





e.g. gradient descent

How it is done

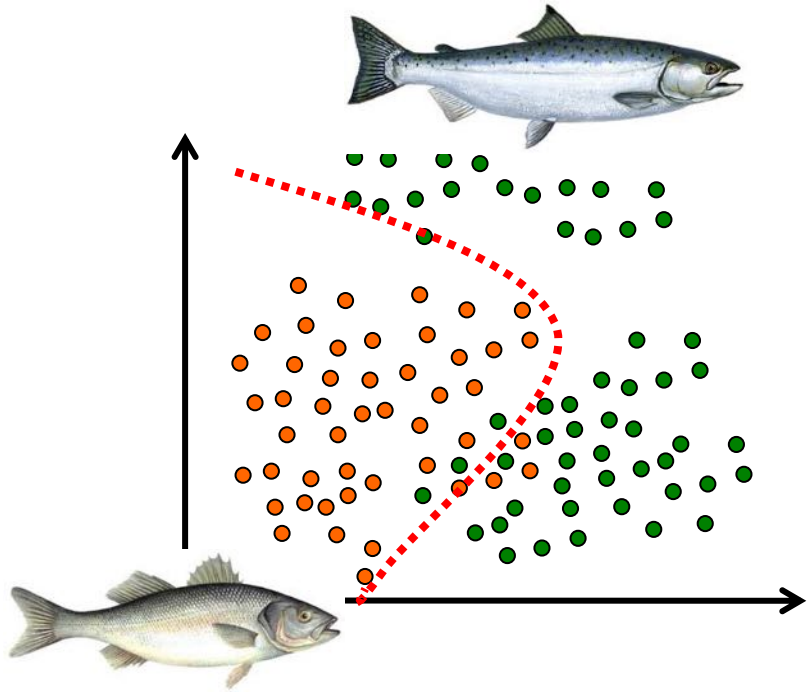


Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function

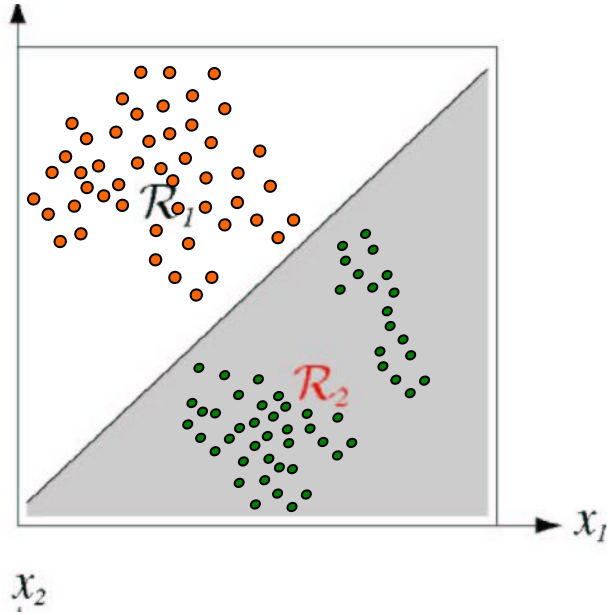


Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



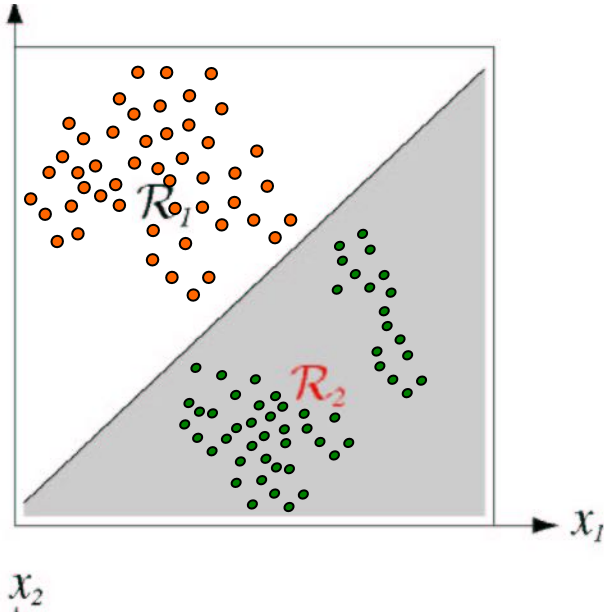
$$g(\vec{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \vec{w}^t \vec{x} + w_0 = 0$$

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



$$g(\vec{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \vec{w}^t \vec{x} + w_0 = 0$$

The error functon J($\vec{w}$) is essentially the sum number of missclasified examples

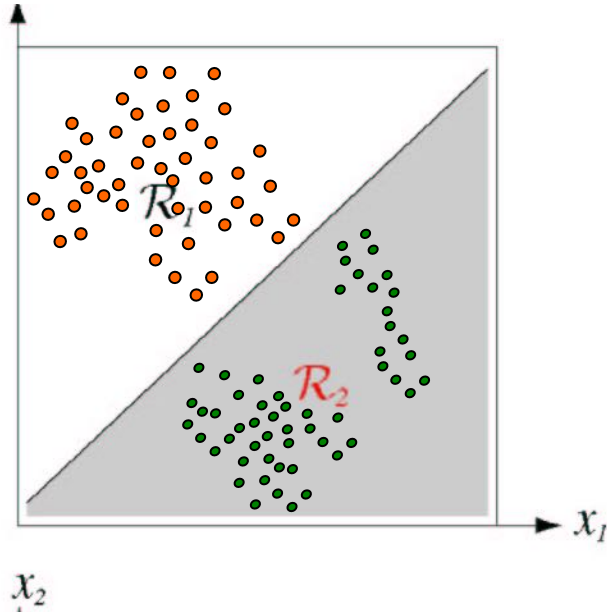
Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

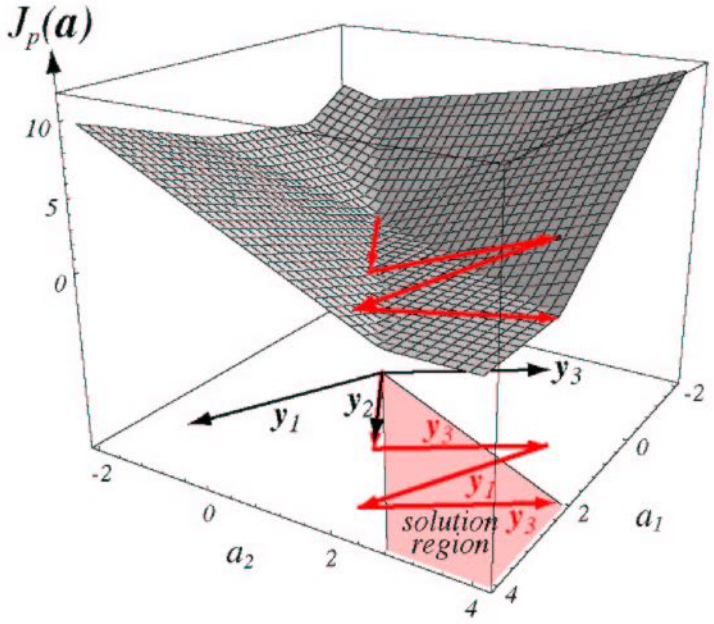
3. use an appropriate optimization techniques to find parameter values that minimize the error function

$$J(\vec{a}) = \sum_{\vec{y} \in Y'} -(\vec{a}^t \vec{y})$$

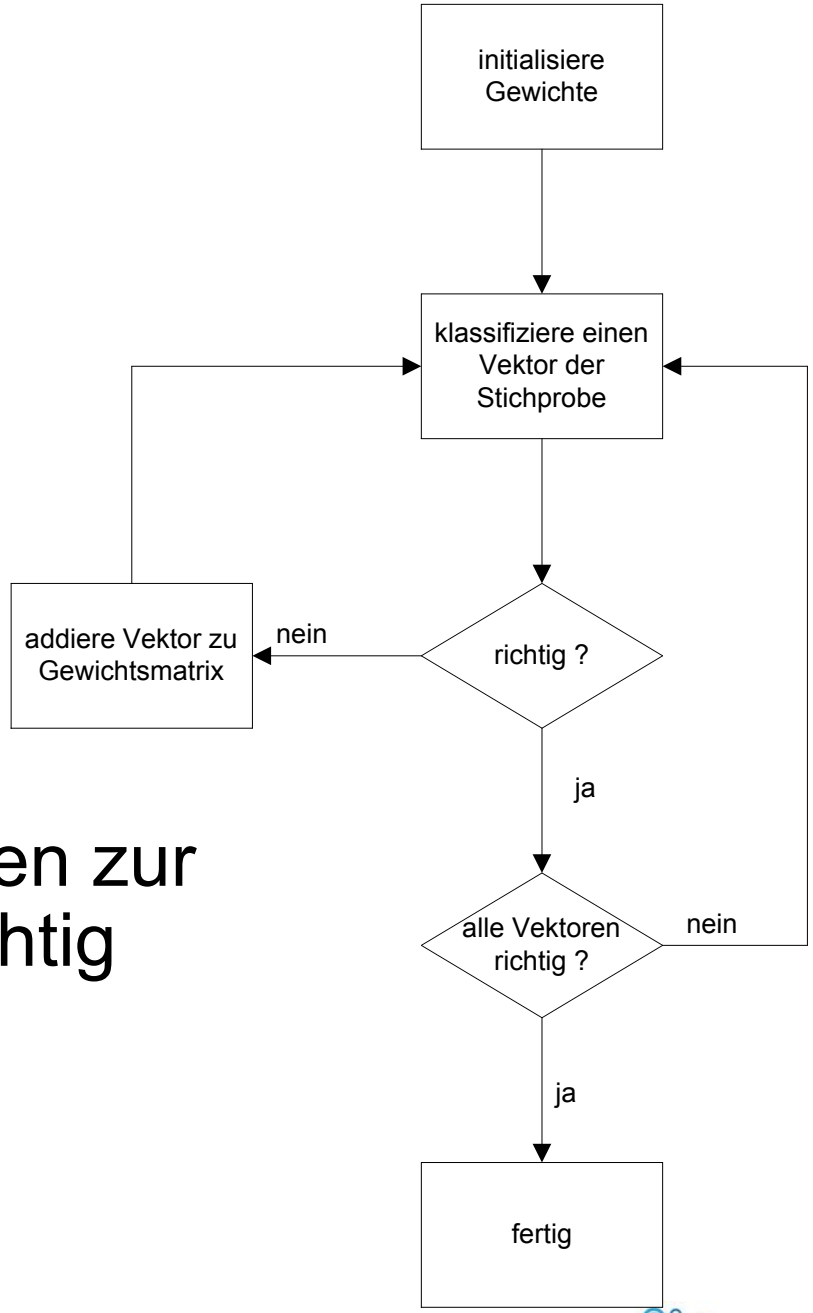


$$\nabla J(\vec{a}) = \sum_{\vec{y} \in Y'} (-\vec{y})$$

$$\vec{a}(k+1) = \vec{a}(k) + \eta(k) \sum_{\vec{y} \in Y'} \vec{y}$$



Lernverfahren



- Es werden solange fehl-klassifizierte Vektoren zur Gewichtsmatrix addiert, bis alle Vektoren richtig Klassifiziert sind

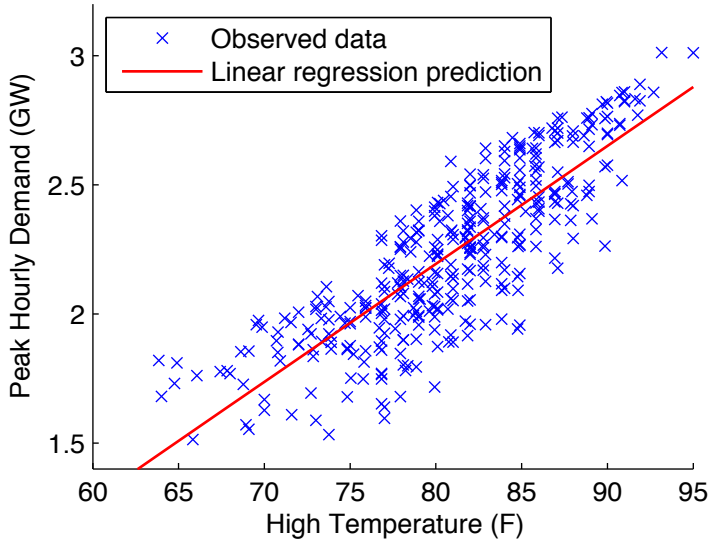
Learning in Linear Regression

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



$$\text{predicted peak demand} = \theta_1 \cdot (\text{high temperature}) + \theta_2$$

Parameters of model: $\theta_1, \theta_2 \in \mathbb{R}$

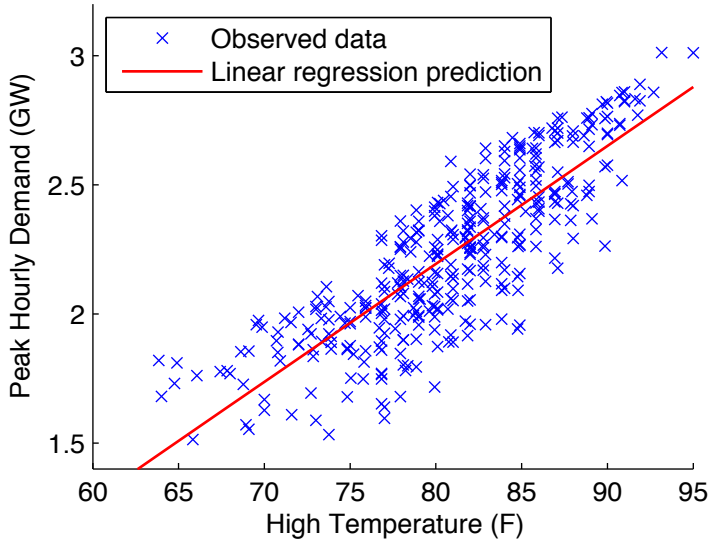
Learning in Linear Regression

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



Want a model that performs "well" on the data we have

$$\text{I.e.,} \quad \hat{y}_i \approx y_i, \quad \forall i$$

We measure "closeness" of $\hat{y}_i$ and y_i using *loss function*

$$\ell(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2 \quad \ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$$

Want to find model parameters such that minimize sum of costs over all input/output pairs

$$J(\theta) = \sum_{i=1}^{m} \ell(\hat{y}_i, y_i) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2$$

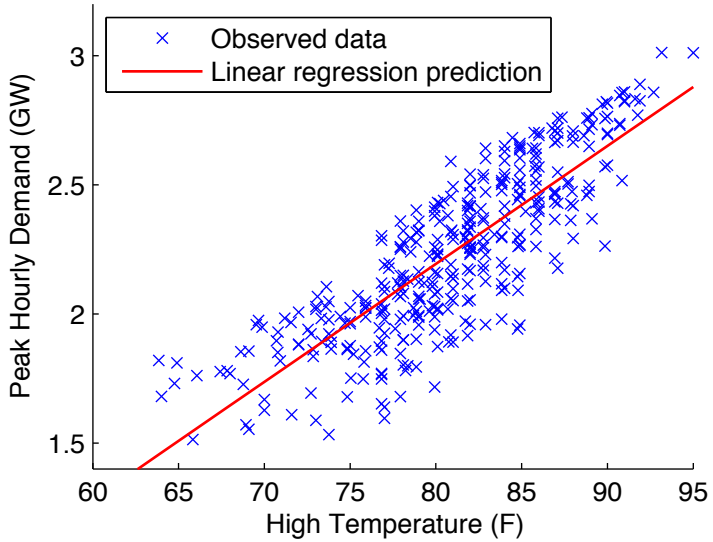
Learning in Linear Regression

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



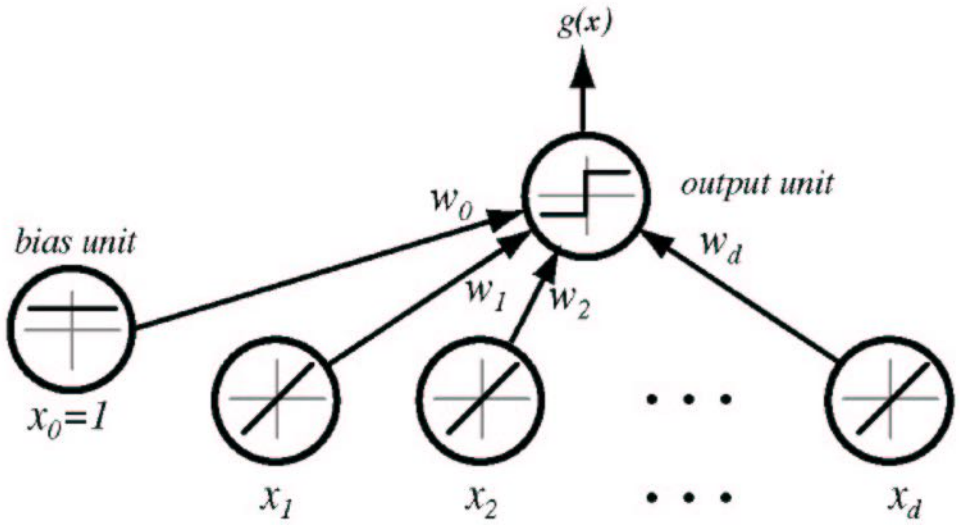
Search algorithm: Start with an initial guess for θ. Keep changing θ (by a little bit) to reduce $J(\theta)$

$$J(\theta) = \sum_{i=1}^{m} \ell(\hat{y}_i, y_i) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2$$

Now a different vew of the same algorithm……

Das Perceptron

- Die Unterscheidung zwischen 2 Klassen mit einer linearen Funktion kann durch ein ein Netzwerk aus d+1 Eingabeeinheiten und 1 Ausgabeeinheit realisiert werden
- Das Netzwerk wird als *Perceptron* bezeichnet
- Das *Perceptron* ist ein einfaches Beispiel sog . *Neuronaler Netze*



$$g(\vec{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \vec{w}^t \vec{x} + w_0 = 0$$

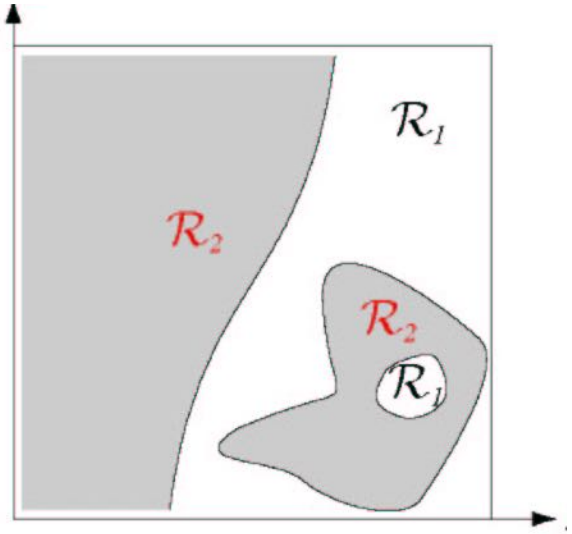
How do we deal with non linear functions ?

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



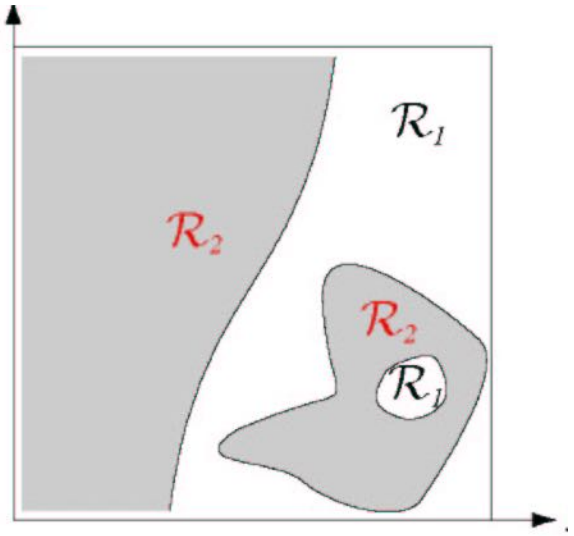
How do we deal with non linear functions ?

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

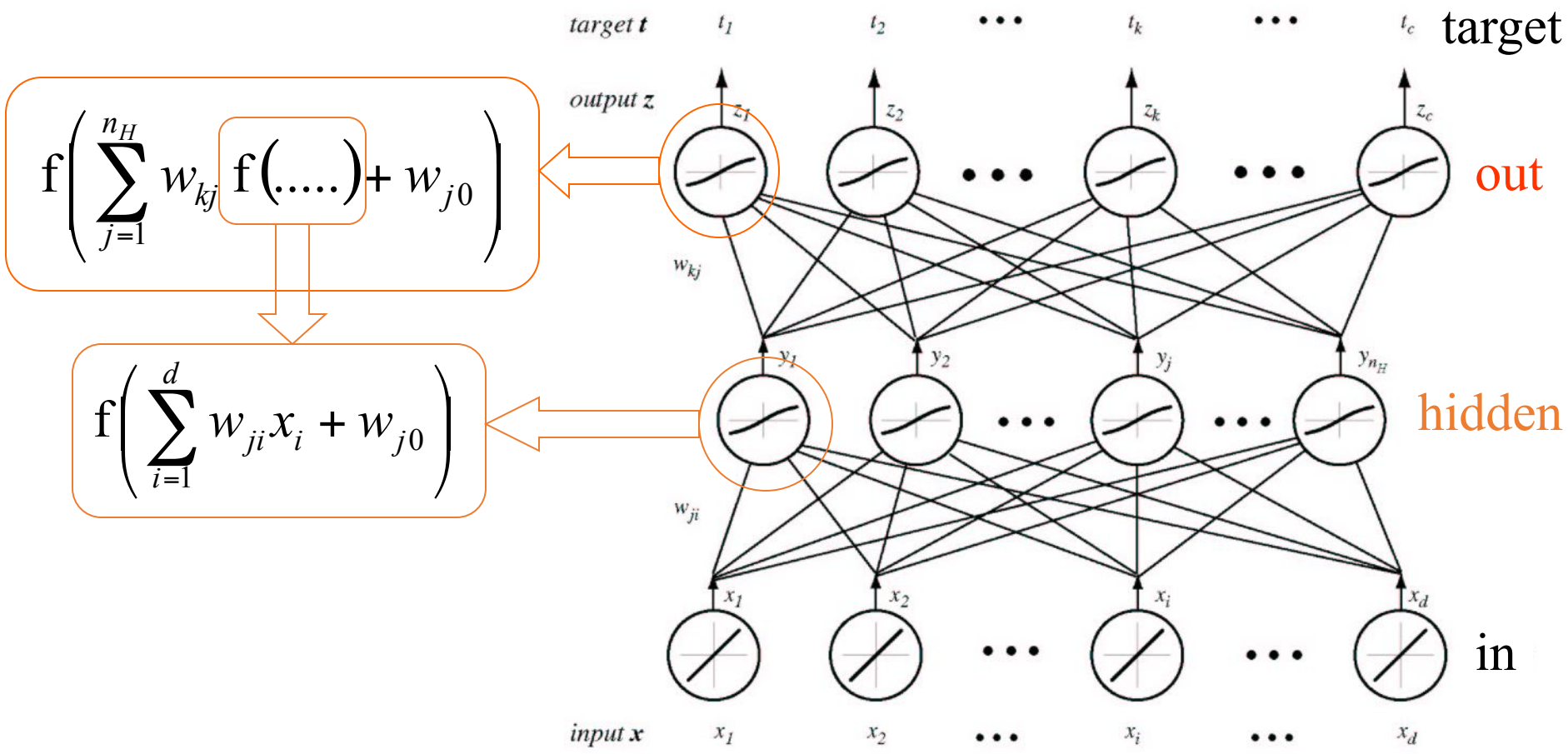
3. use an appropriate optimization techniques to find parameter values that minimize the error function



$$g_k(\vec{x}) = \mathrm{f}\left(\sum_{j=1}^{n_H} w_{kj}\, \mathrm{f}\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0} \right) + w_{j0} \right)$$

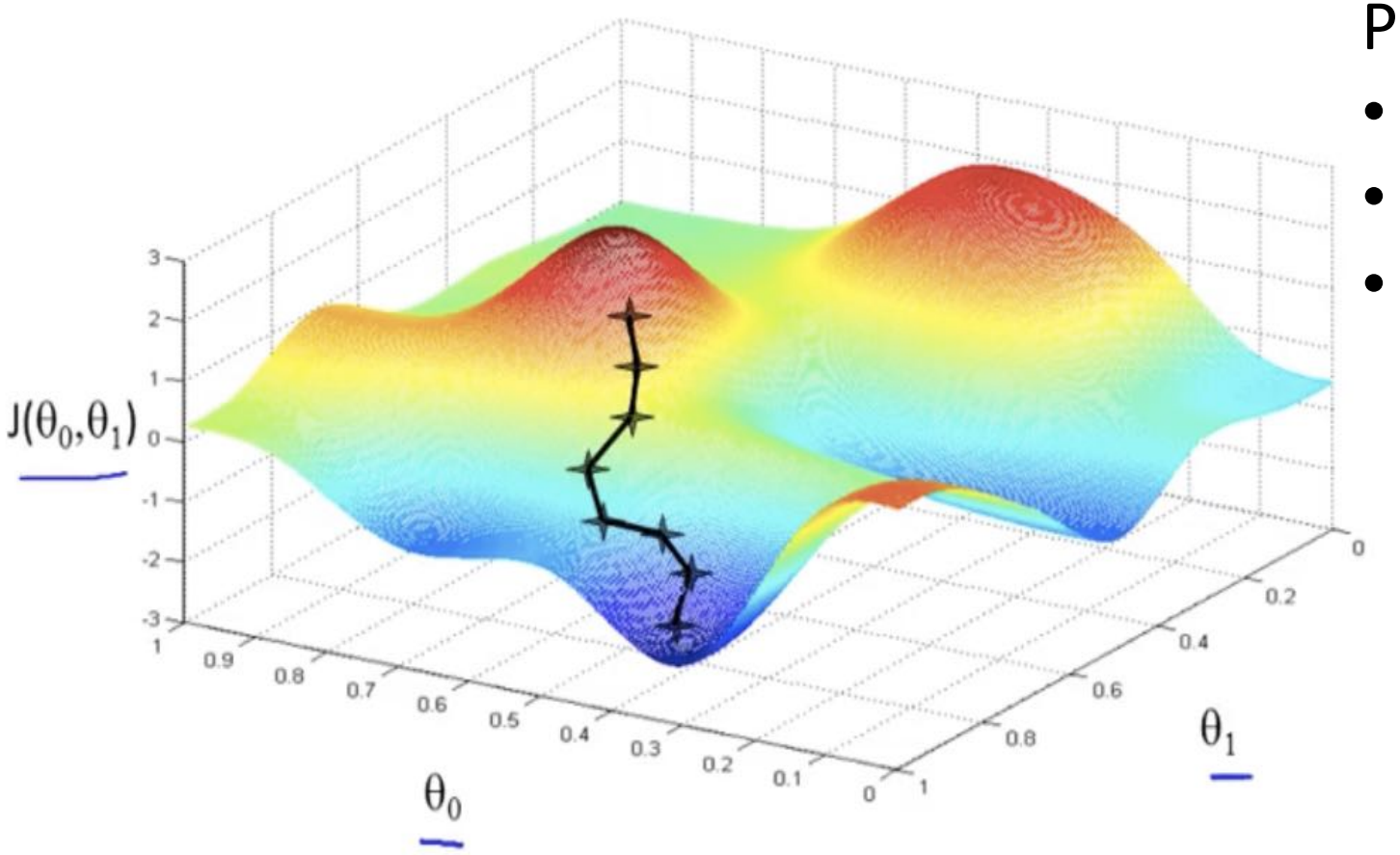
Mehrschichtige Feed Forward Neuronale Netze

- Die Approximation durch Grundfunktionen kann mit einem mehrschichtigen sog ‚feed forward' neuronalen Netz realisiert werden



Embedded Intelligence

Gradientenabstieg (Gradient Descent)

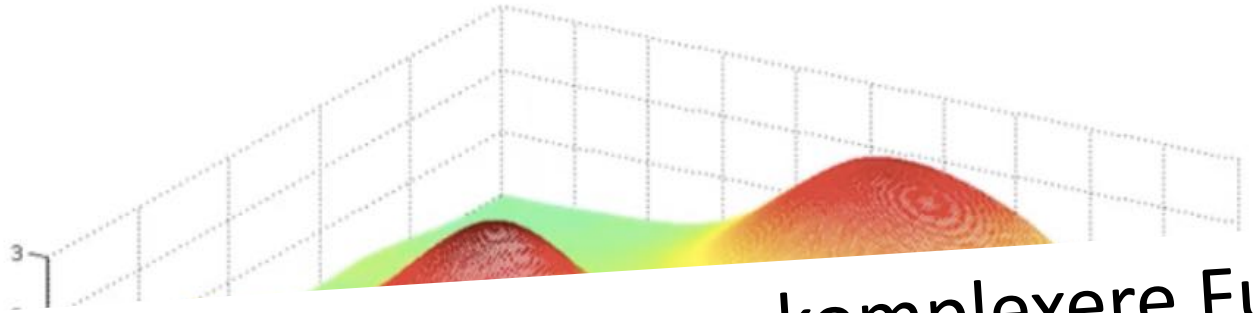


Probleme:

- lokale Minima
- schwache Gradient („Ebenen“)
- Bestimmung des Gradienten für komplexe Funktionen

Und das allem bei mehreren Hundert Dimensionen (eine Dimension pro Parameter) !

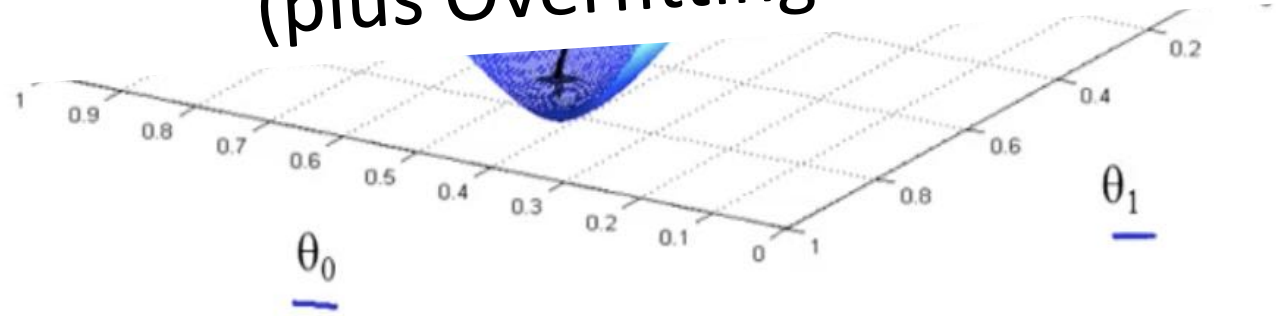
Gradientenabstieg (Gradient Descent)



Probleme:

- lokale Minima
- schwache Gradient („Ebenen“)

Je mehr Parameter um so komplexere Funktionen kann Gradienten für man darstellen aber dafür wird die Optimierung schwerer nen (plus Overfitting aber dazu später)



Und das allem bei mehreren Hundert Dimensionen (eine Dimension pro Parameter) !

Maßnahmen beim Training

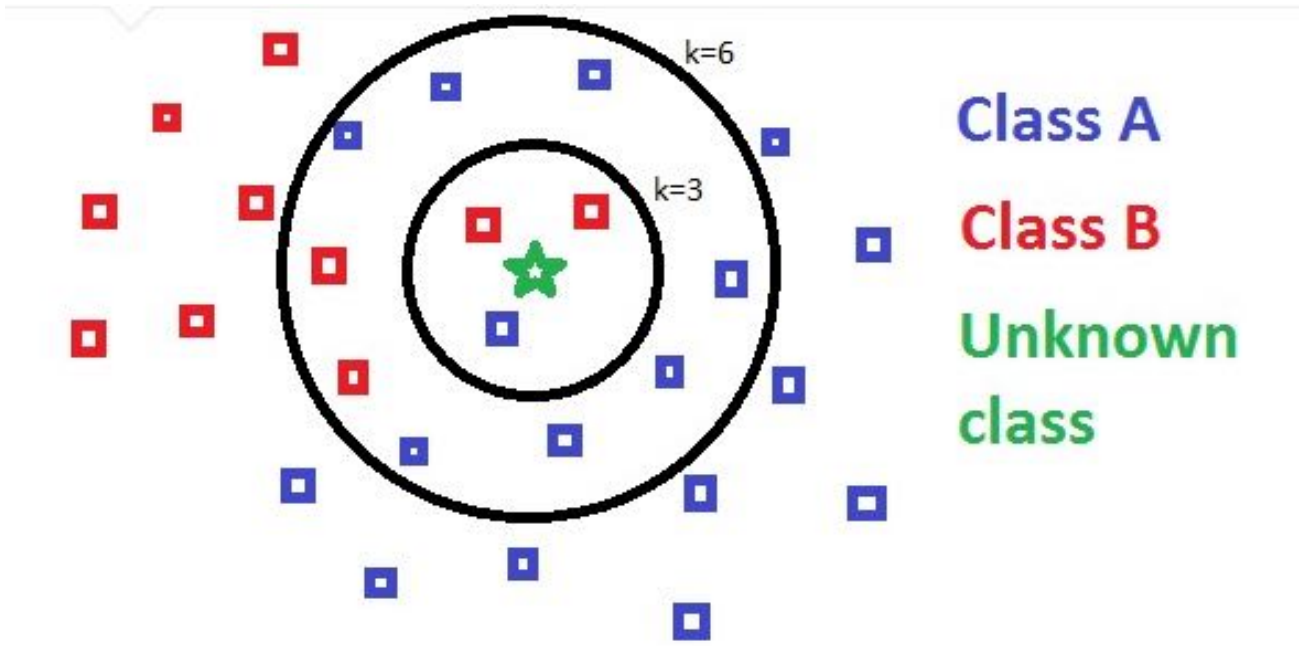
Beim Design und Training des Netzes muss vieles beachtet werden

- Skalierung der Gewichte
- Wahl der Aktivierungsfunktionen
- Anzahl der hidden Neuronen
  - zu gross -> overfitting
  - zu klein -> gewünschte Funktion kann nicht dargestellt werden
- Initialisierung der Gewichte
- Wahl der Trainingsdaten
- Trainingsmodus
- .....

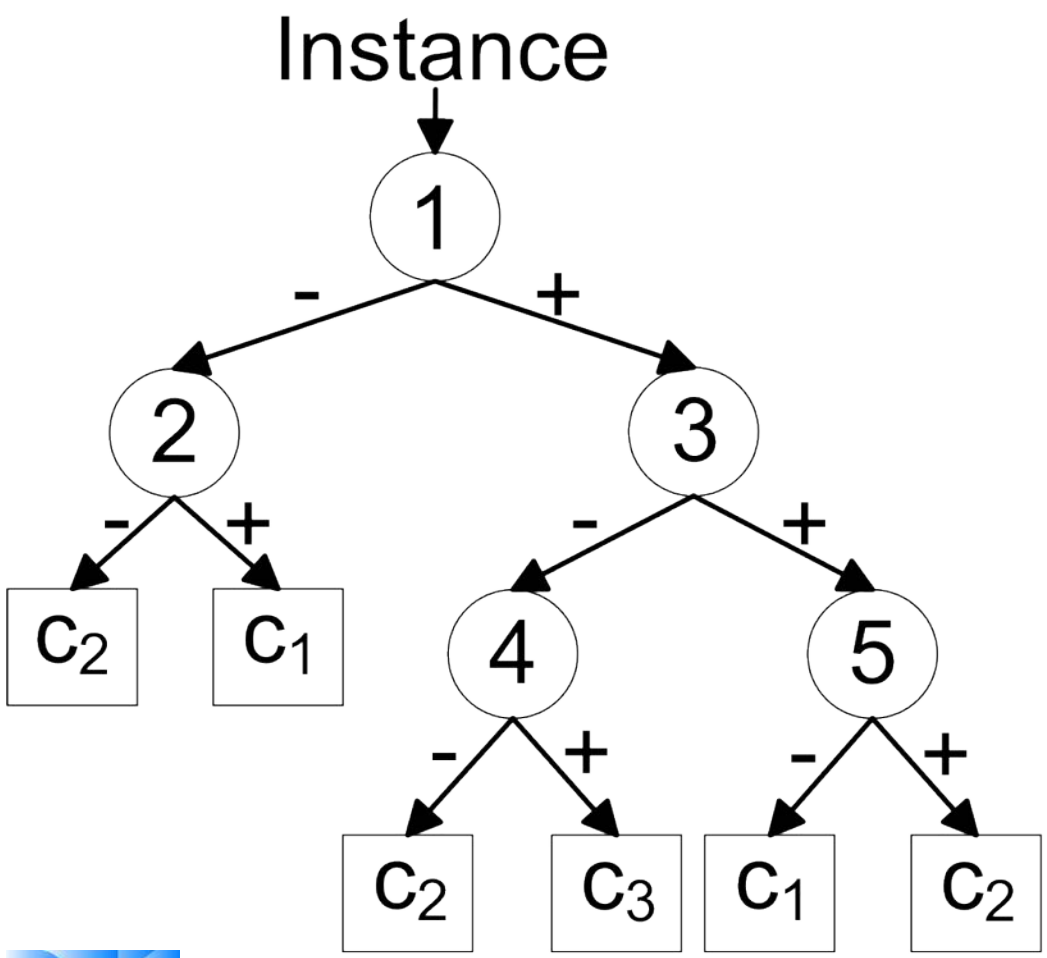
....oder einfache Methoden

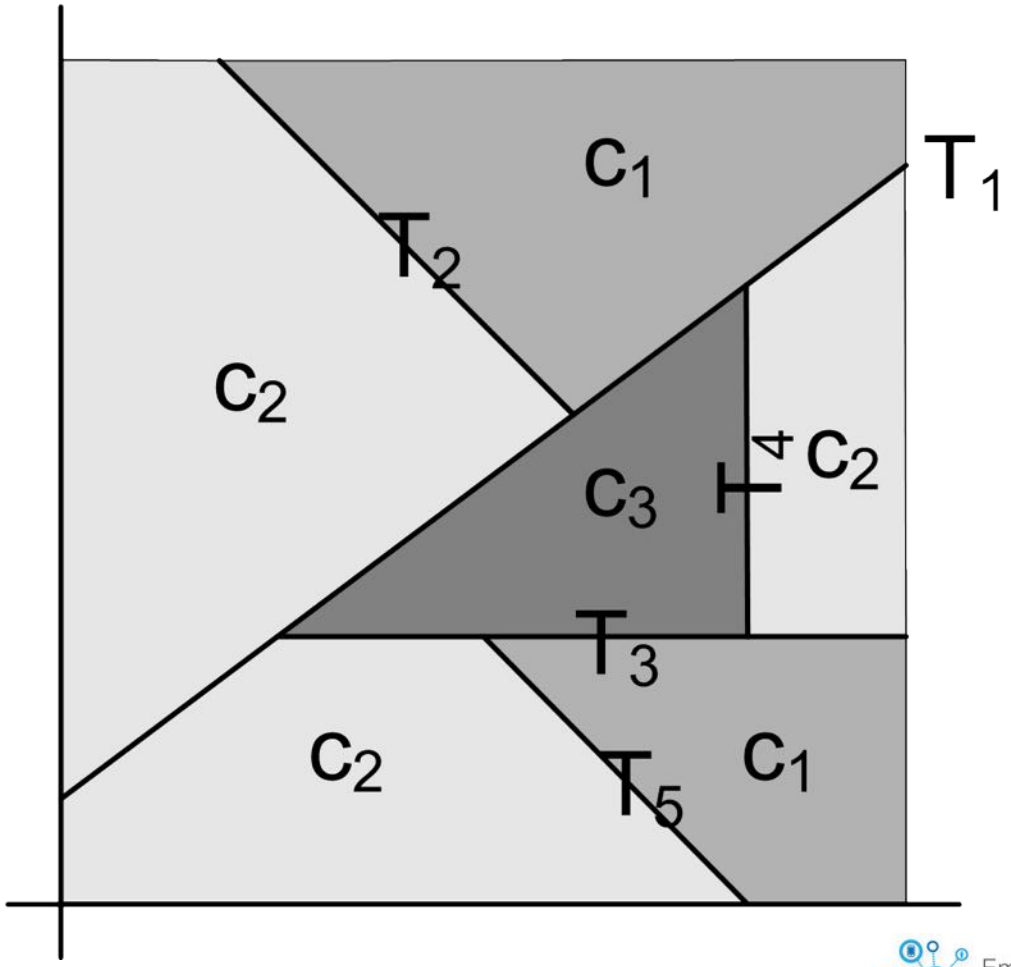
KNN (K Nearest Neighbour)

- store all points in the training set including labels
- assign points to be classsified according to the class of most ist N neghbours

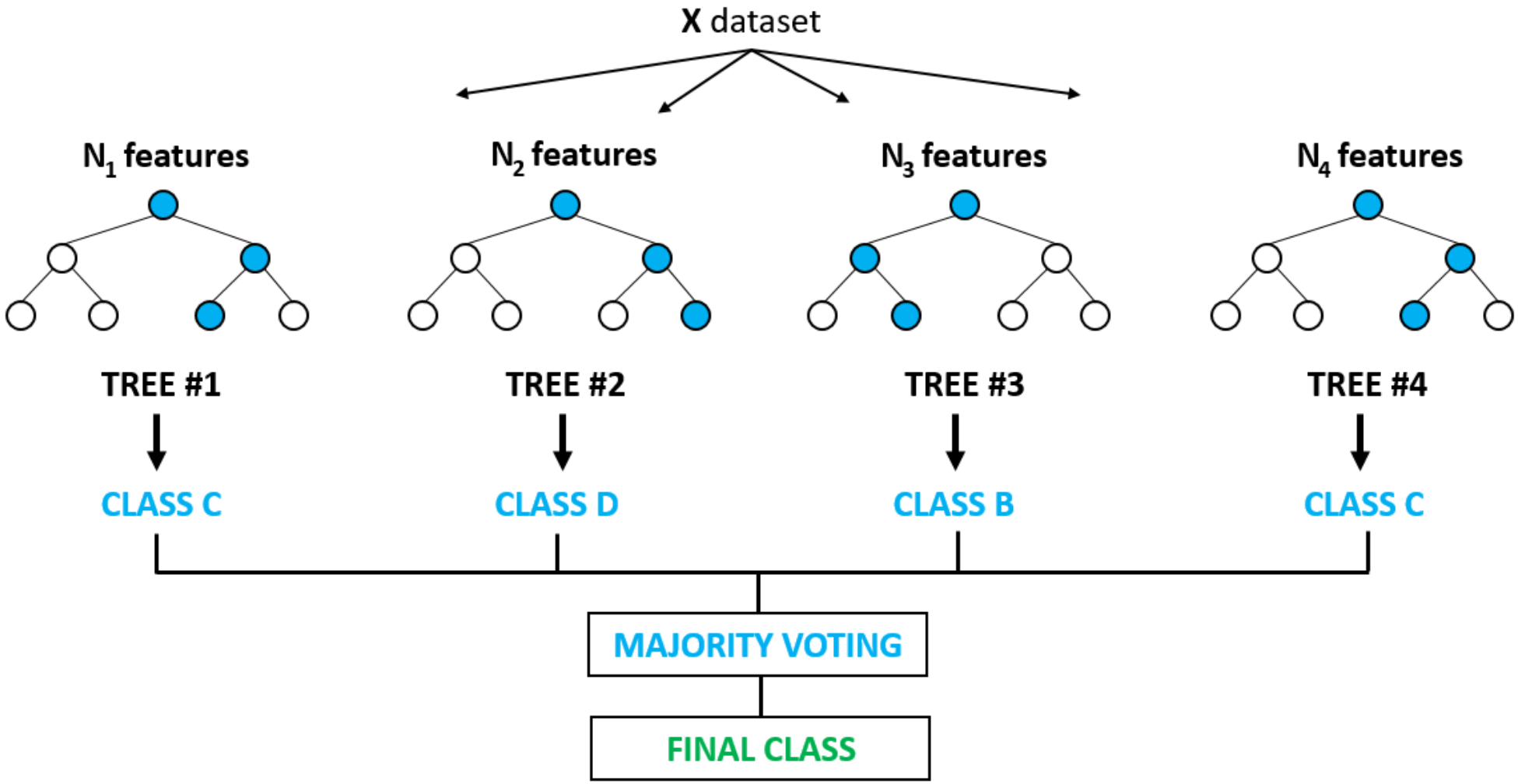


Decision Tree Classifier



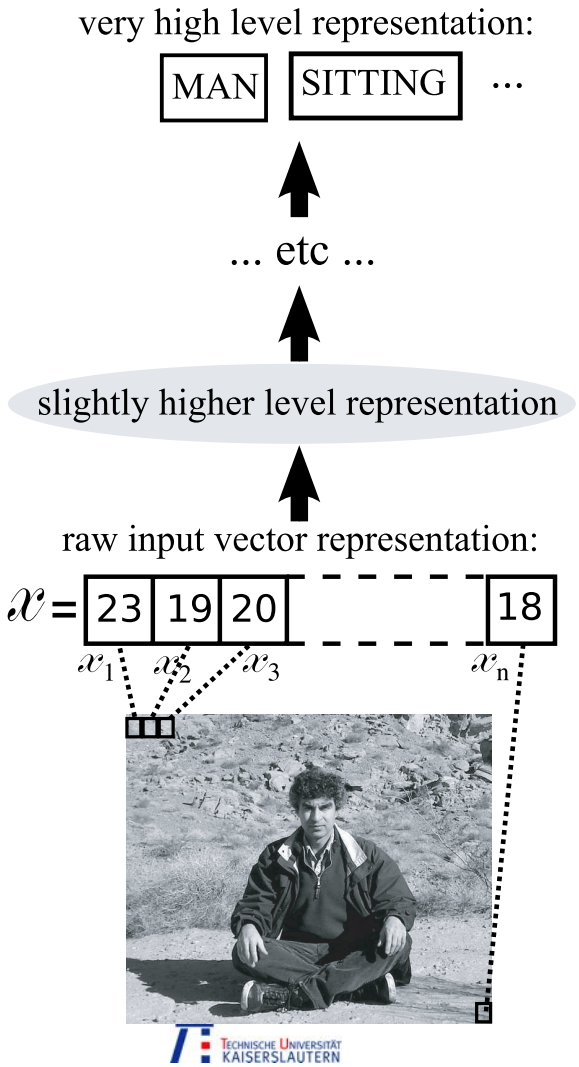


Random Forest



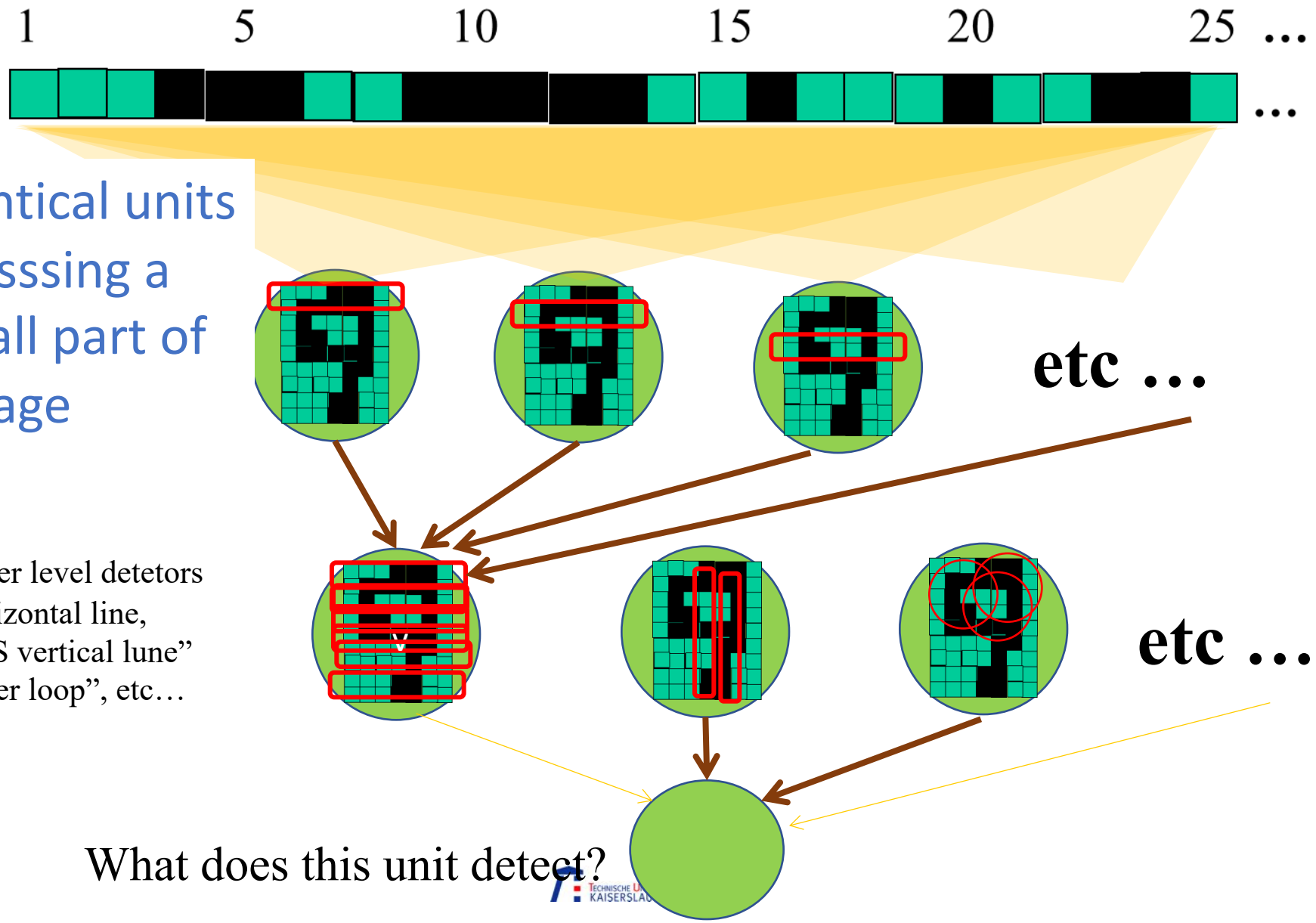
….oder komplexe Architekturen (Stichwort Deep Learning)

Reducing the number of parameters: distributed representations



Two lessons to learn…

1. Simple, identical units each processsing a different small part of the image



2. Succesive layers each representing higher levels of abstracton

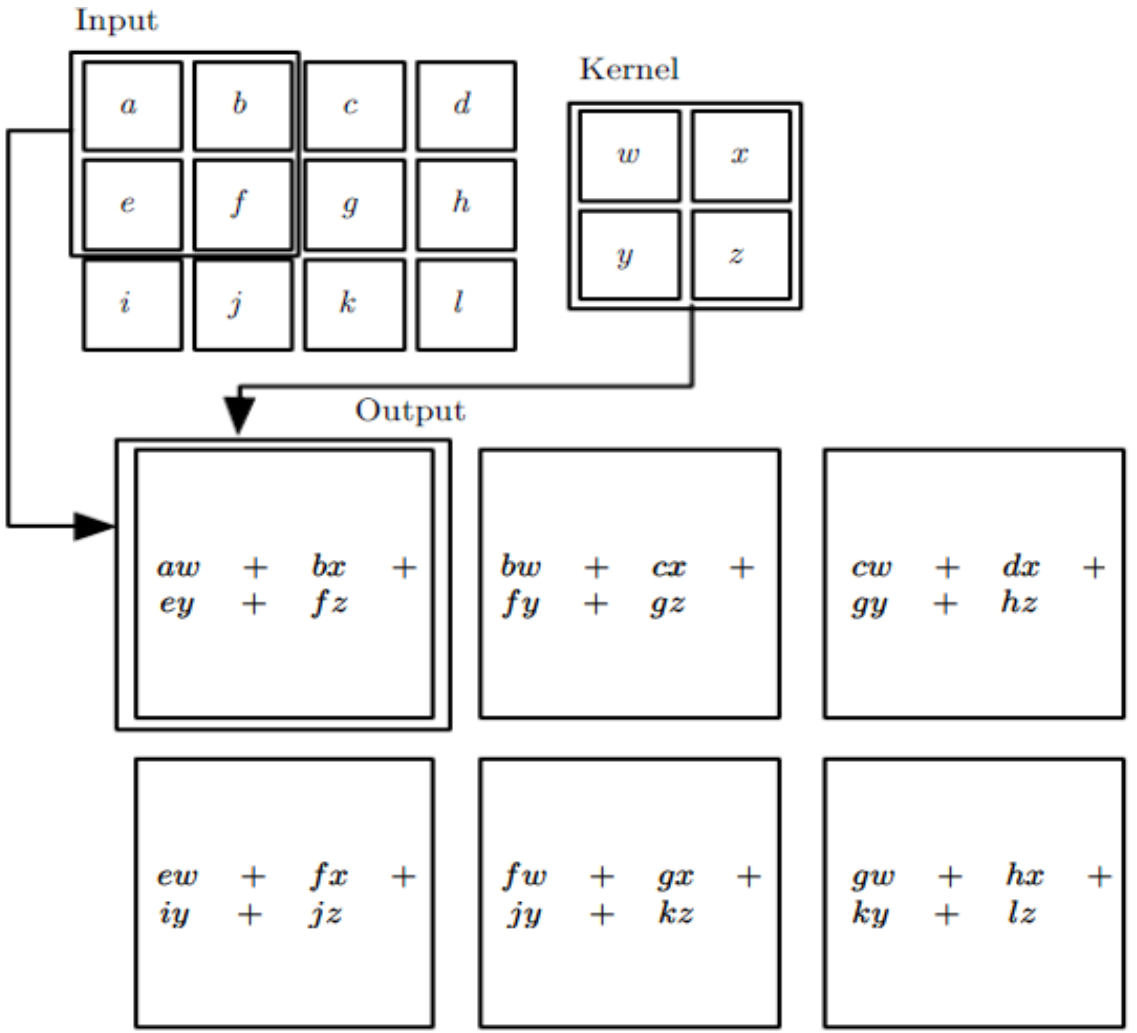
Convolution

Convolution in 1 Dimension:

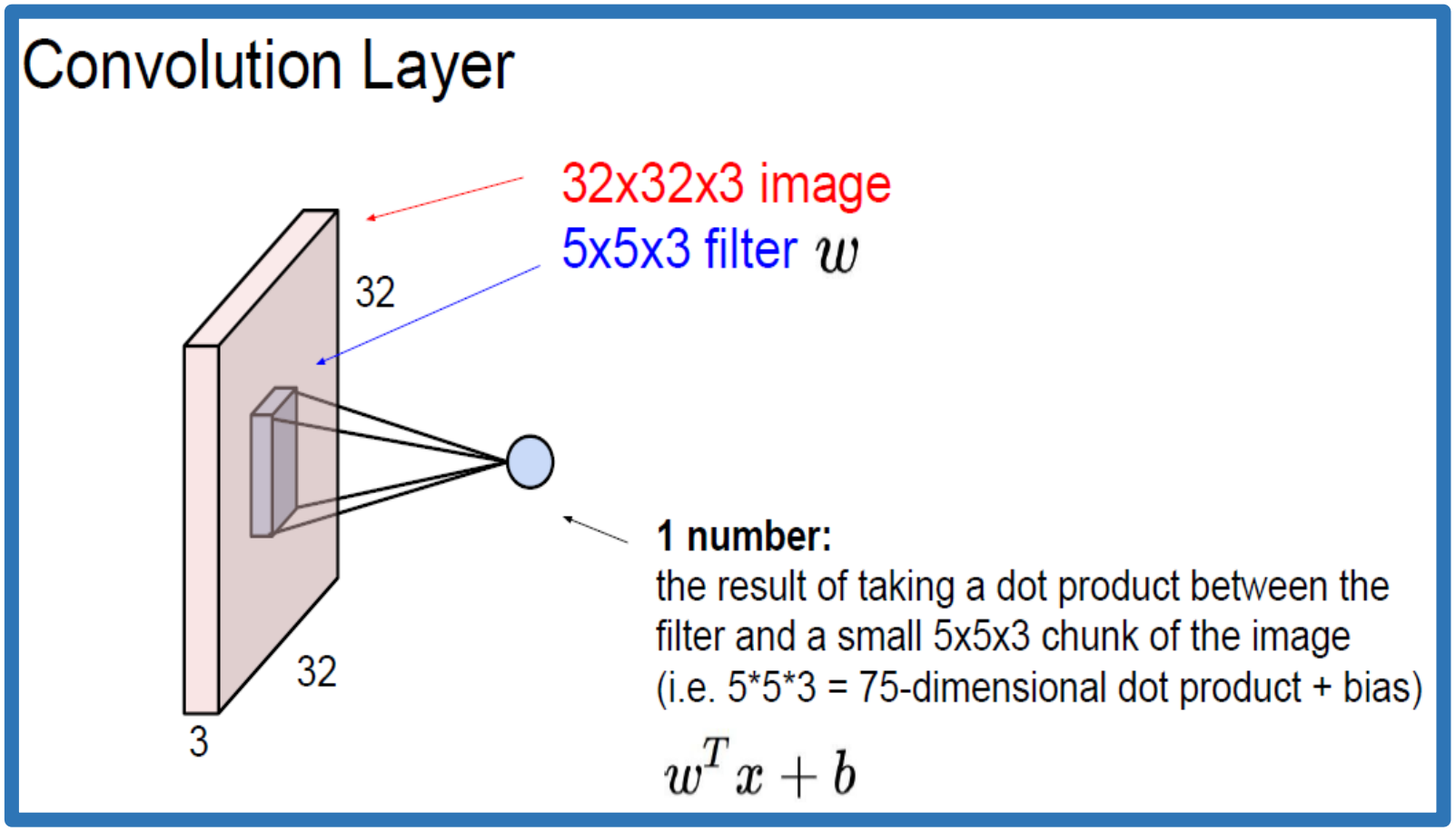
$$y[n] = \sum_{k=-\infty}^{k=\infty} x[k]h[n-k]$$

Convolution in 2 Dimensions:

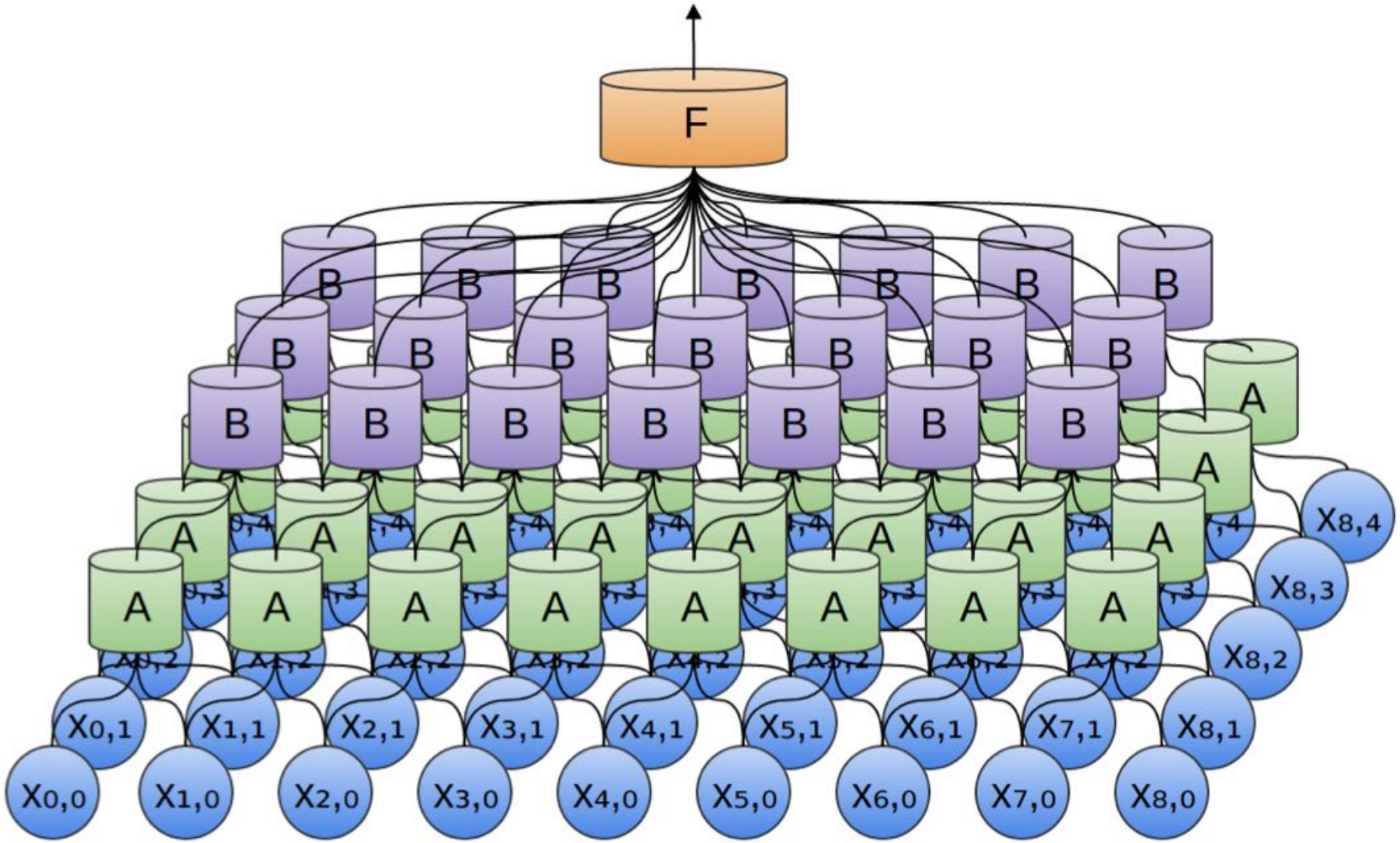
$$y[n_1, n_2] = \sum_{k1=-\infty}^{k1=-\infty} \sum_{k2=-\infty}^{k2=\infty} x[k_1, k_2]h[(n_1 - k_1), (n_2 - k_2)]$$



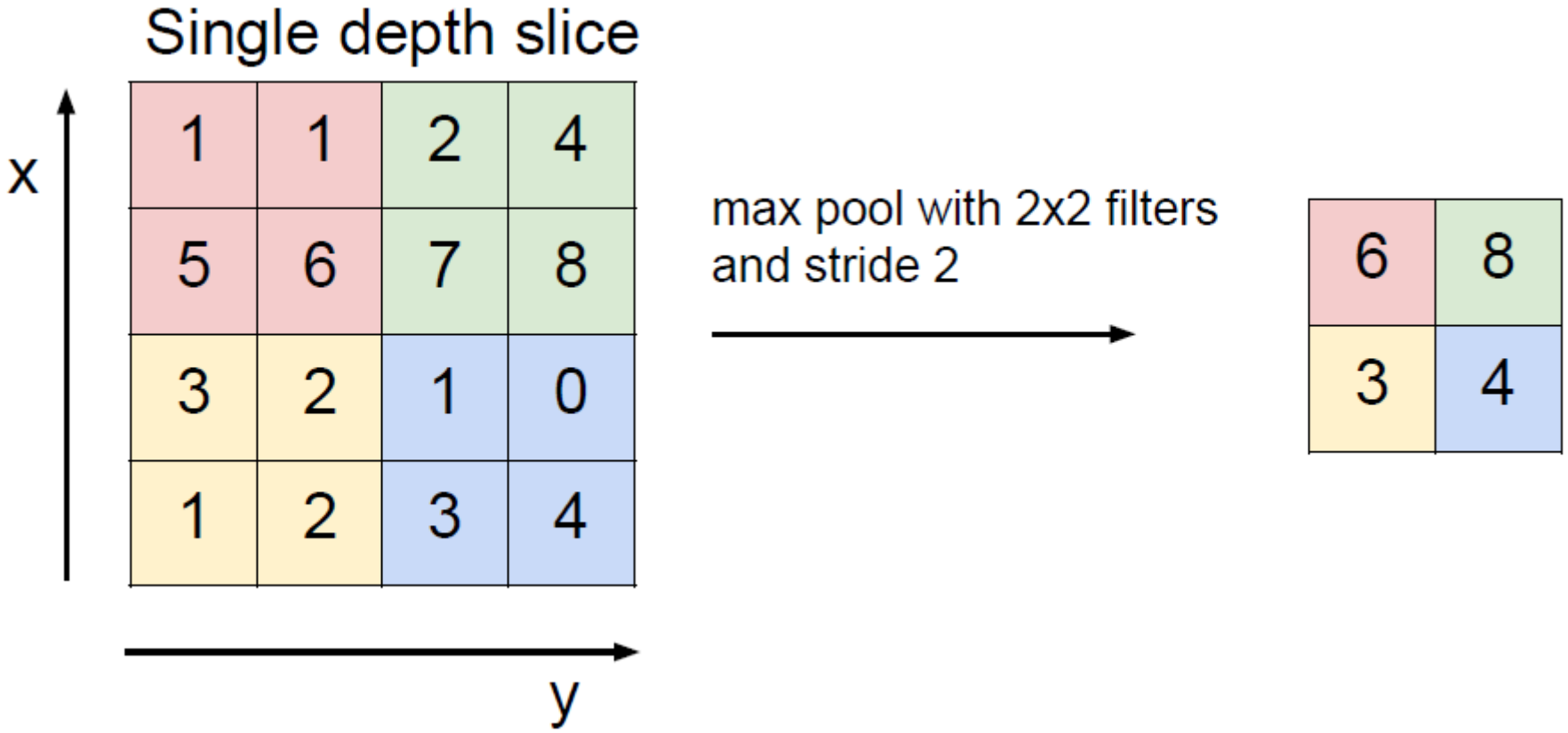
- The convolution layer computes dot products between the filter and a piece of image as it slides along the image
- The step size of slide is called stride
- Without any padding, the convolution process decreases the spatial dimensions of the output

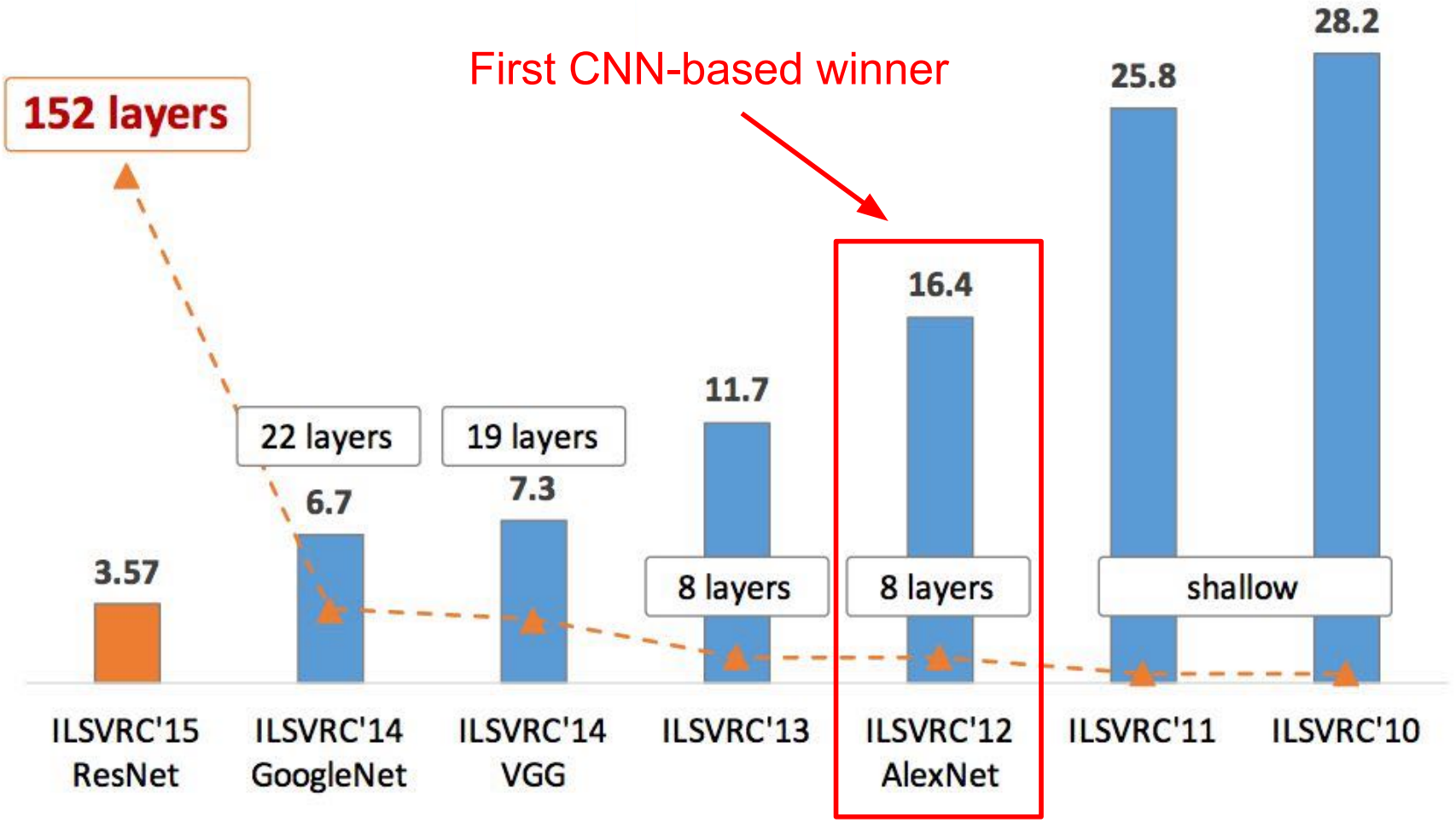


Other Networks: Convolution Networks

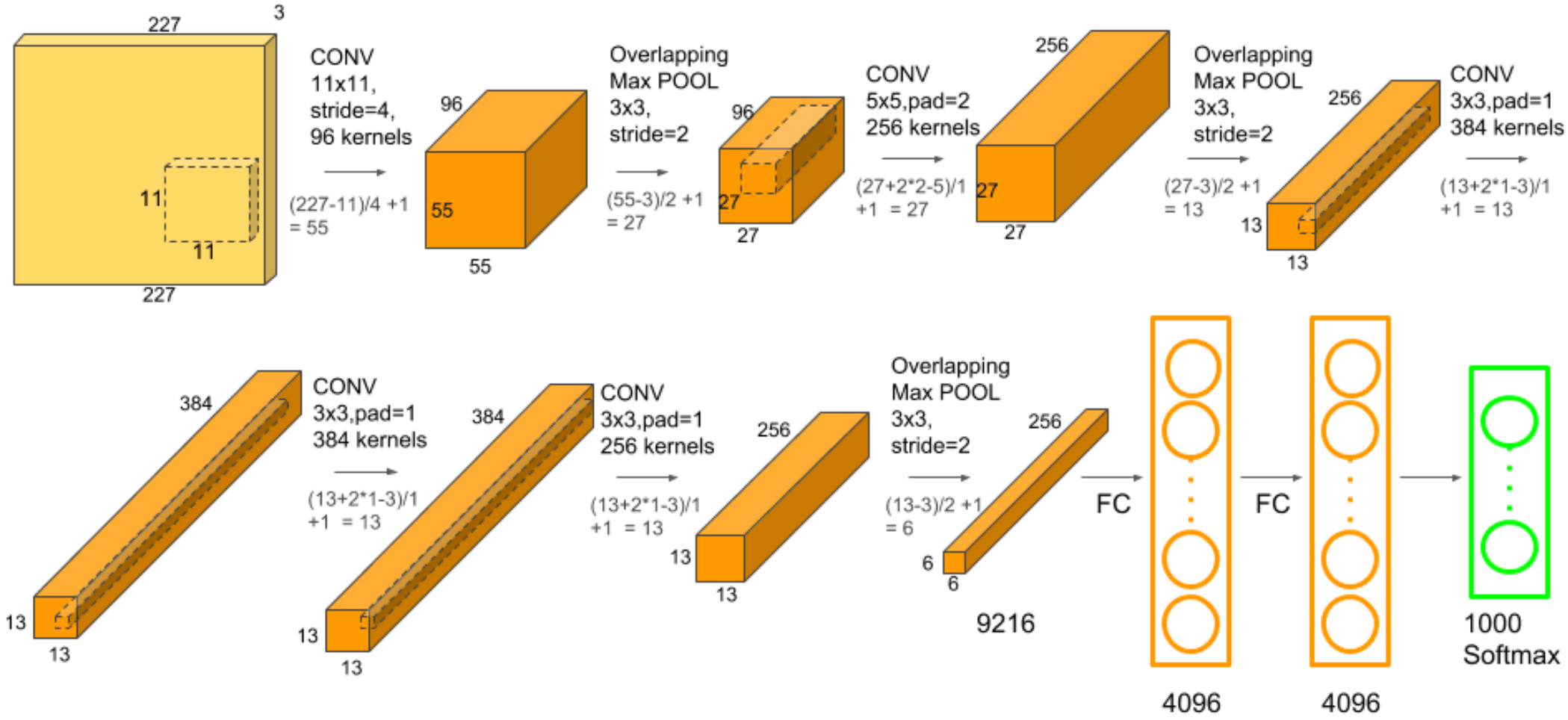


Max Pooling Illustration




First CNN-based winner
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10

AlexNet



vollverbundenes „standard“ Netz

A more systematic view

Machine Learning

target functions

1. describe all possible ~~shapes~~ through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function

Input: A sample of N points in D dimensional space possibly with additional attributes

Result: A target function derived from the sample to do „something usefull“

on data points that are related to the original space but as such have not been seen before !!!

A more systematic view

Machine Learning

Input: A sample of N points in D dimensional space possibly with additional attributes

1. describe all possible target functions through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

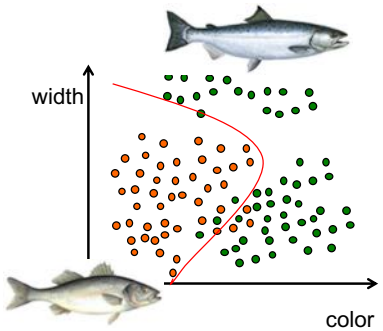
- **supervised (we tell the system what is right)**
- **un-supervised (system analyses „structure“)**

3. use an appropriate optimization techniques to find parameter values that minimize the error function

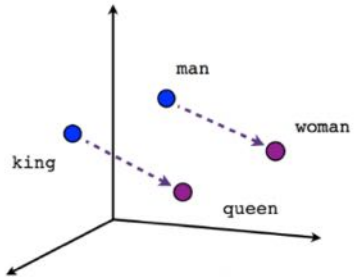
Result: A target function derived from the sample to do „something usefull“

Result: A target function derived from the sample
to do „something usefull“

on data points that are related to the original space but as such have not been seen before !!!

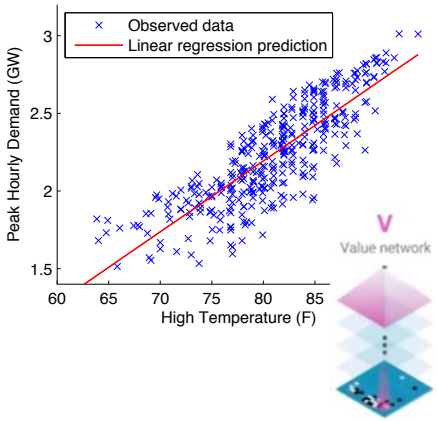


classification



Male-Female

(mapping to)
new representation

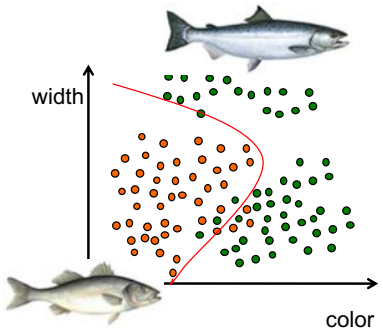


value predicton

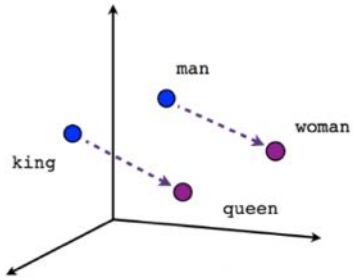
generation of new
samples (completion)

Result: A target function derived from the sample
to do „something usefull“

on data points that are related to the original space but as such have not been seen before !!!

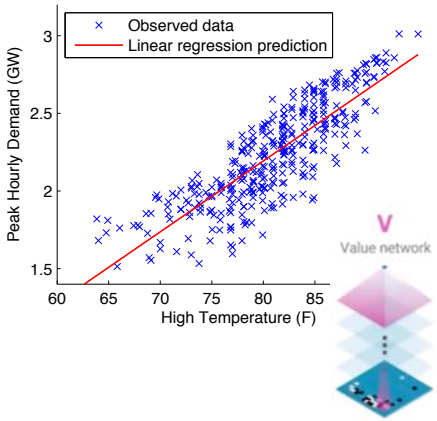


classification
(mostly supervised)



Male-Female

(mapping to)
new representation
(often unsupervised)

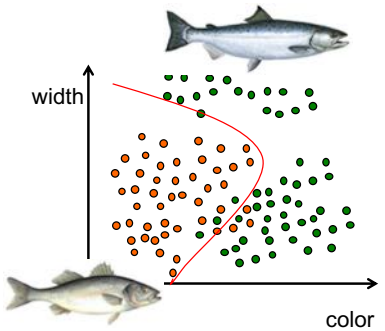


value predicton
(mostly supervised)

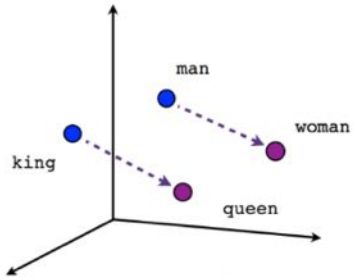
generation of new
samples (completion)
(supervised or/and unsupervised)

Result: A target function derived from the sample
to do „something usefull“

Some general considerations valid for all cases on the example of classification

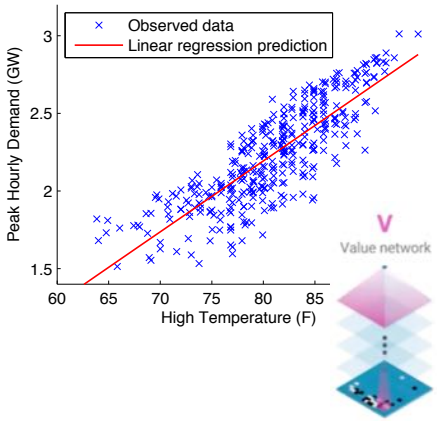


classification



Male-Female

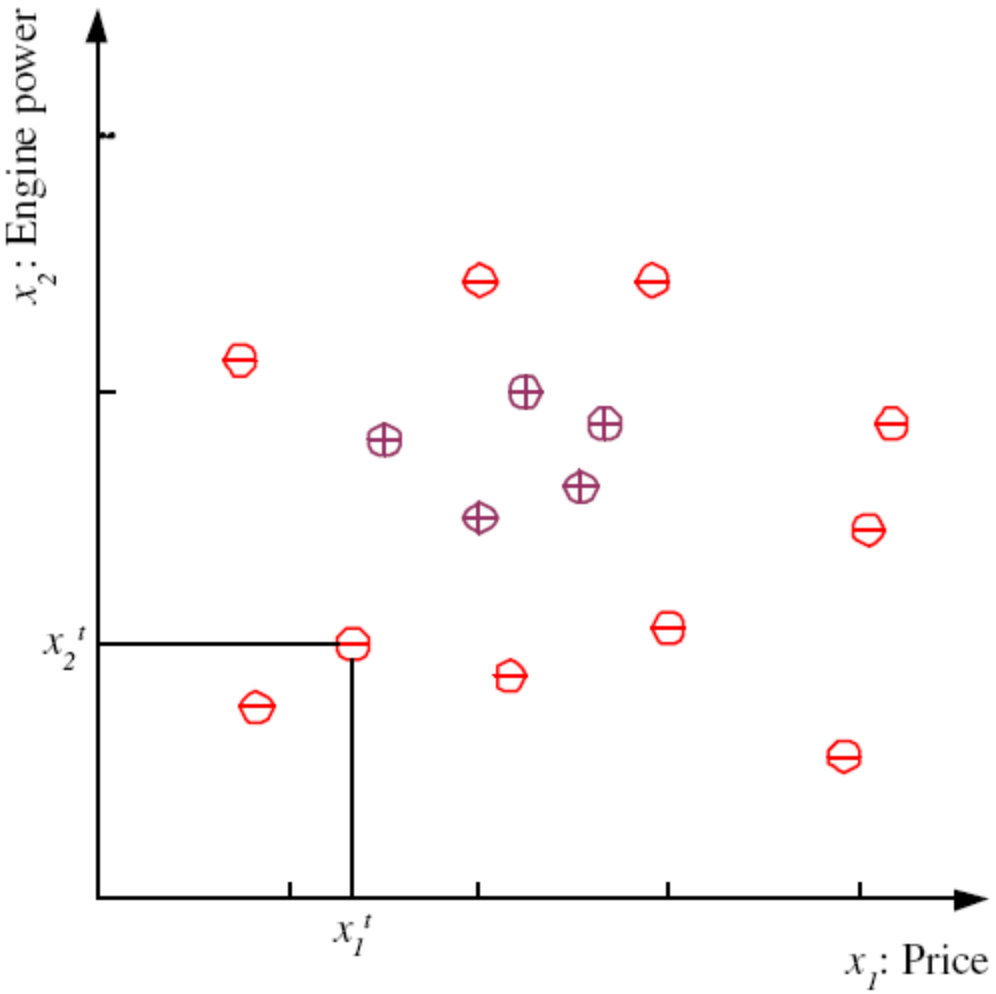
(mapping to)
new representation



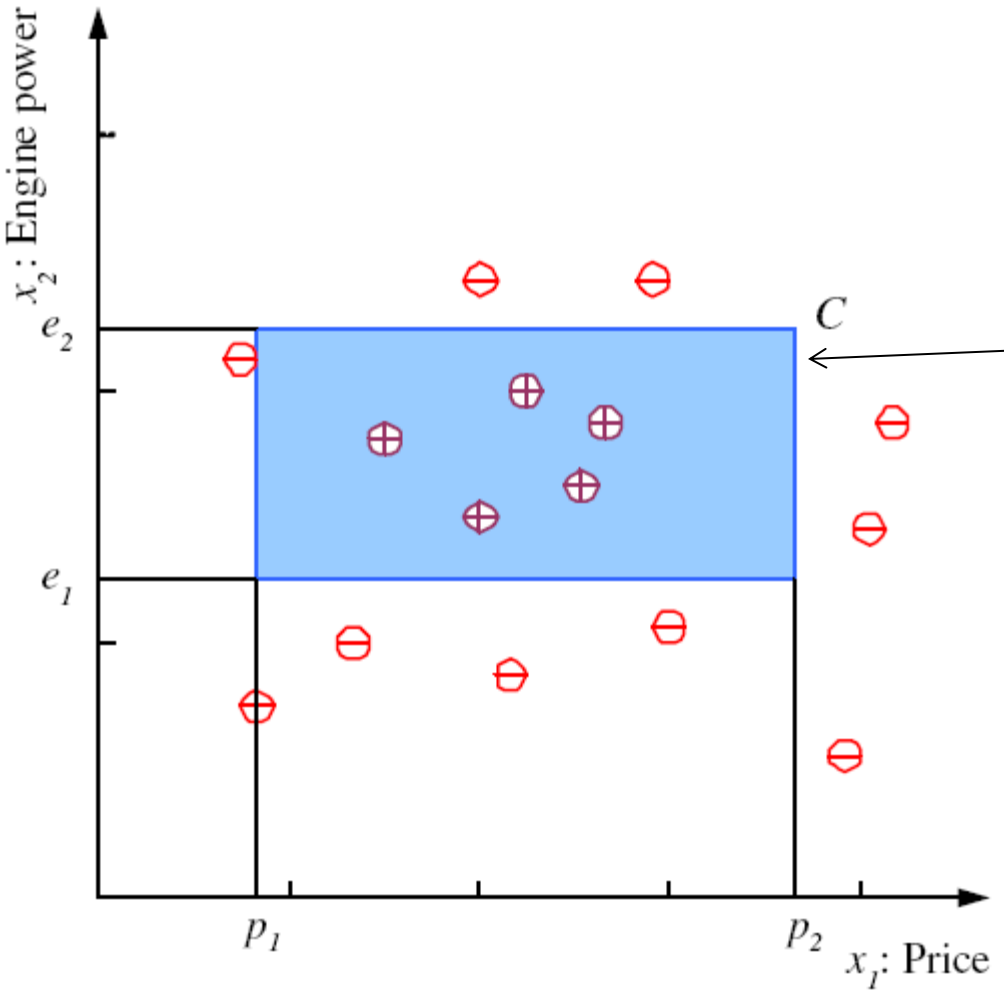
value predicton

generation of new
samples (completion)

Trainingset für positive und negative Beispiele

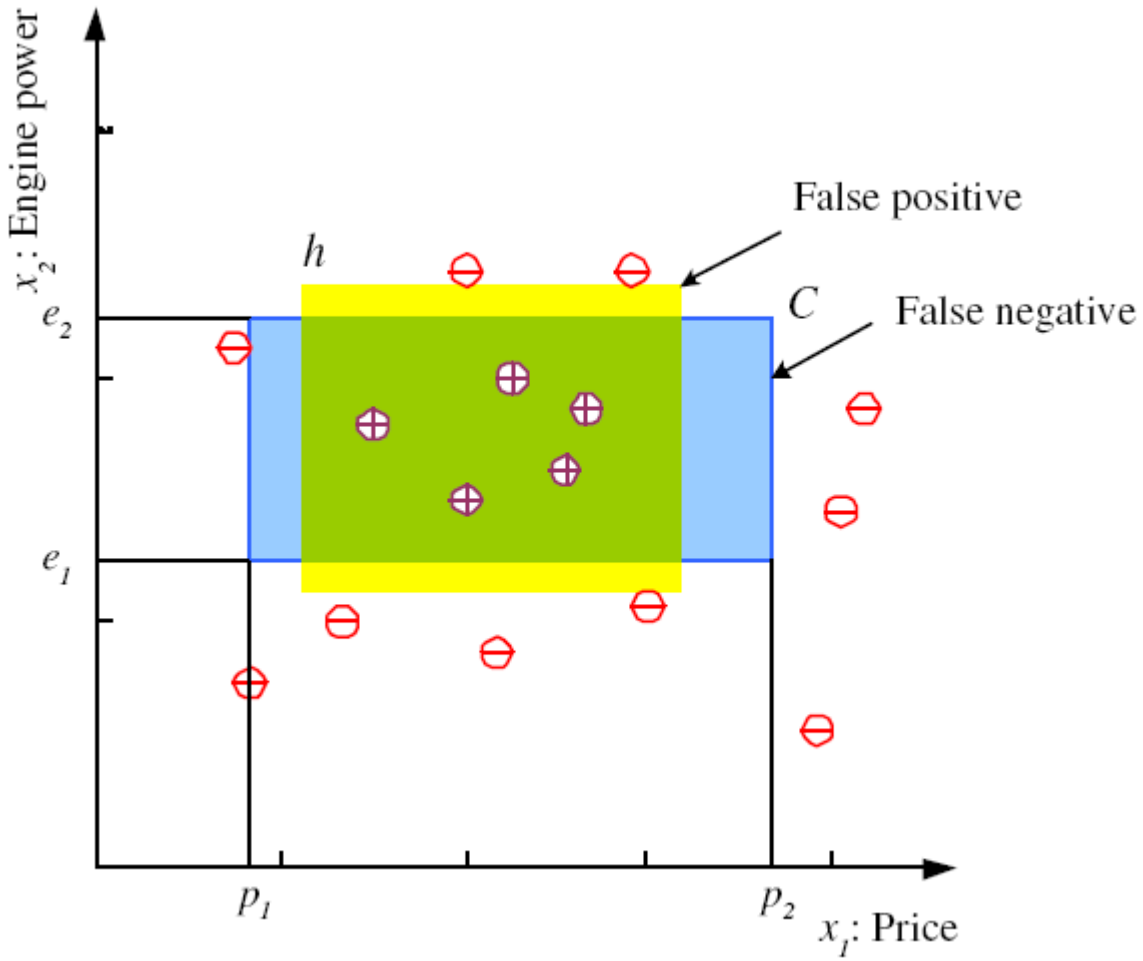


Annahme über C

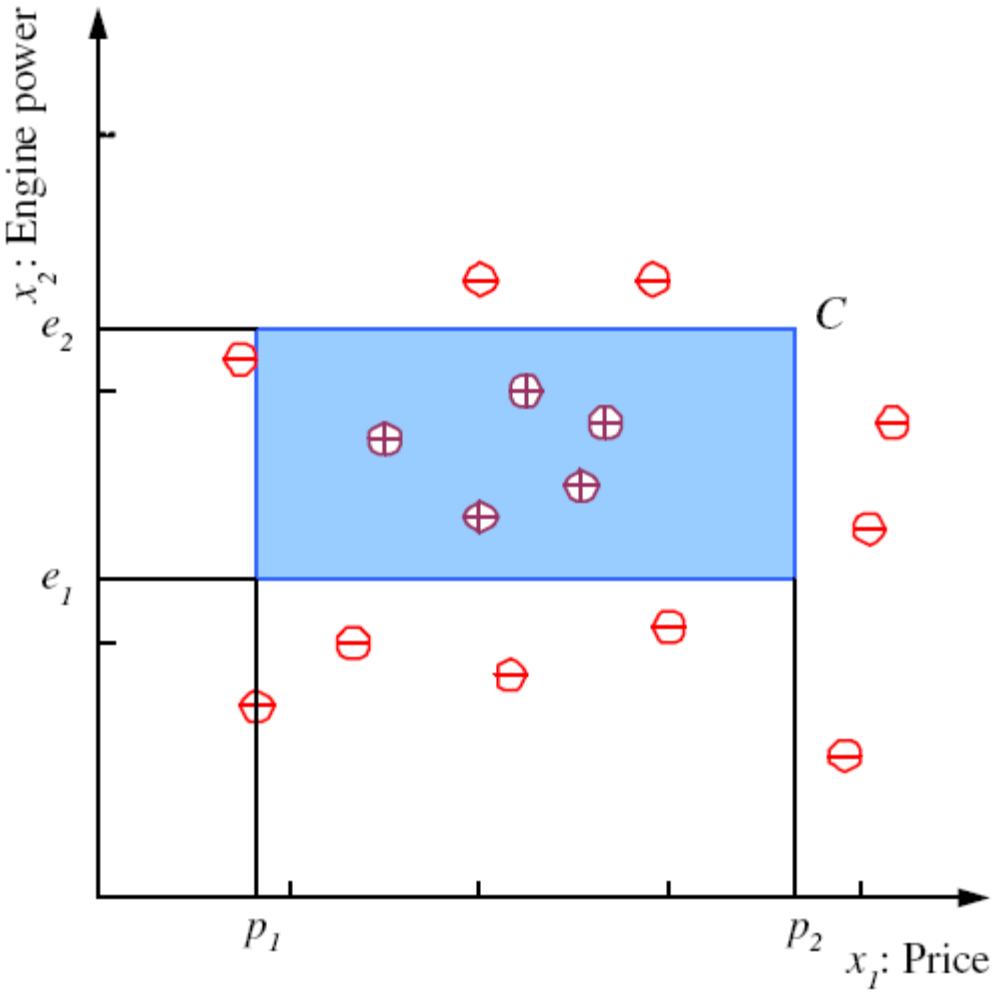


C in Realität
(uns) unbekannt

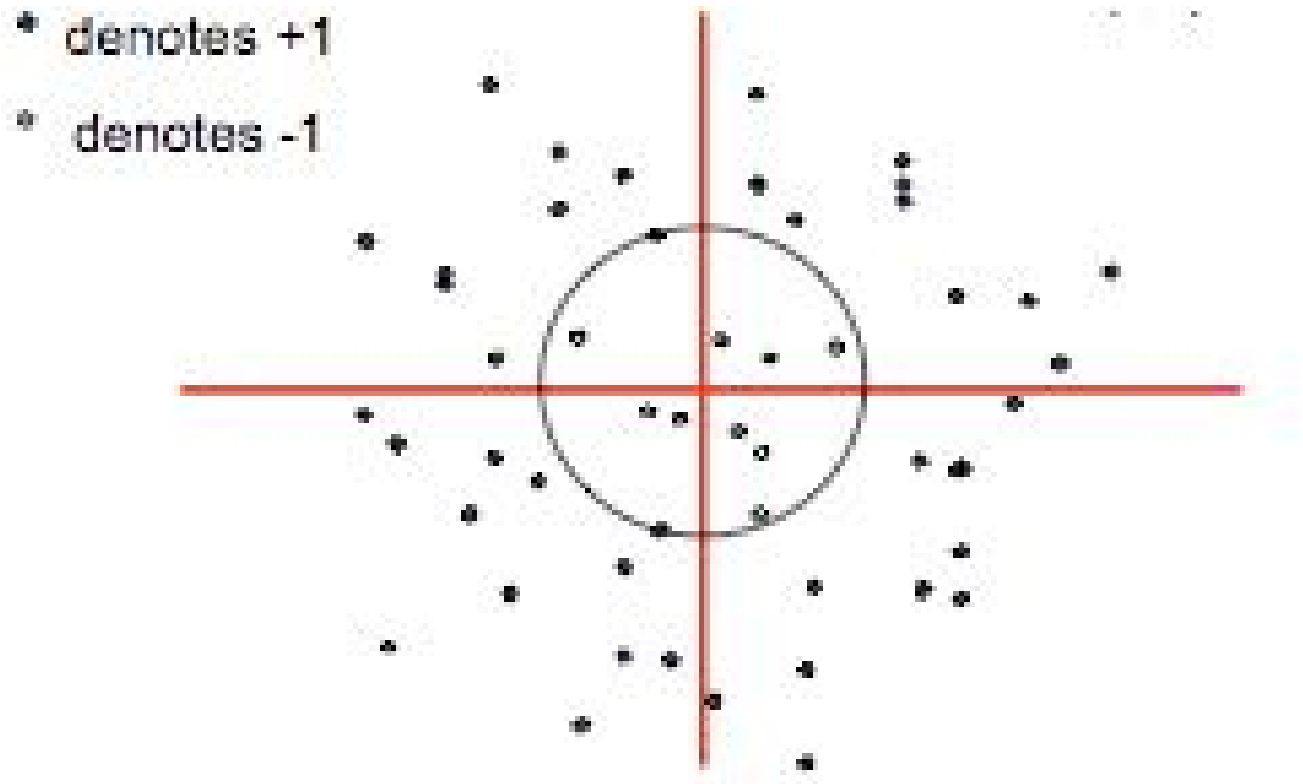
Hypothese h aus $\mathcal{H}$



Hypothesenklasse $\mathcal{H}$

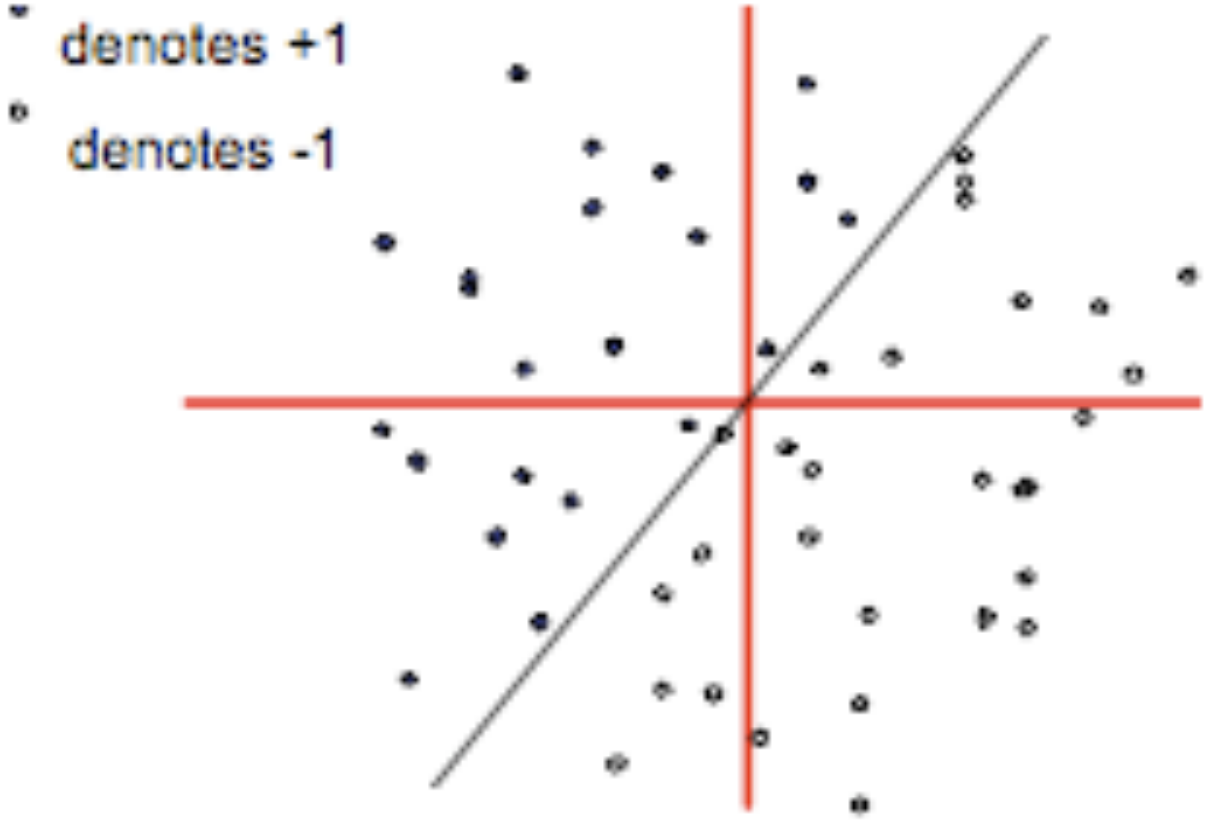


Hypothesenklasse $\mathcal{H}$



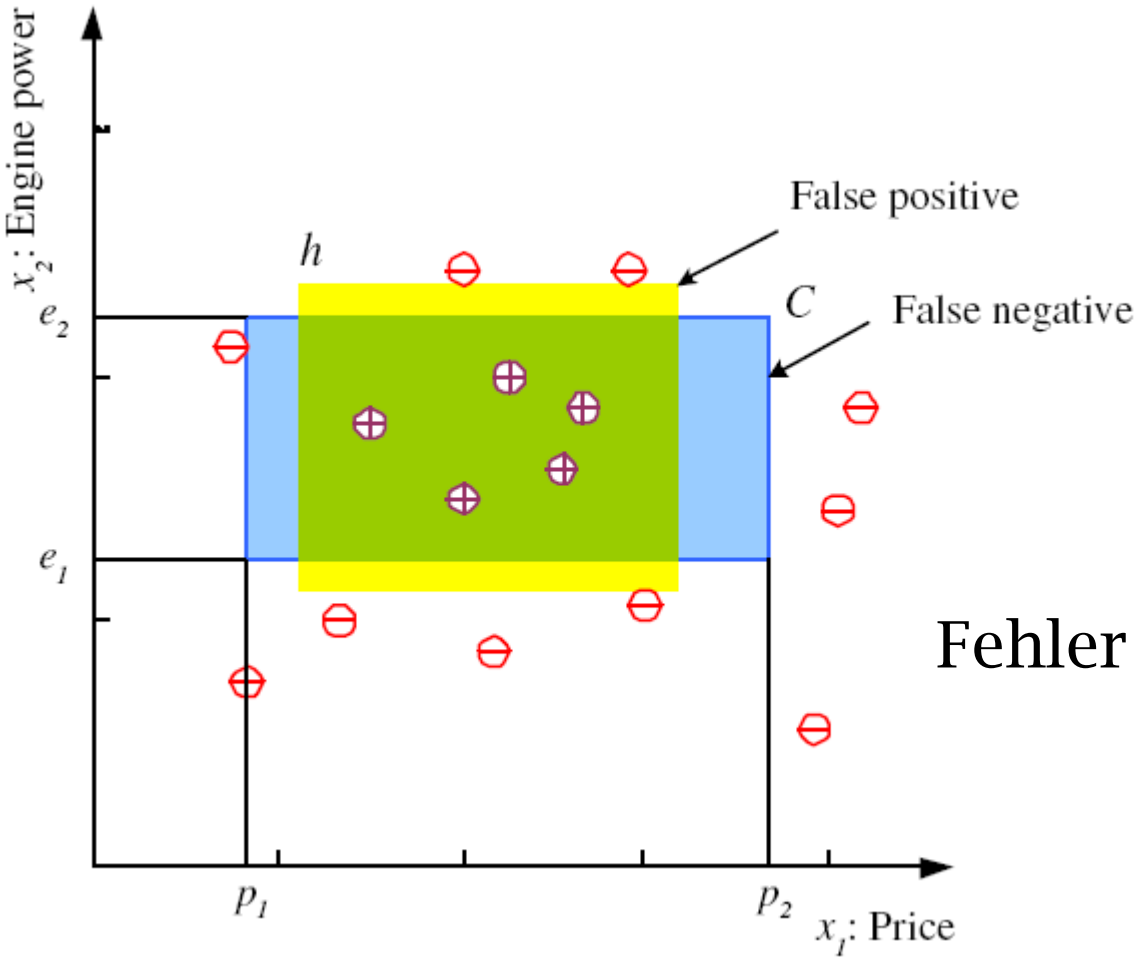
Menge aller geeigneten Kreise wäre eine andere *Hypothesenklasse*

Hypothesenklasse $\mathcal{H}$



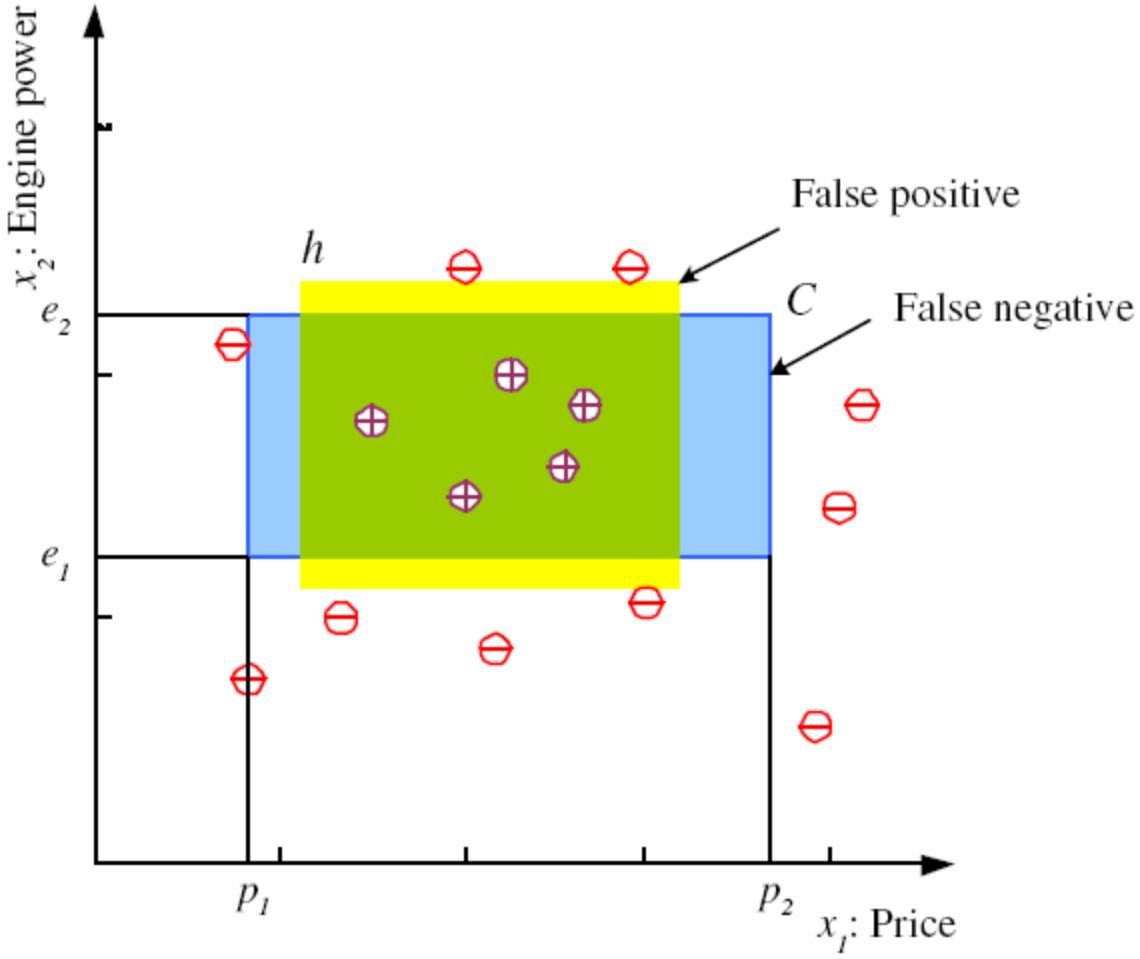
Menge aller geeigneten Geraden wäre eine weitere *Hypothesenklasse*

Empirischer Fehler E(h|$\mathcal{X}$)



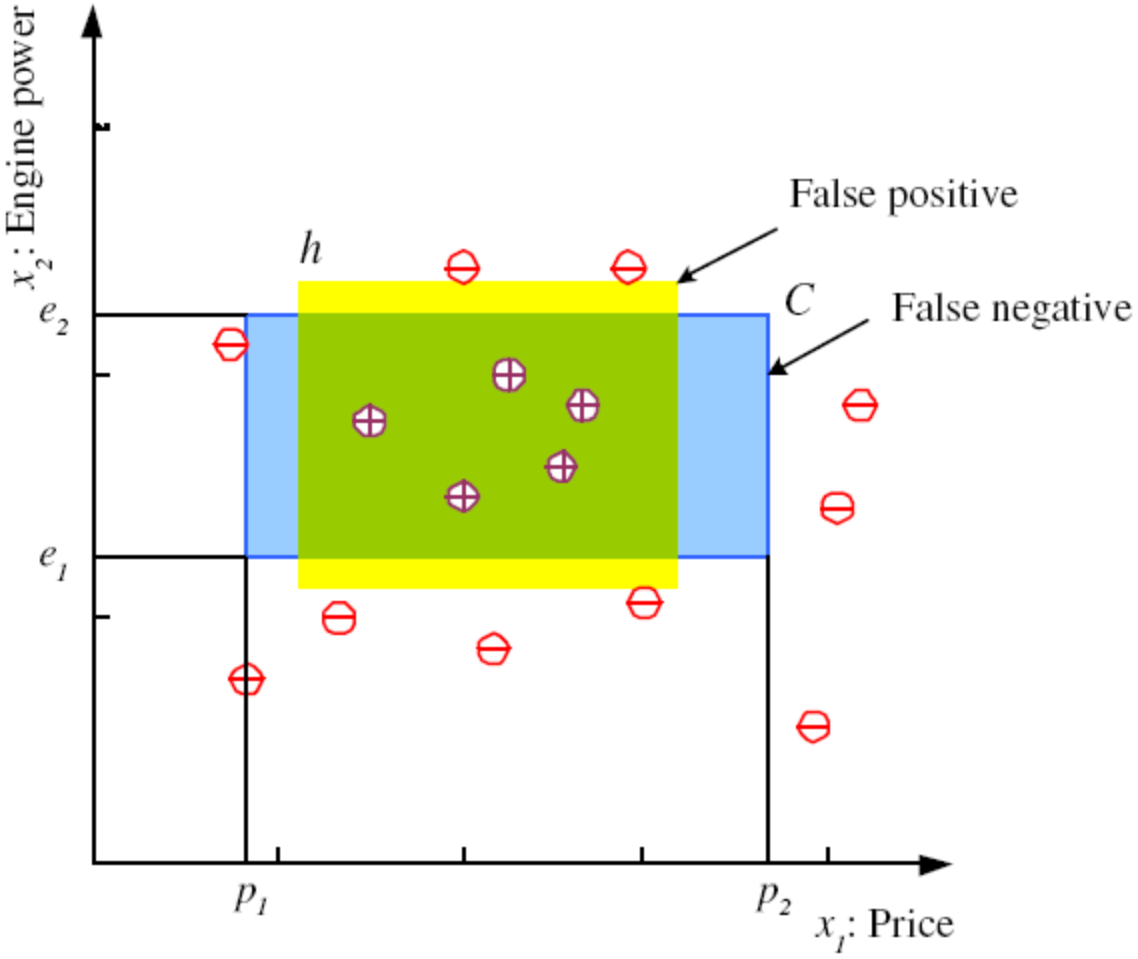
Fehler von h auf $\mathcal{X}$

Generalisierung



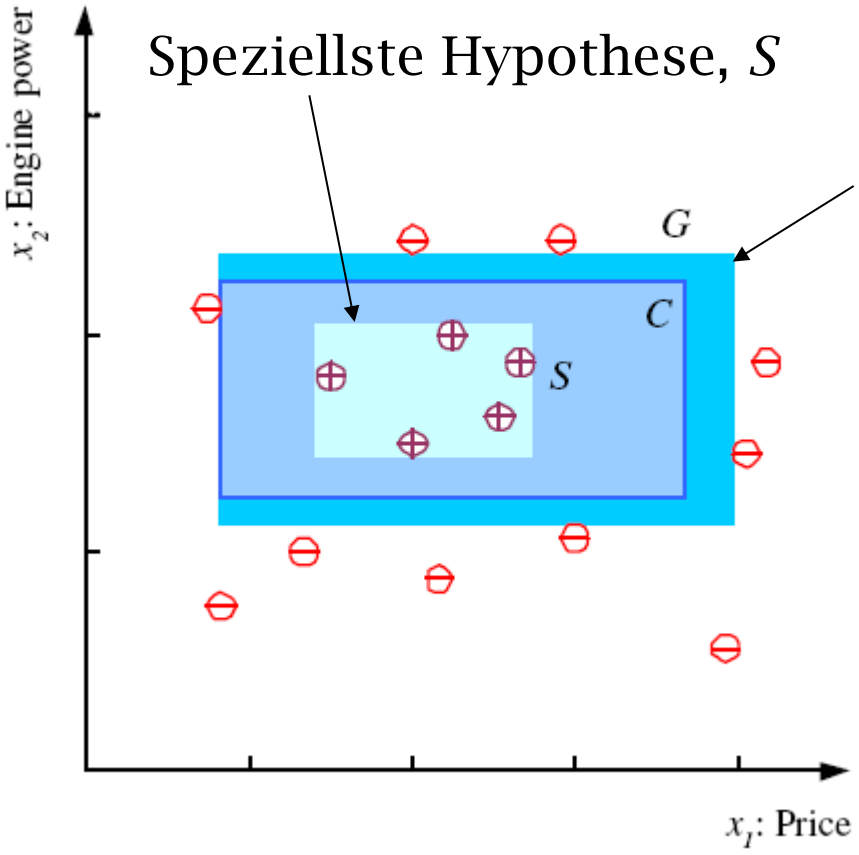
- Obwohl hier die Hypothese nicht mit der Klasse übereinstimmt (also Fehlklassifikationen produzieren wird), ist der empirische Fehler für den konkreten Trainingssatz 0

Generalisierung



Generalisierung ist die Frage, wie gut die Hypothese künftige Beispiele (nicht im Trainingssatz) korrekt einordnen wird

S, G, und der Versionsraum



$h \in \mathcal{H}$, zwischen S und G sind konsistent

und definieren den Versionsraum

(Mitchell, 1997)

Zweifelsfälle

Klassifiziert man neue Beispiele so kann man sagen:

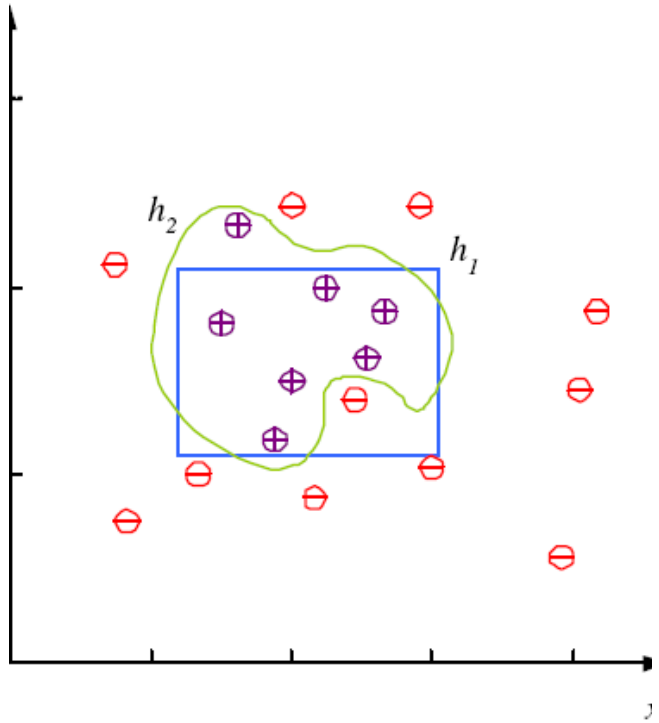
- Jede durch S abgedeckte Instanz ist ein positives Beispiel
- Jede nicht durch G abgedeckte Instanz ist ein negatives Beispiel
- Jede Instanz dazwischen ist ein Zweifelsfall

Setzt voraus, dass die Annahme über die Hypothesenklasse richtig und der Datensatz nicht verrauscht ist

Rauschen und Modellkomplexität

- Ungewollte Anomalien in den Daten werden als Rauschen bezeichnet

- Rauschen erschwert das Lernen
  - Mit einer einfachen Hypothesenklasse ist ein Fehler gleich Null nicht zu erzielen



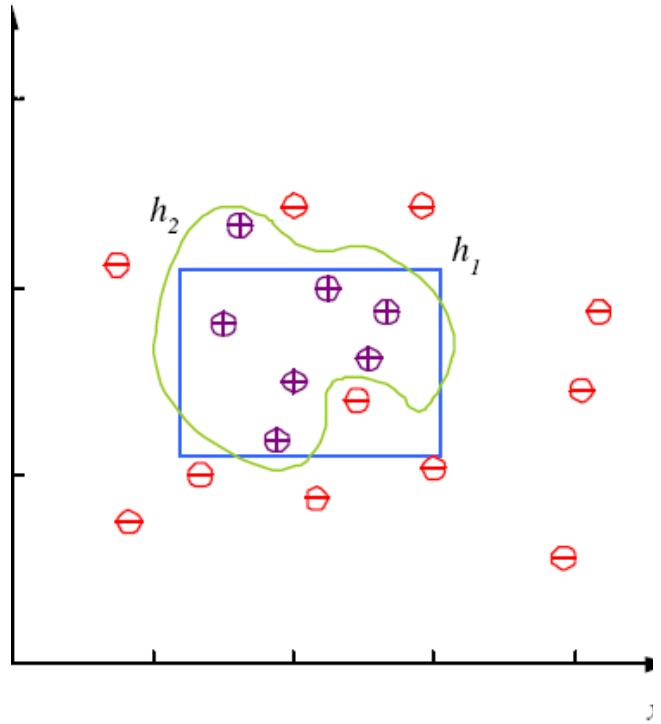
Ursachen für Rauschen

- Ungenauigkeiten bei der Aufnahme der Daten
- Fehler bei einteilen der Daten in positive und negative Beispiele
- Wirklichkeit ist komplexer als das Model
  - verborgene (latente) Attribute die nicht beobachtet werden können
  - grundsätzliche Unregelmäßigkeiten

Rauschen und Modellkomplexität

Auch bei verrauschten Daten macht es sinn einfache Hypothesenklassen zu benutzen

- geringere Berechnungskomplexität für die Erkennung
- einfacheres Training
- besser interpretierbar
- bessere Generalisierung



Modelselektion

- Lernen ist ein unbestimmtes Problem
  - Daten reichen nicht für die Bestimmung einer eindeutigen Lösung

- Um Lernen trotzdem zu ermöglichen müssen Annahmen getroffen werden (z.B. Rechtecke als Hypothesenklasse)
  - solche Annahmen werden als induktive Verzerrung, bezeichnet

- Modellselektion: Auswahl und Grad an induktiver Verzerrung

Generalisierung

- Generalisierung: Wie gut kann das Modell mit Instanzen außerhalb des Trainingssatzes Umgehen

- Überanpassung(Overfitting):
  - $\mathcal{H}$ ist komplexer als C

- Unteranpassung (Underfitting):
  - $\mathcal{H}$ ist weniger komplex als C

Grundlegende Tradeoffs

- Es gibt einen grundlegenden Tradeoff zwischen drei Faktoren (Dietterich, 2003):

1. Komplexität vom $\mathcal{H}$, $c(\mathcal{H})$,
2. Anzahl der Trainingsdaten, N,
3. Generalisierungsfehler E, auf neuen Daten

- ☐ Wenn $N \rightarrow E\downarrow$
- ☐ Wenn $c(\mathcal{H})$, dann zuerst $E\downarrow$ und dann E

Cross-Validation

- Um den Generalisierungsfehler zu bestimmen brauchen wir Daten, die beim Lernen nicht gesehen wurden

- Datensatz wird in 3 Teile geteilt
  - Training set (50%)
  - Validation set (25%) wird iterativ bei der Optimierung des Modells verwendet
  - Test (publication) set (25%) ist die eigentliche Überprüfung des Datensatzes

- Bei Bedarf müssen mehr Daten aufgenommen werden

Confusion Matrix

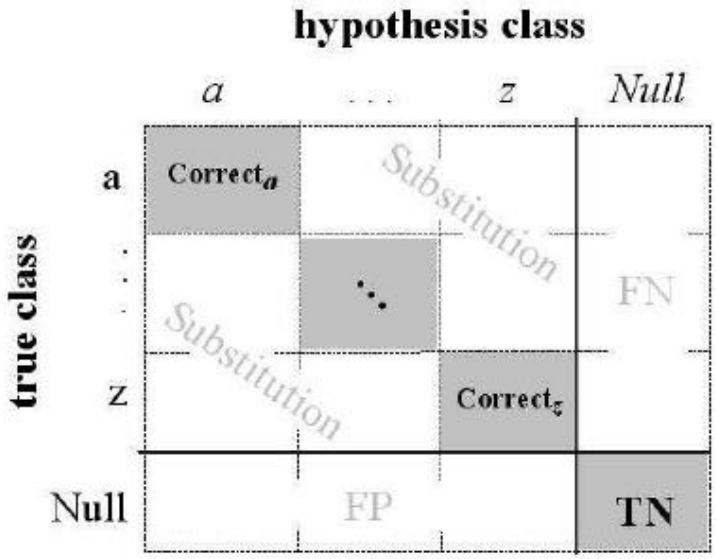


Figure 5.4: *Multi-class confusion matrix: diagonal marks the* correct positive *for positive classes, and True Negative (TN) for* NULL*; off-diagonal marks the positive class substitutions, the sum of the False Positives (FP) and the sum of False Negative (FN) errors*

Evaluation through confusion matrix

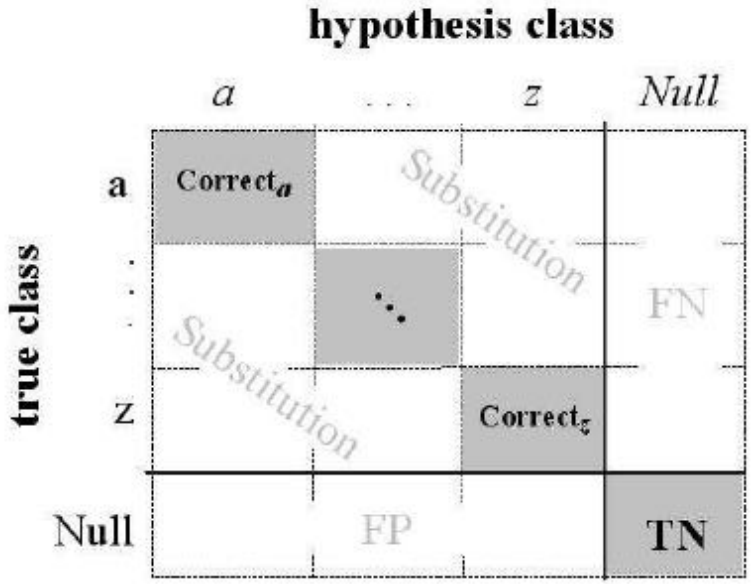
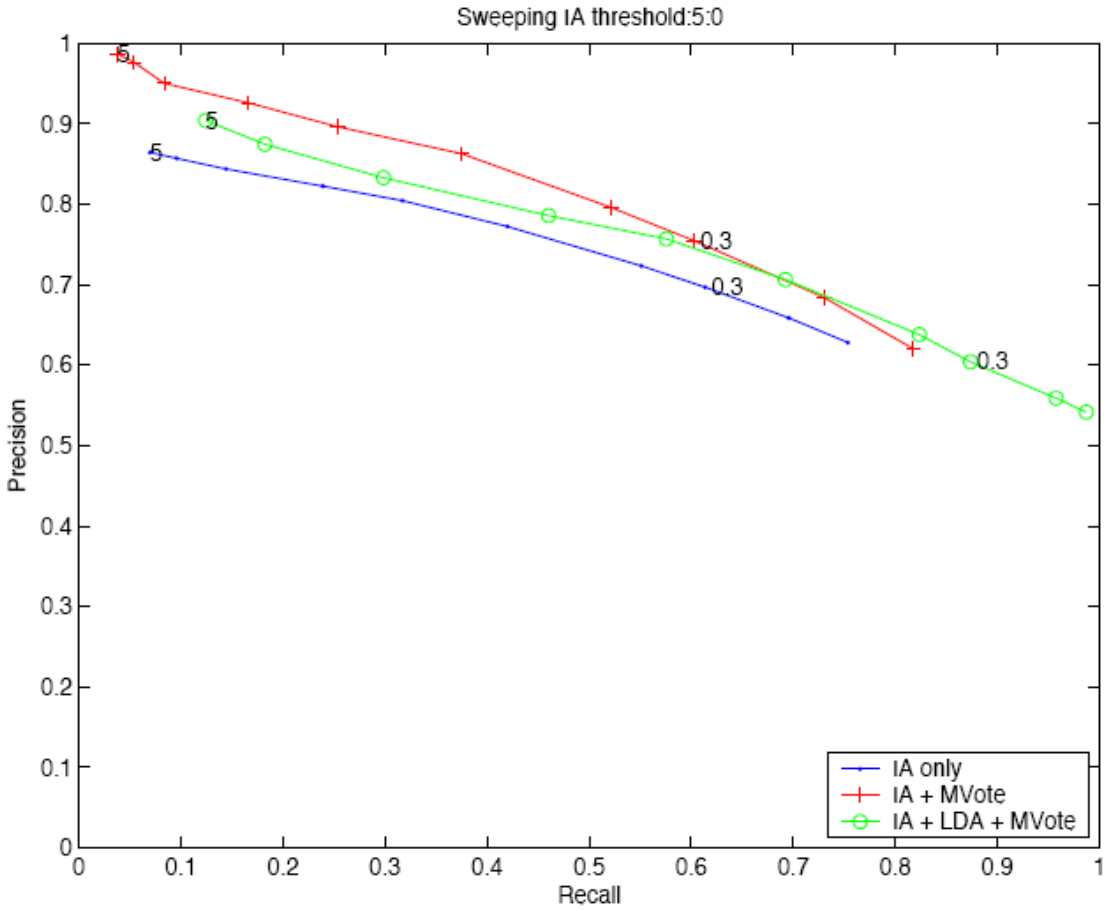


Figure 5.4: *Multi-class confusion matrix: diagonal marks the* correct positive *for positive classes, and True Negative (TN) for* NULL*; off-diagonal marks the positive class substitutions, the sum of the False Positives (FP) and the sum of False Negative (FN) errors*

Precission Recall Graphs

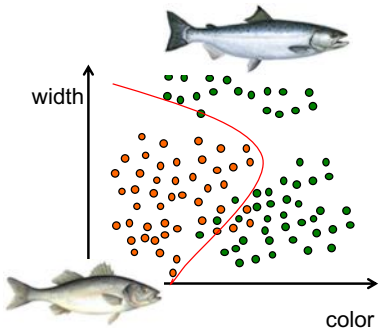


$$recall = \frac{true\ positive\ time}{total\ positive\ time} = \frac{TP}{TP + FN} \tag{5.1}$$

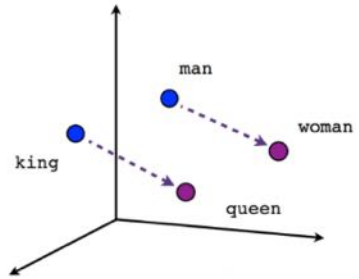
$$precision = \frac{true\ positive\ time}{hypothesized\ positive\ time} = \frac{TP}{TP + FP} \tag{5.2}$$

Result: A target function derived from the sample
to do „something usefull“

on data points that are related to the original space but as such have not been seen before !!!

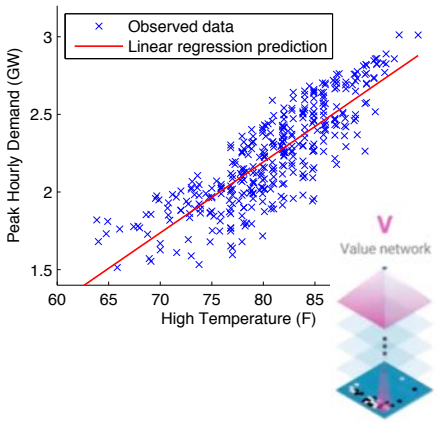


classification



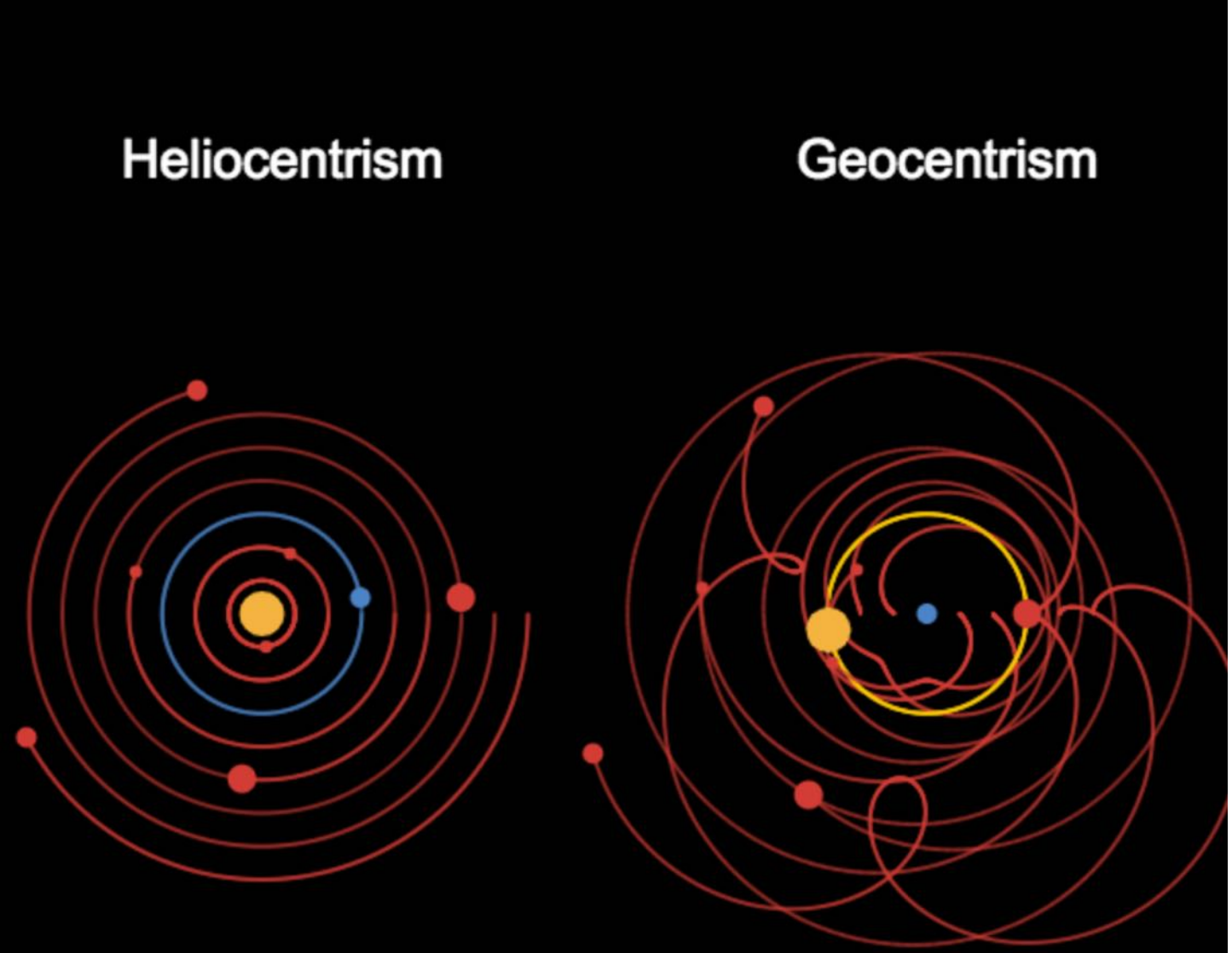
Male-Female

(mapping to)
new representation



value predicton

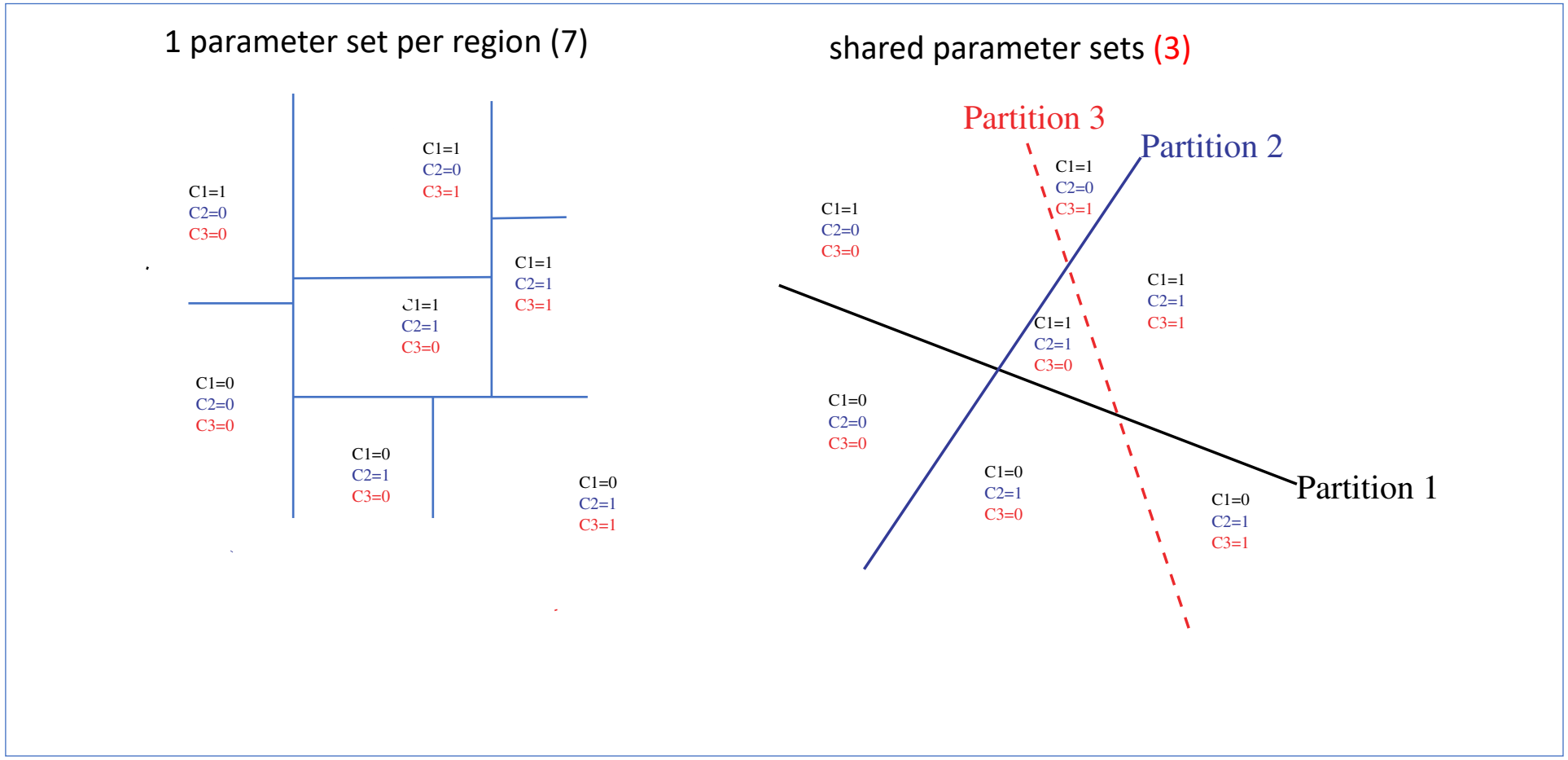
generation of new
samples (completion)



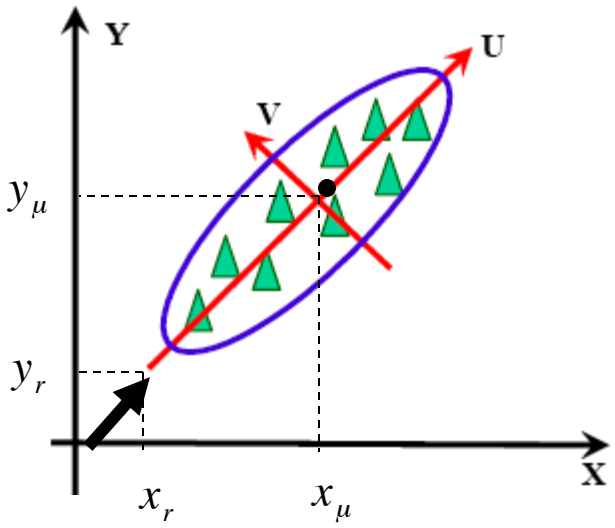
Representations matter

Embedded Intelligence

Reducing the number of parameters: distributed representations



Komponentenanalyse: Prinzip in 2D



- Transformation
  1. Eine Gerade U, gegeben durch einen Mittelpunkt $\vec{\mu} = (x_\mu, y_\mu)$ und einen normalisierten Richtungsvektor $\vec{r} = (x_r, y_r)$
  2. Featureraum (Koordinatensystem) der Dimension m=1 wobei ein Punkt durch die Entfernung u von Punkt m entlang der Gerade U (also als u-faches von $\vec{r}$) gegeben ist
  3. Eine nxm (2x1) Matrix , die einen als (x,y) Koordinaten gegebenen Punkt in das neue Koordinatensystem überführt und als (u) darstellt

Punkte, die nicht auf U liegen werden auf U projiziert (entlang V)!

Principal Component Analysis (PCA)

Die Elemente eines *d* dimensionalen Featureraumes werden auf einen *d'* dimensionalen Raum (*d'*< *d*) projiziert

- Die Abbildung soll so erfolgen, dass die kumulierte Distanz $J_{d'}$ der neuen Vektoren $\vec{x}_k$ zu den alten Vektoren $\vec{x}_k'$ minimiert wird

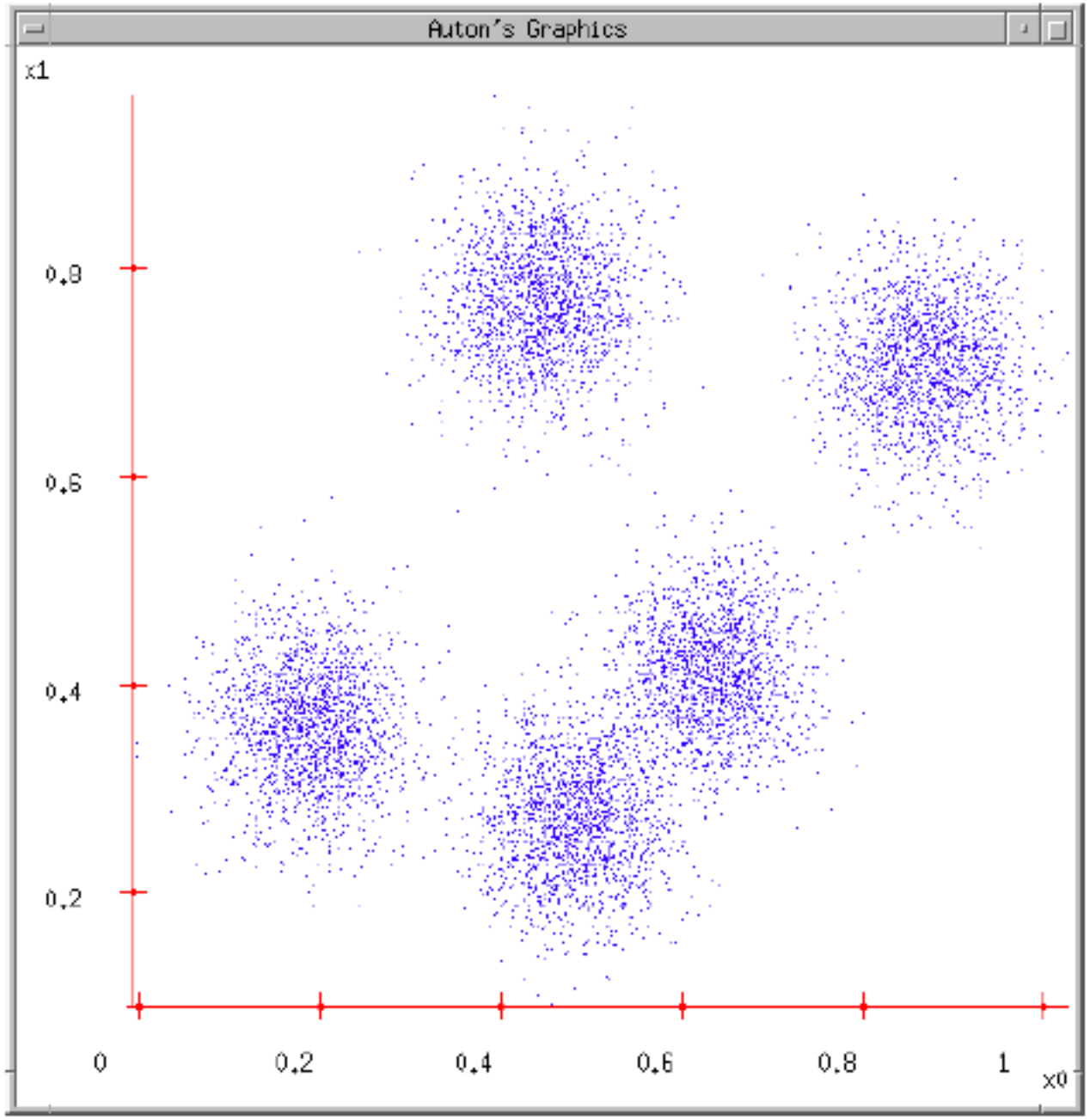
$$J_{d'} = \sum_{k=1}^{n} \left\| \vec{x}_k' - \vec{x}_k \right\|^2$$

- Die Abbildungsparameter werden aus einer Strichprobe von *n* Vektoren $\vec{x}_1 \ldots . \vec{x}_n$ ermittelt.

Struktur der Stichprobe soll möglichst gut erhalten bleiben !

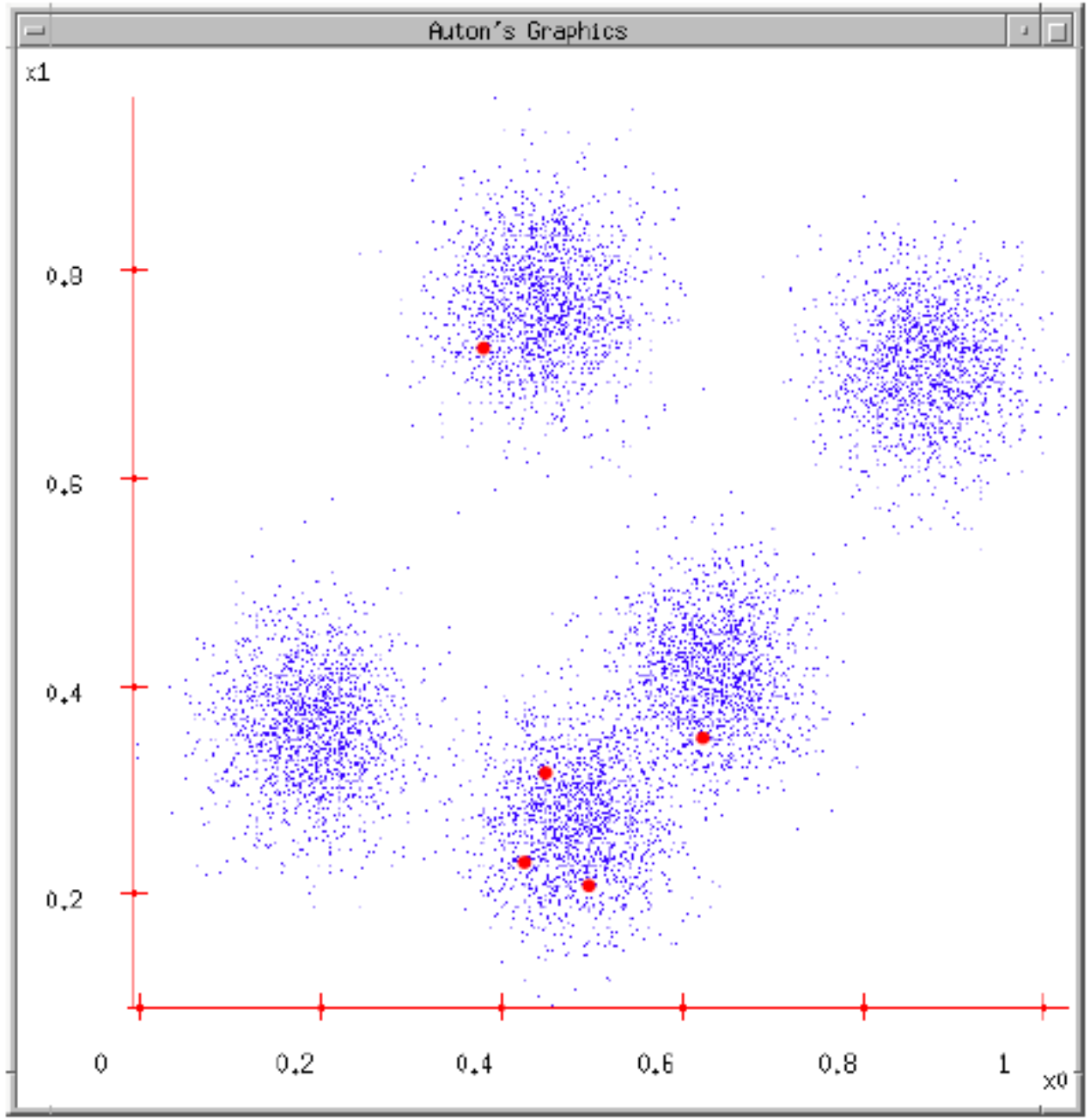
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*



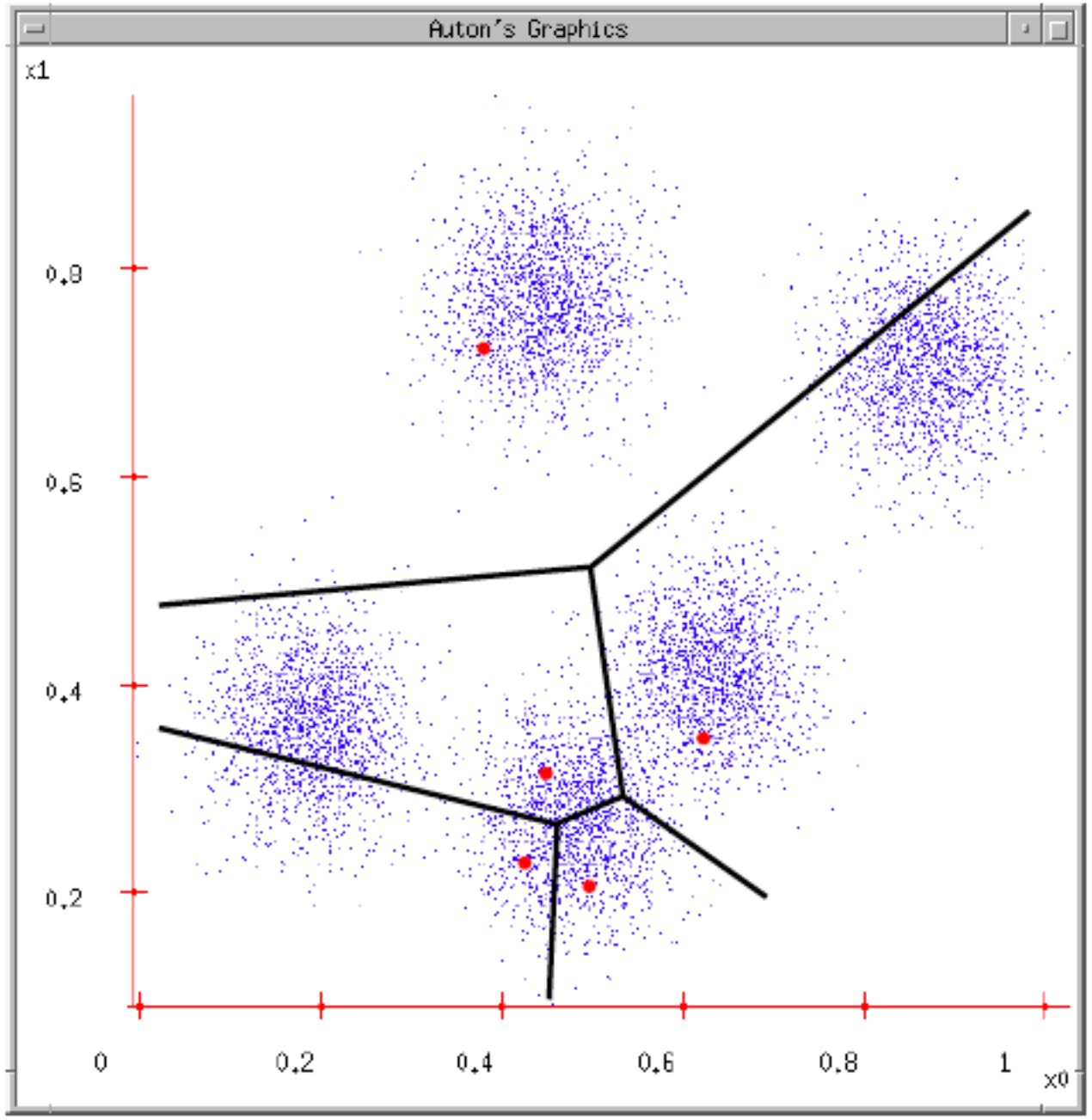
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



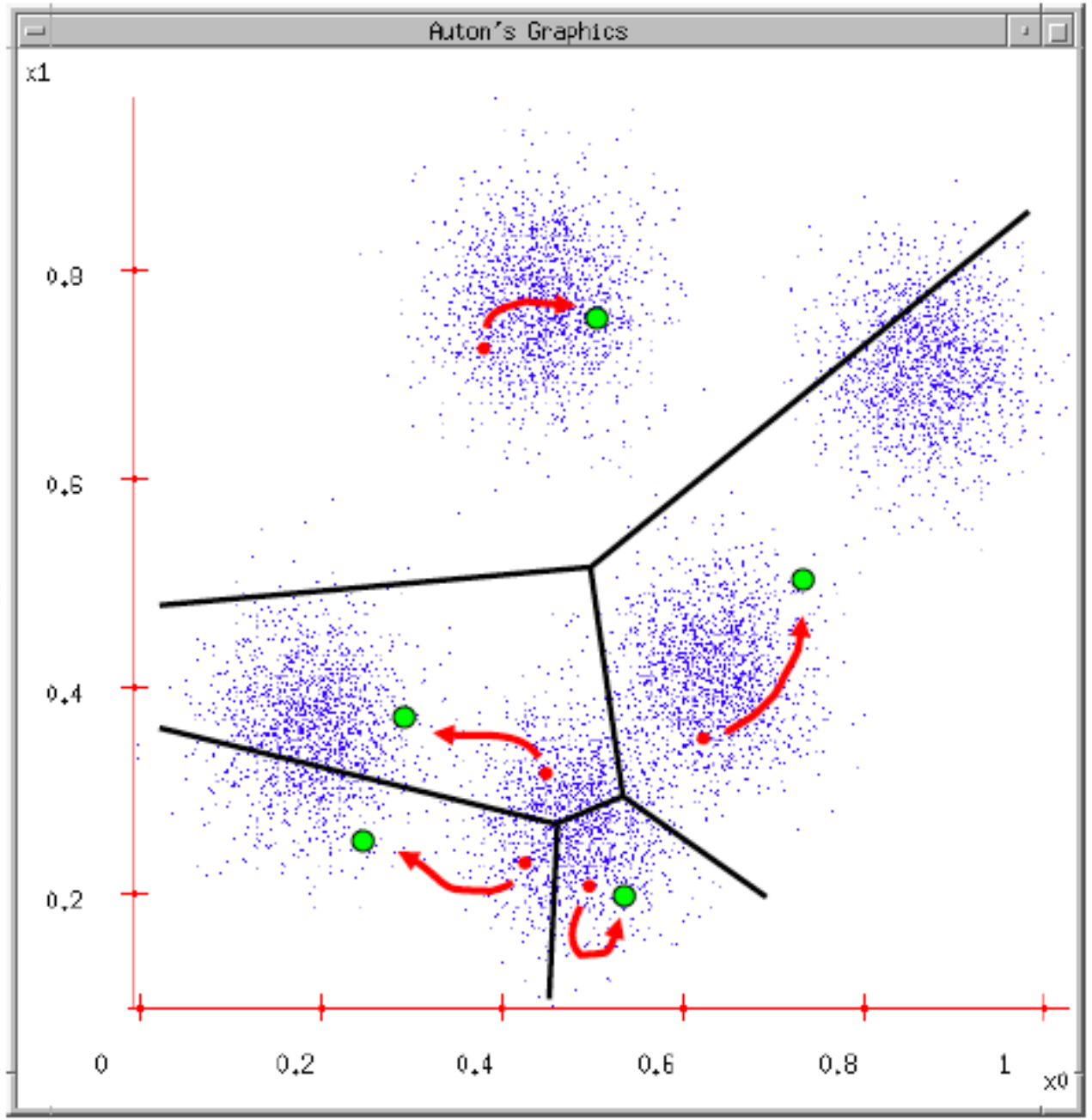
K-means

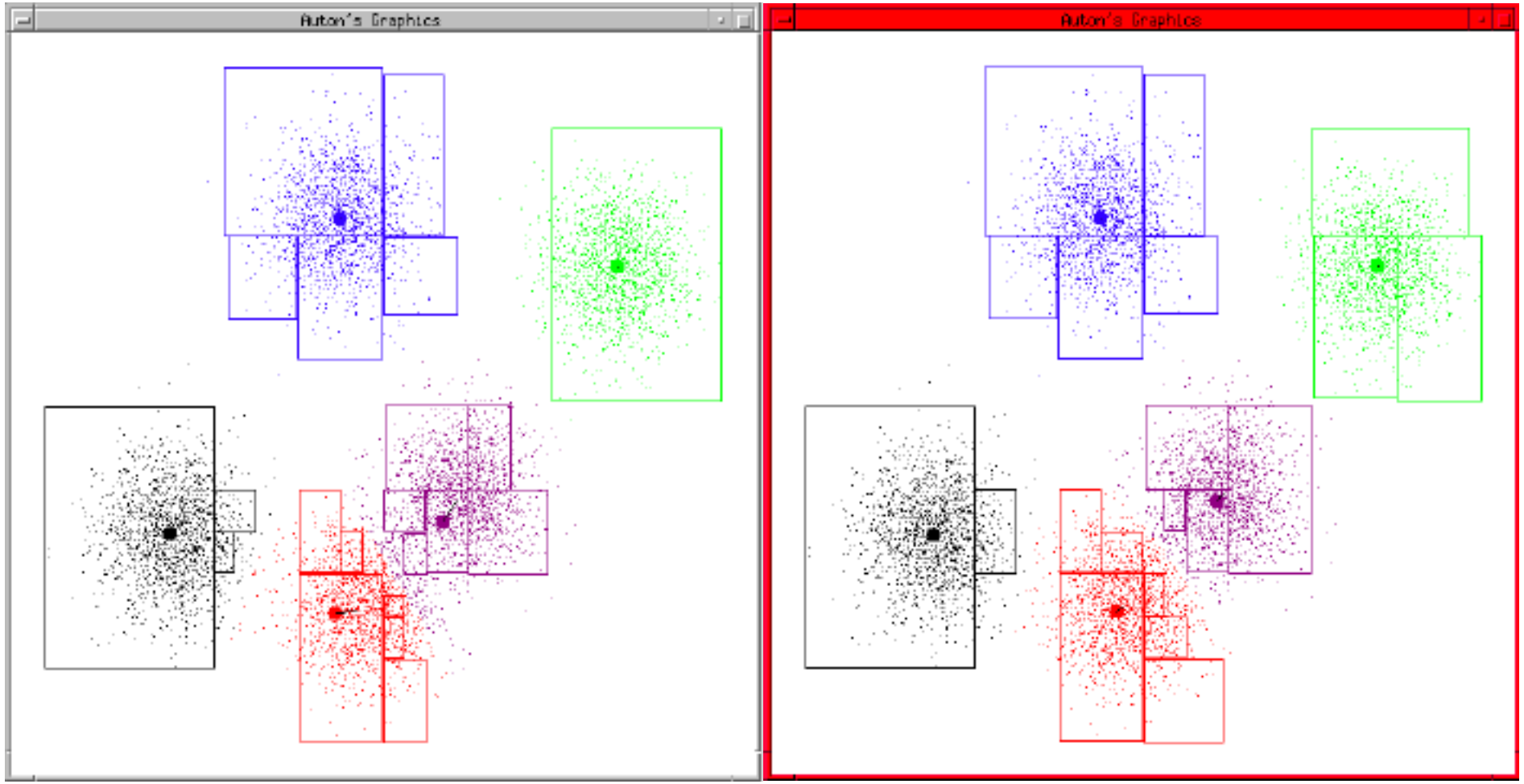
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



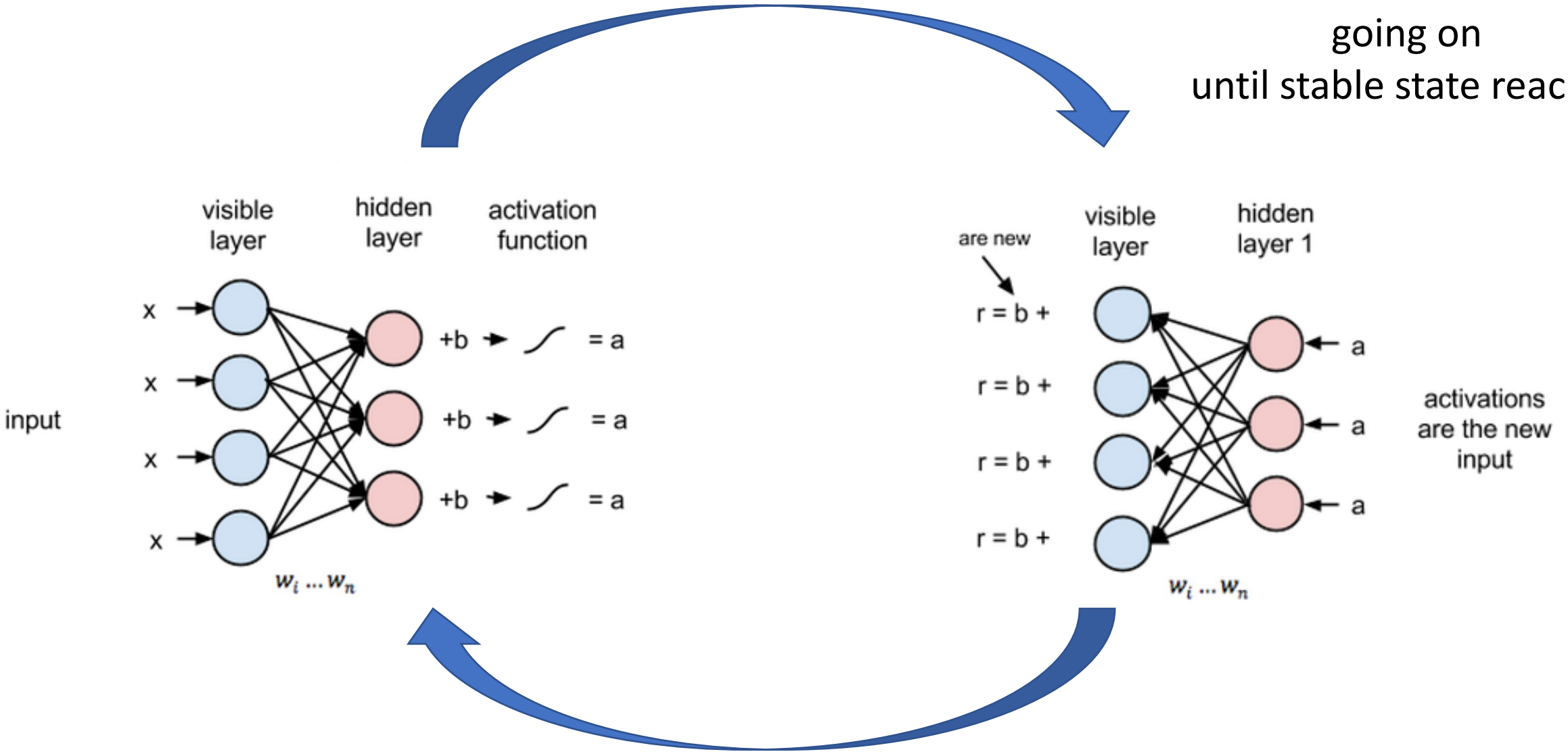

Auton's Graphics
Auton's Graphics

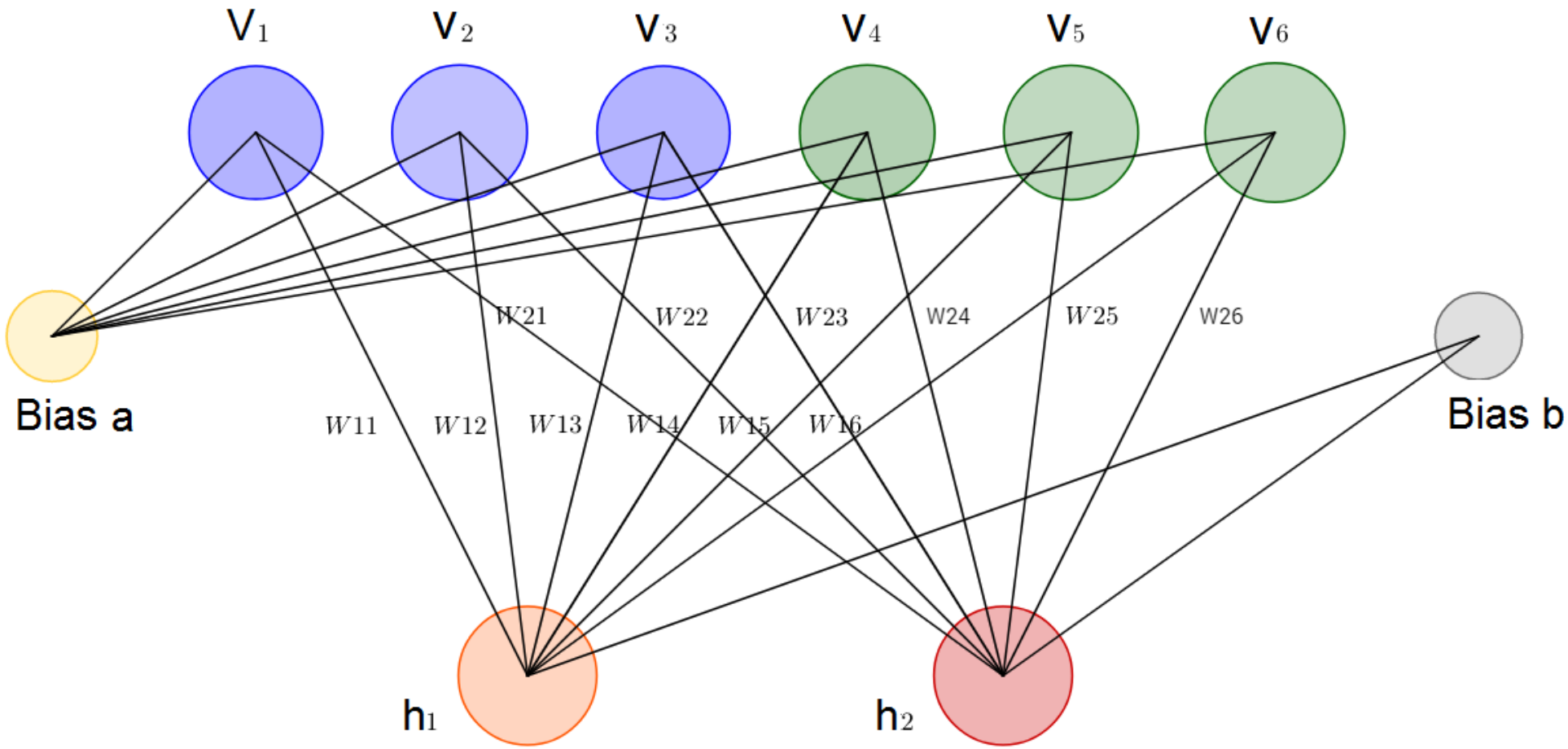
K-means

- If n is the known number of patterns and c the desired number of clusters, the k-means algorithm is:

Begin
initialize n, c, μ1, μ2, …, μc(randomly selected)
do classify n samples according to nearest μi
recompute μi
until no change in μi
return μ1, μ2, …, μc
End

Restricted Boltzman Machines



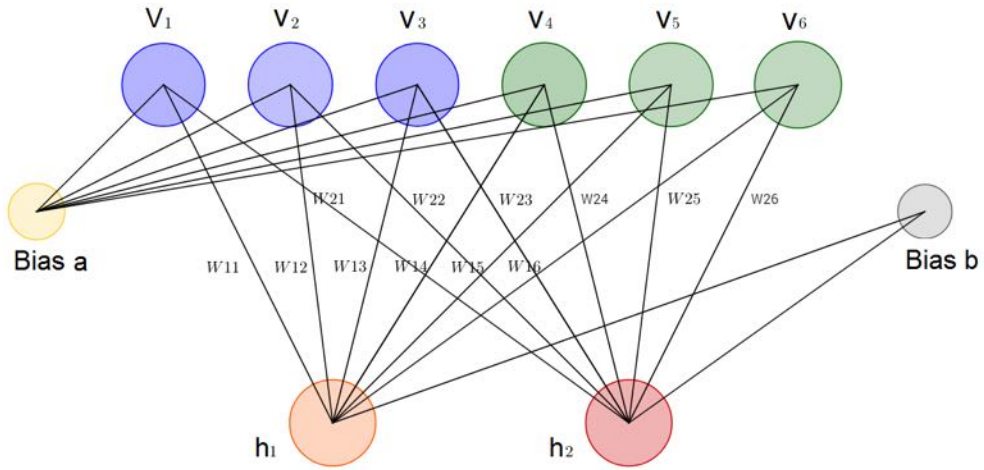


$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

Restricted Boltzman Machines

At each point in time the RBM is in a certain state.

- The state refers to the values of neurons in the visible and hidden layers **v** and **h**.
- The probability that a certain state of **v**and **h** can be observed is given by the following joint distribution:



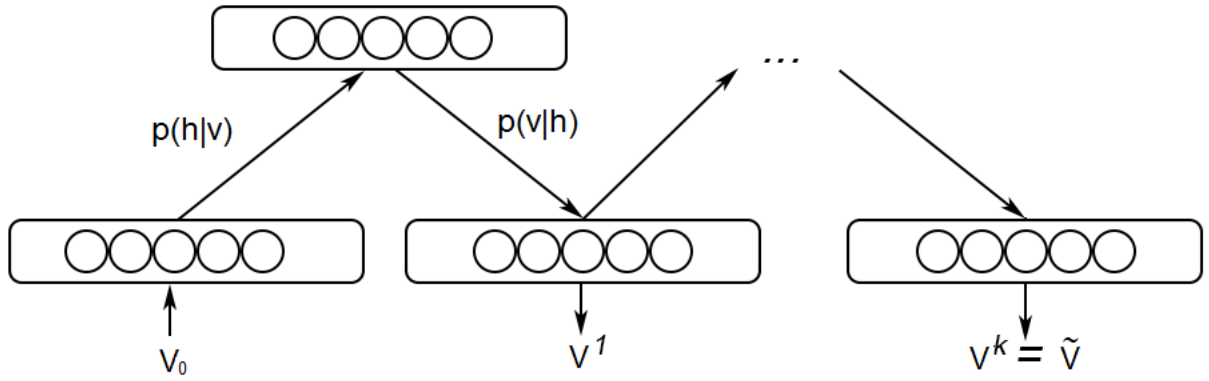
$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}$$

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

RBM learning: Step 1 Gibs Samplng

1. Given an input vector **v** we are using **p(h|v)** for prediction of the hidden values **h.**
2. Knowing the hidden values we use **p(v|h)** for prediction of new input values **v**.
3. This process is repeated *k* times. After *k* iterations we obtain an other input vector **v_k** which was recreated from original input values **v_0**.



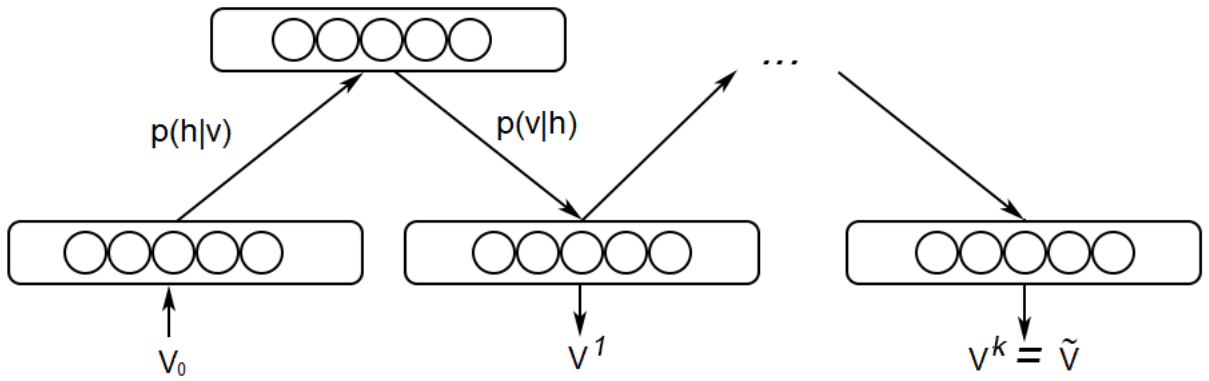
RBM learning: Step 2: *Contrastive Divergence*

- Vectors **v_0** and **v_k** are used to calculate the activation probabilities for hidden values **h_0** and **h_k** (Eq.4).
- The difference between the outer products of those probabilities with input vectors **v_0** and **v_k** results in the update matrix:

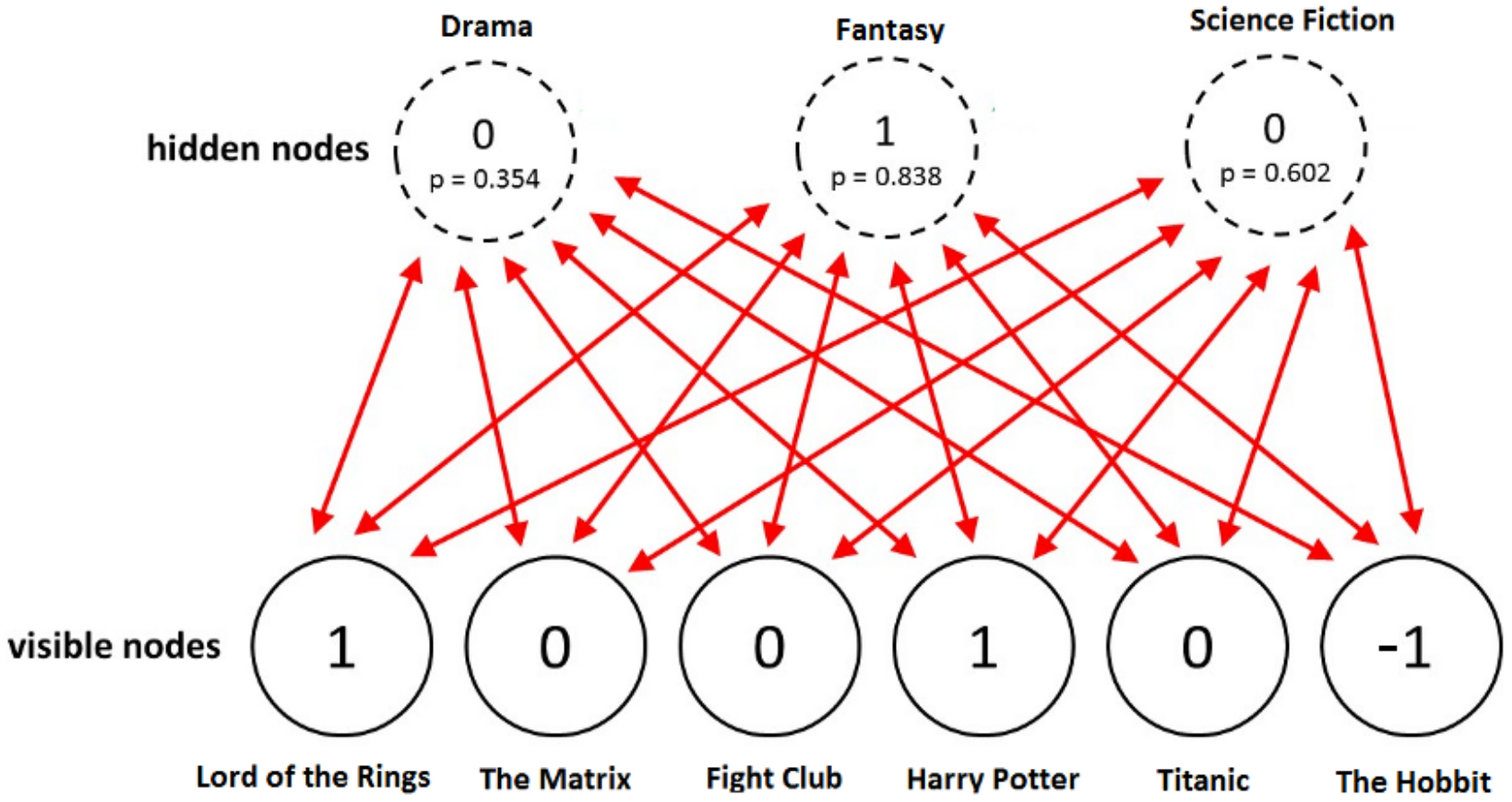
$$\triangle W = \mathbf{v}_0 \otimes p(\mathbf{h}_0|\mathbf{v}_0) - \mathbf{v}_k \otimes p(\mathbf{h}_k|\mathbf{v}_k)$$

- the new weights can be calculated with gradient **ascent,** given by:

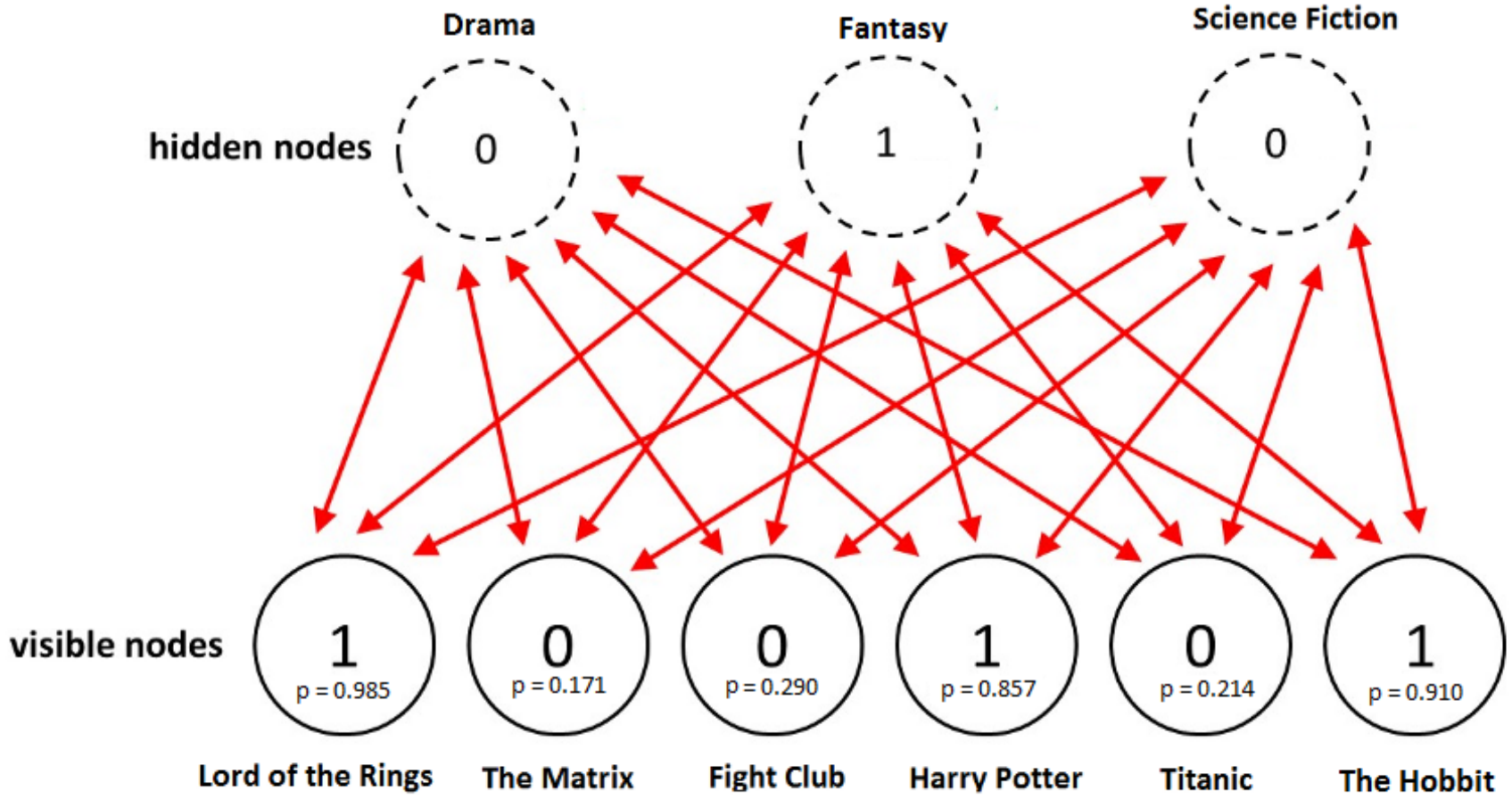
$$W_{new} = W_{old} + \triangle W$$



Example Application

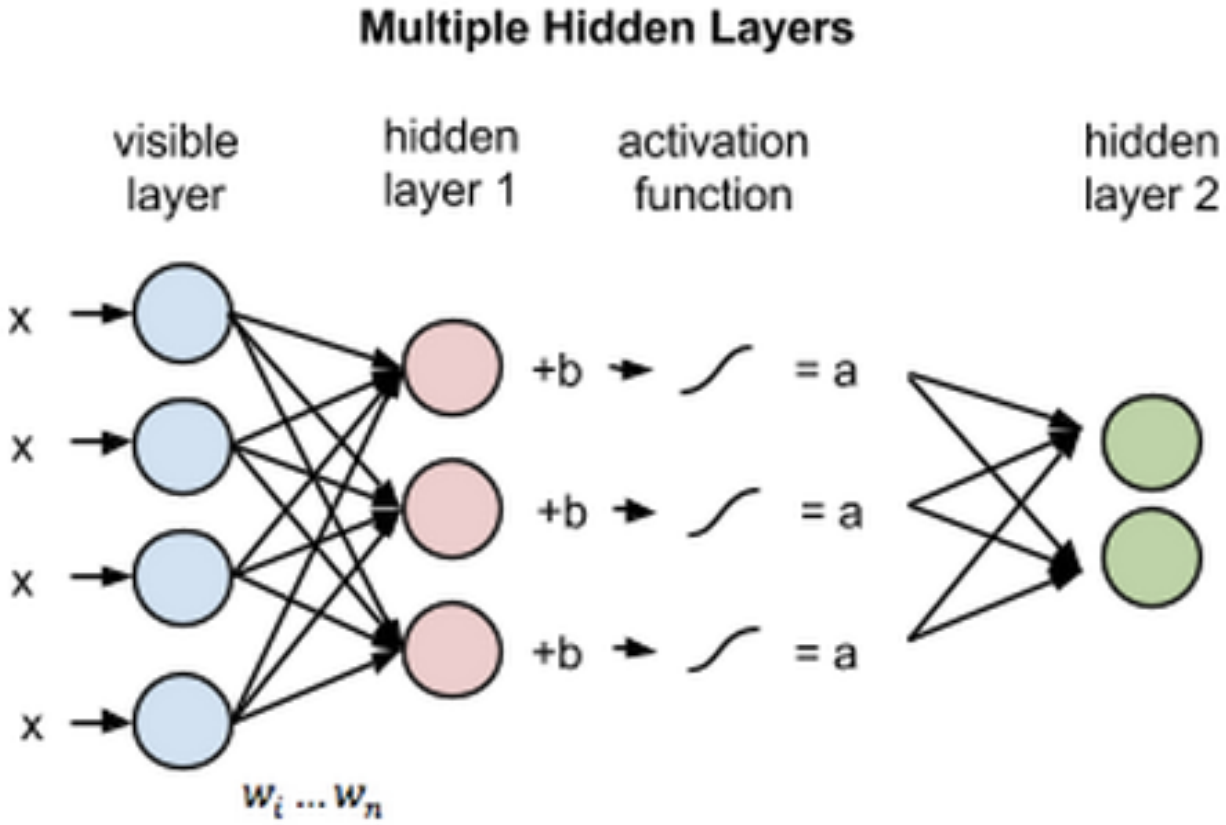


Example Application



Restricted Boltzman Machines

If these two layers were part of a deeper neural network, the outputs of hidden layer no. 1 would be passed as inputs to hidden layer no. 2, and from there through as many hidden layers as present

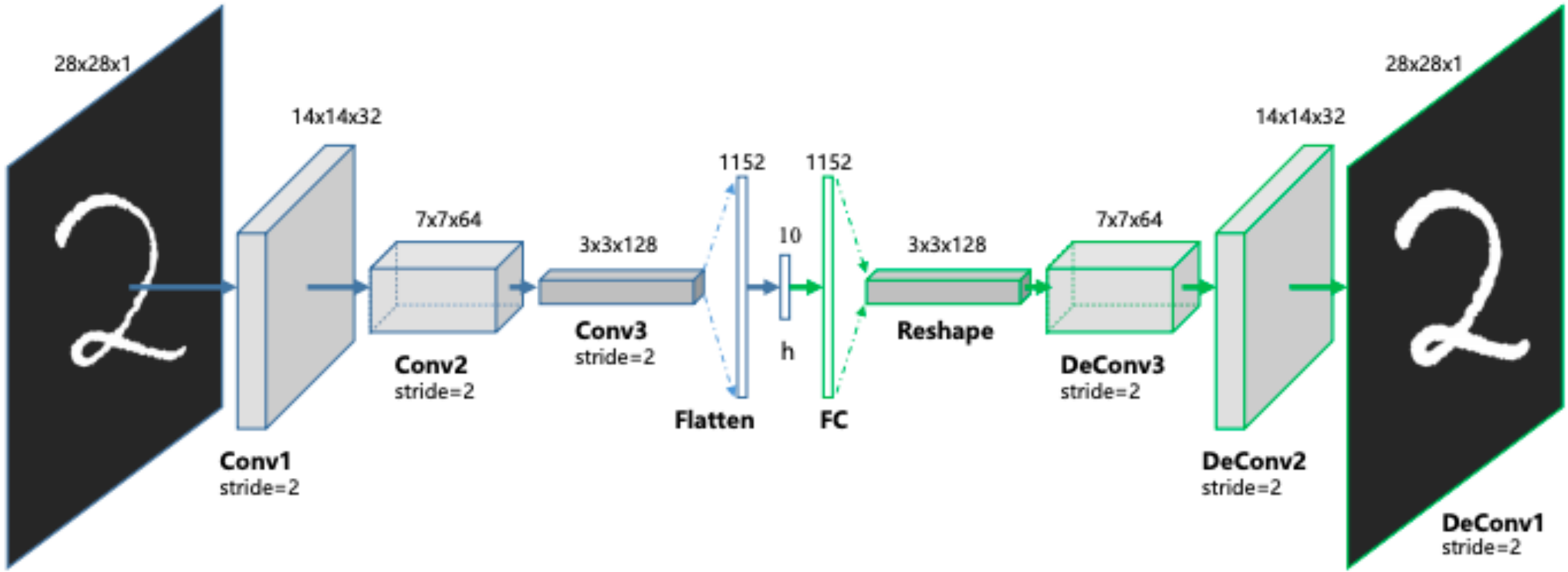


- building blocks of deep belef nets

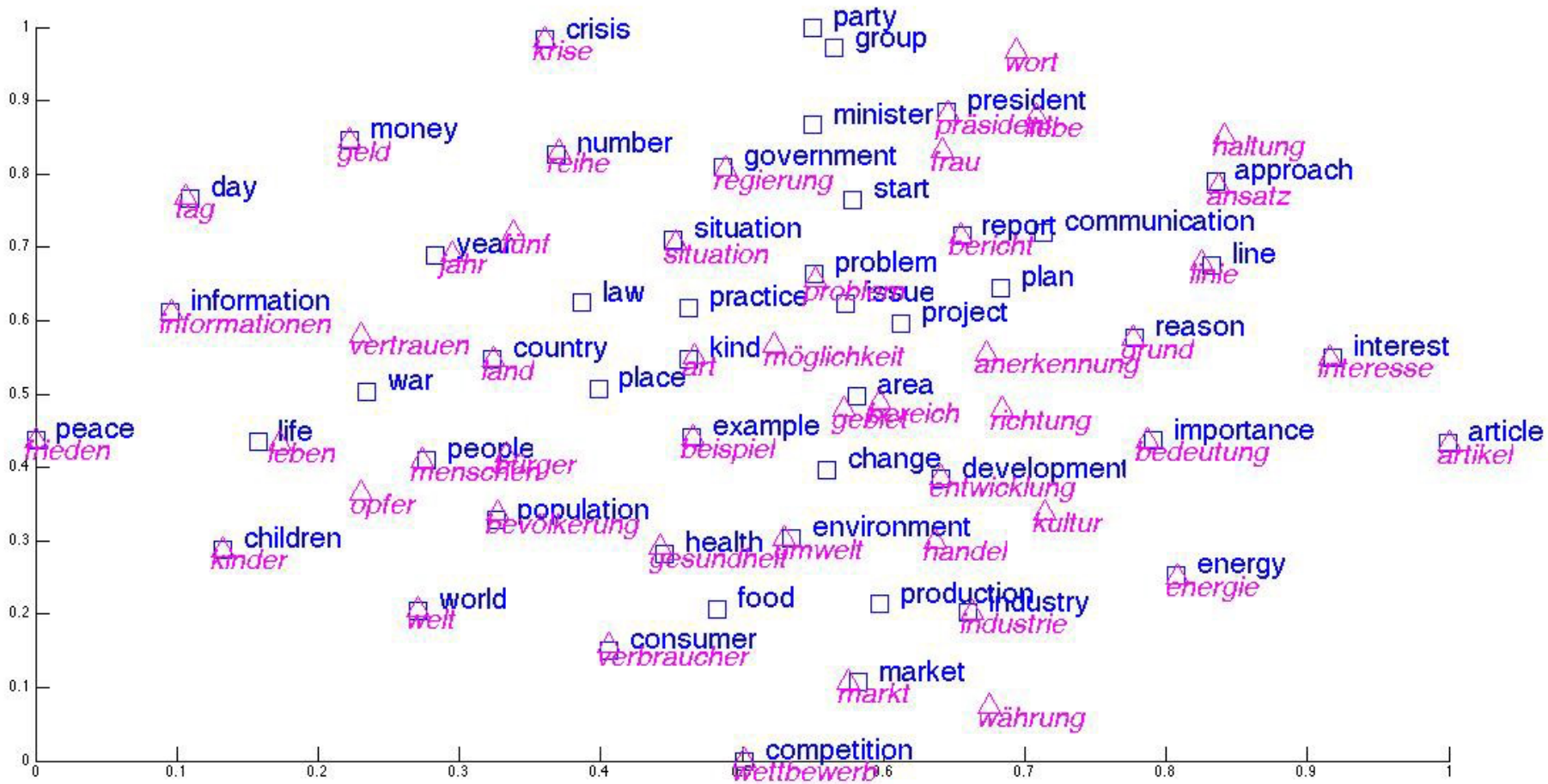
Deep Belief Nets

Some more deep represenation learning wth “old friends”…

A related approach: Convolution Autoencoder



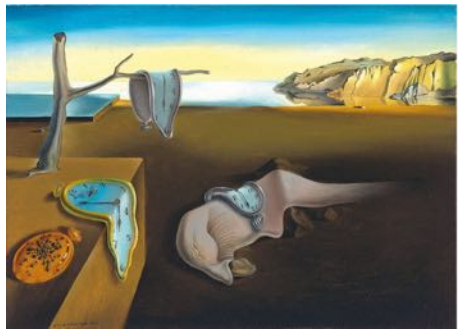
Example of complex deep representation learning: Word Embeddings

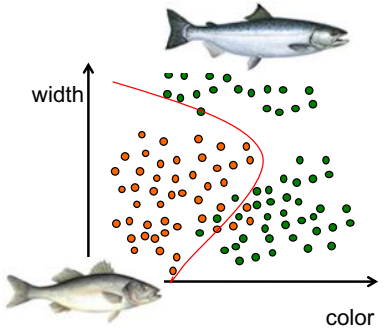


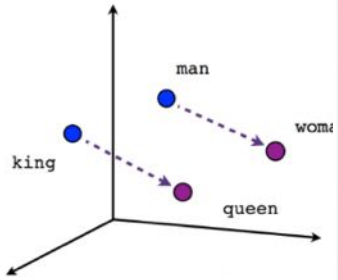
Embedded Intelligence

Result: A target function derived from the sample
to do „something usefull“

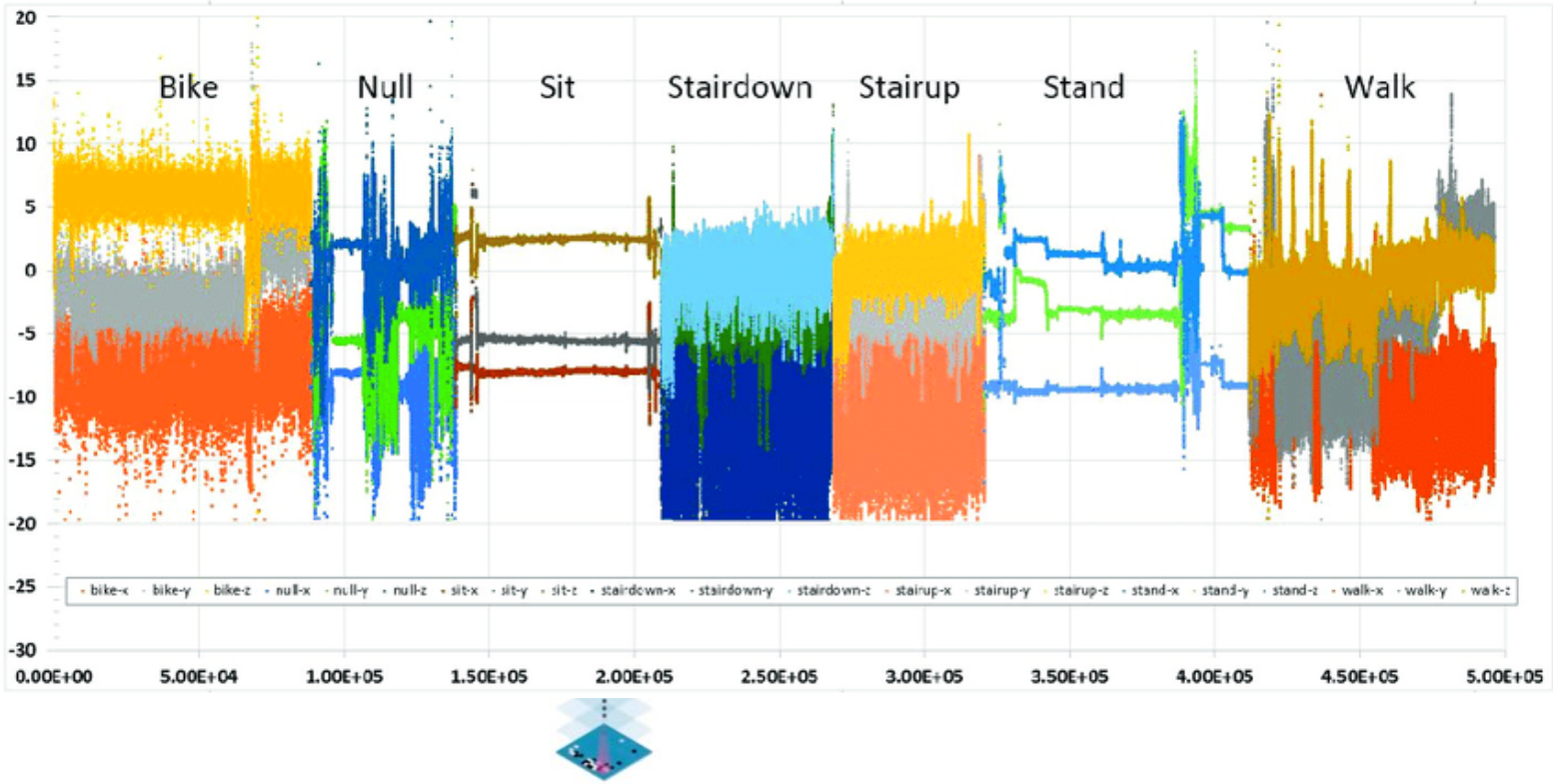
What about time ?







Male-Female



classification

(mapping to)
new representation

value predicton

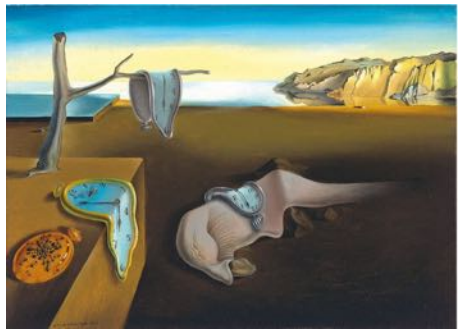
generation of new
samples (completion)

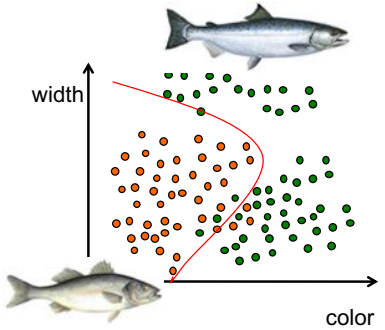
Technische Universität Kaiserslautern

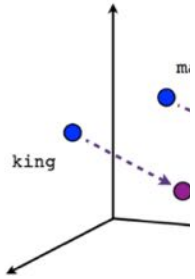
Embedded Intelligence

Result: A target function derived from the sample
to do „something usefull“

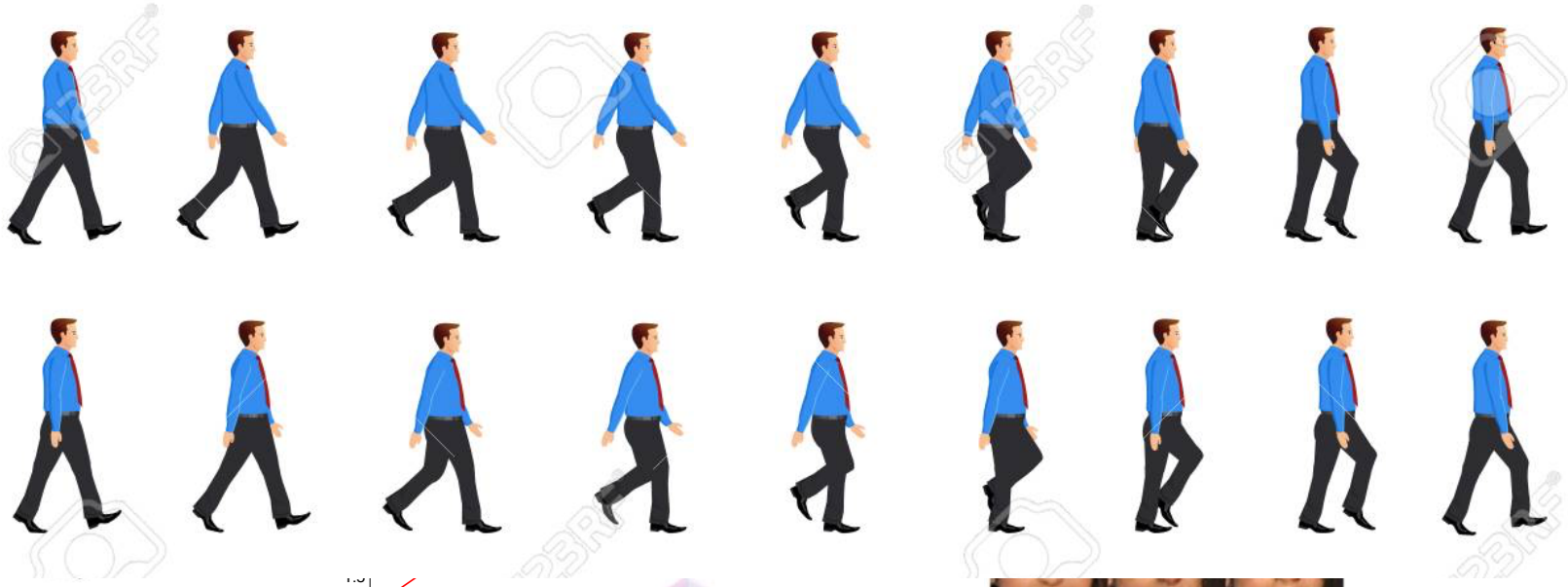
What about time ?

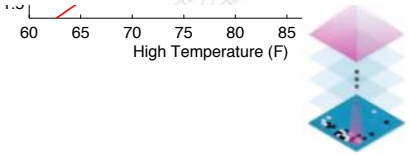






Male-Female





classification

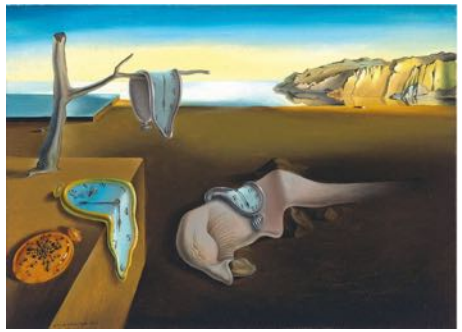
(mapping to)
new representation

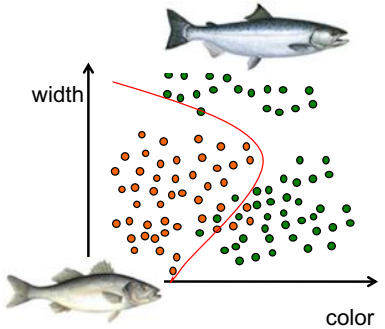
value predicton

generation of new
samples (completion)

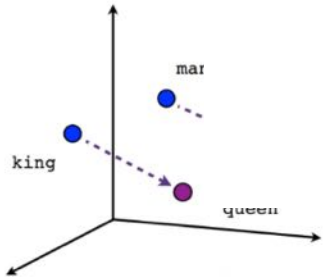
Result: A target function derived from the sample
to do „something usefull“

What about time ?





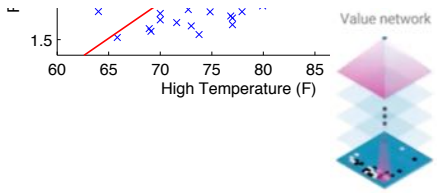
classification



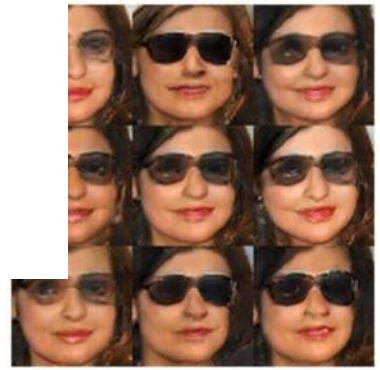
Male-Female

(mapping to)
new representation

Google Translate



value predicton

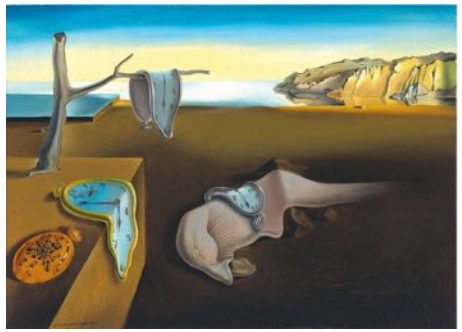


generation of new
samples (completion)

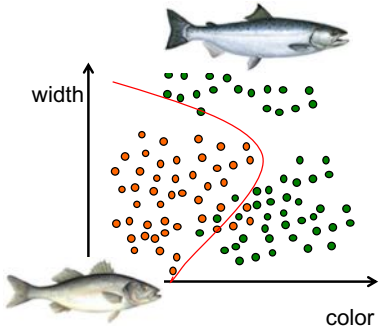
Result: A target function derived from the sample to do „something usefull“

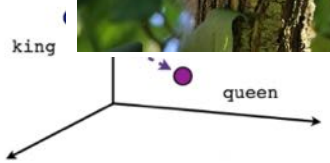
What about time ?

image captioning

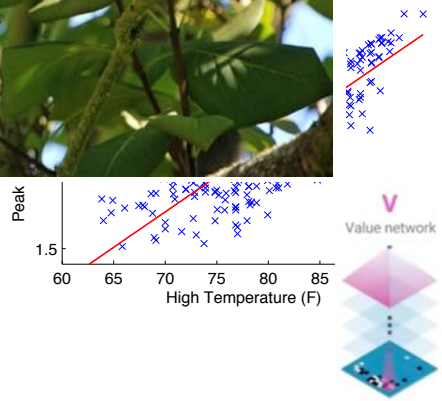


A cat is sitting on a tree branch





Male-Female



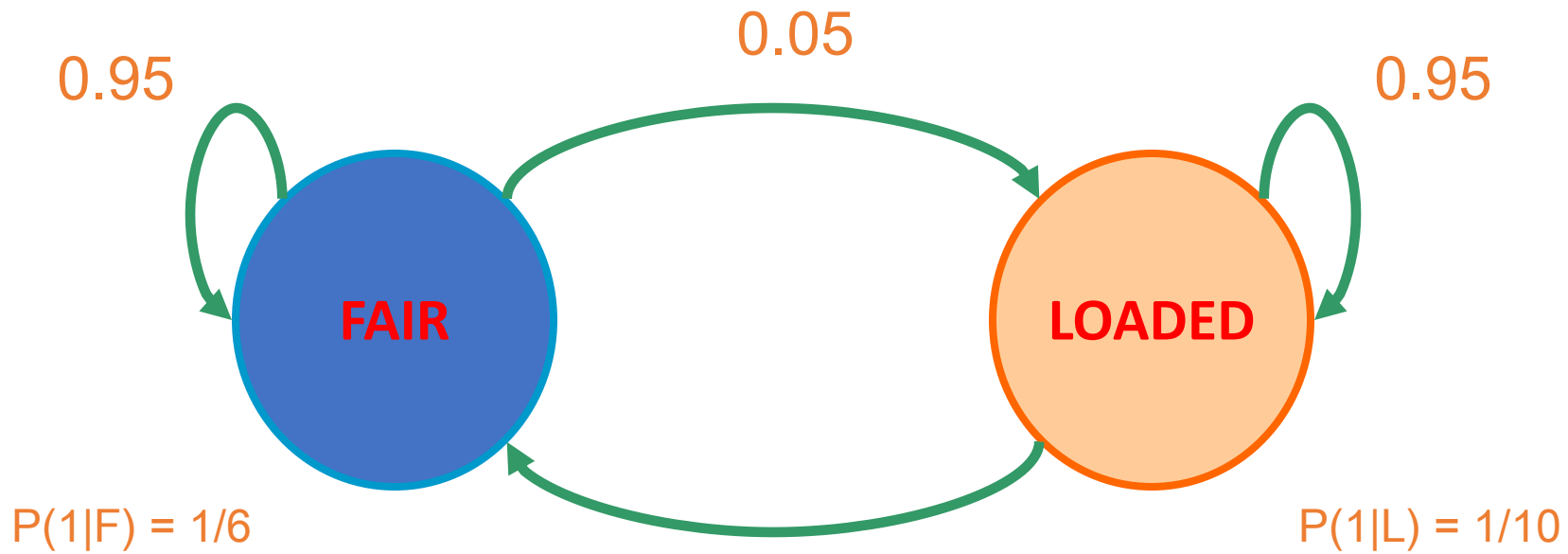
classification

(mapping to)
new representation

value predicton

generation of new
samples (completion)

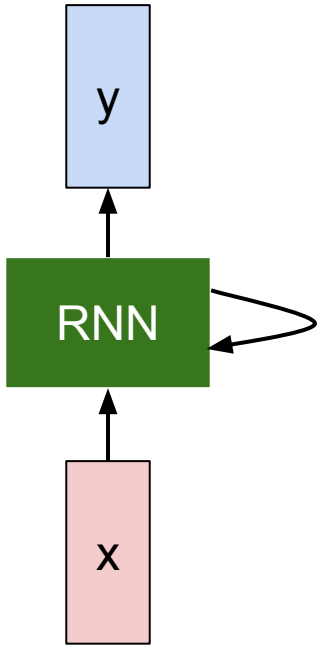
The dishonest casino model



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

Recurrent Neural Network Cell

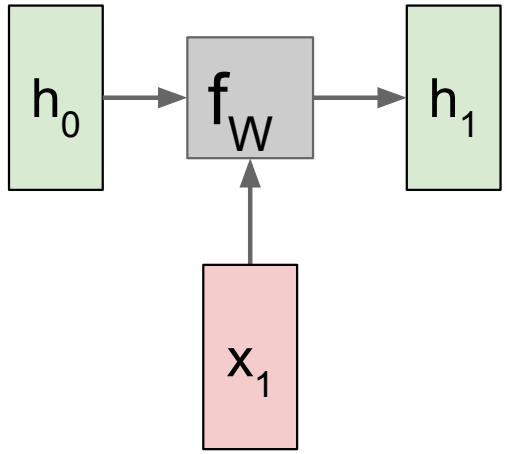


$$h_t = f_W(h_{t-1}, x_t)$$

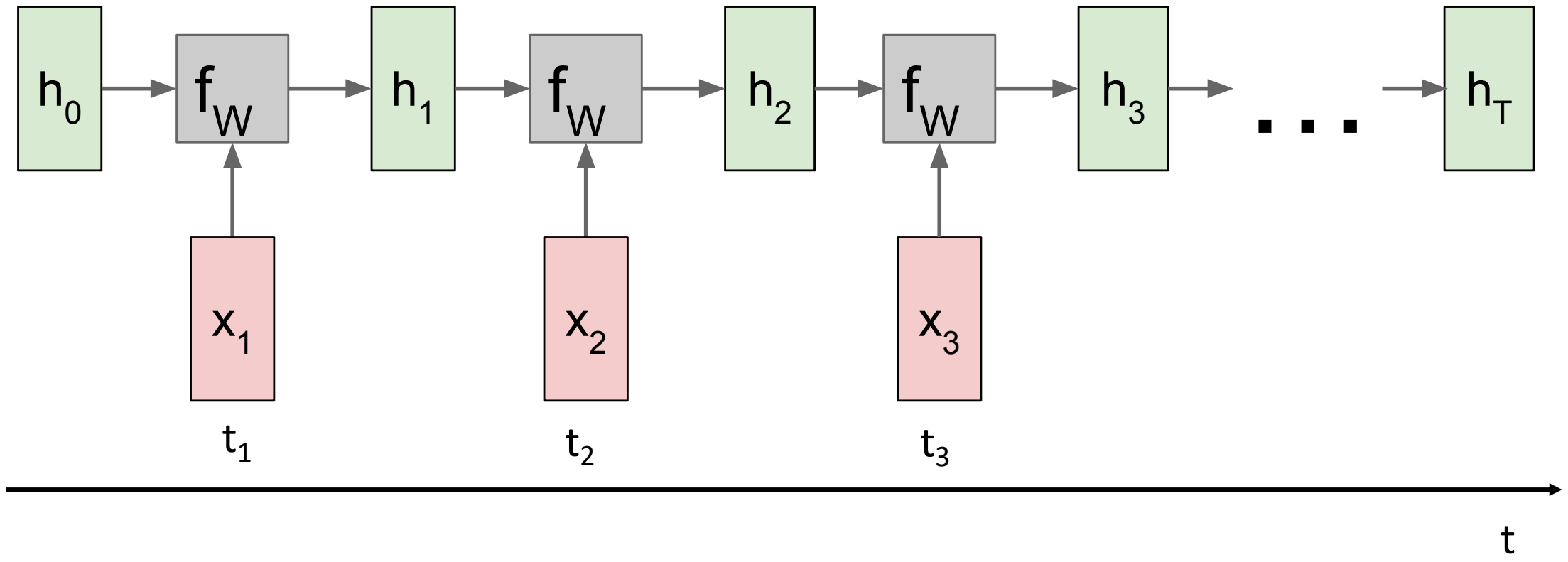
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

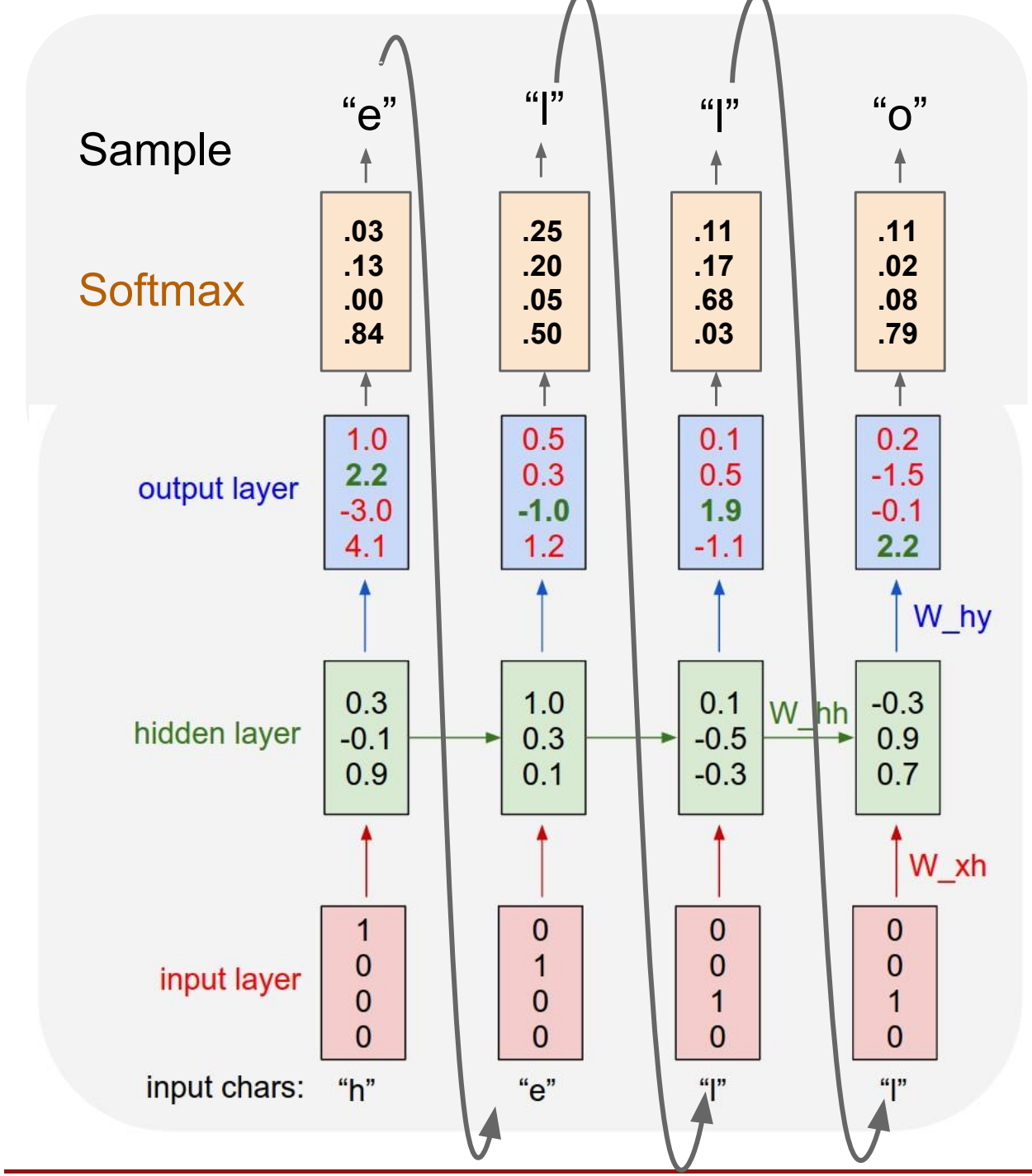
"computational graph"



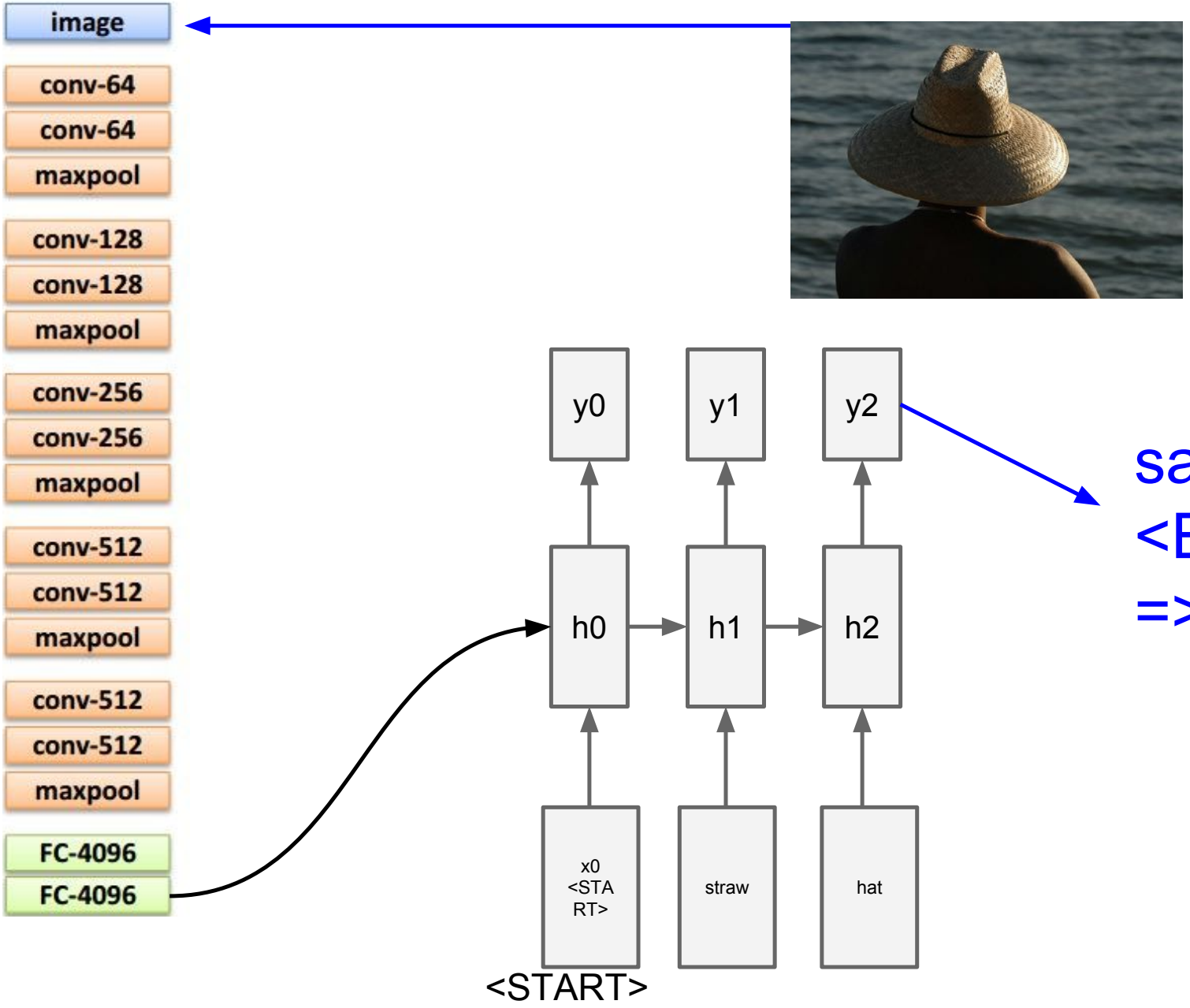
Time „unrolling“



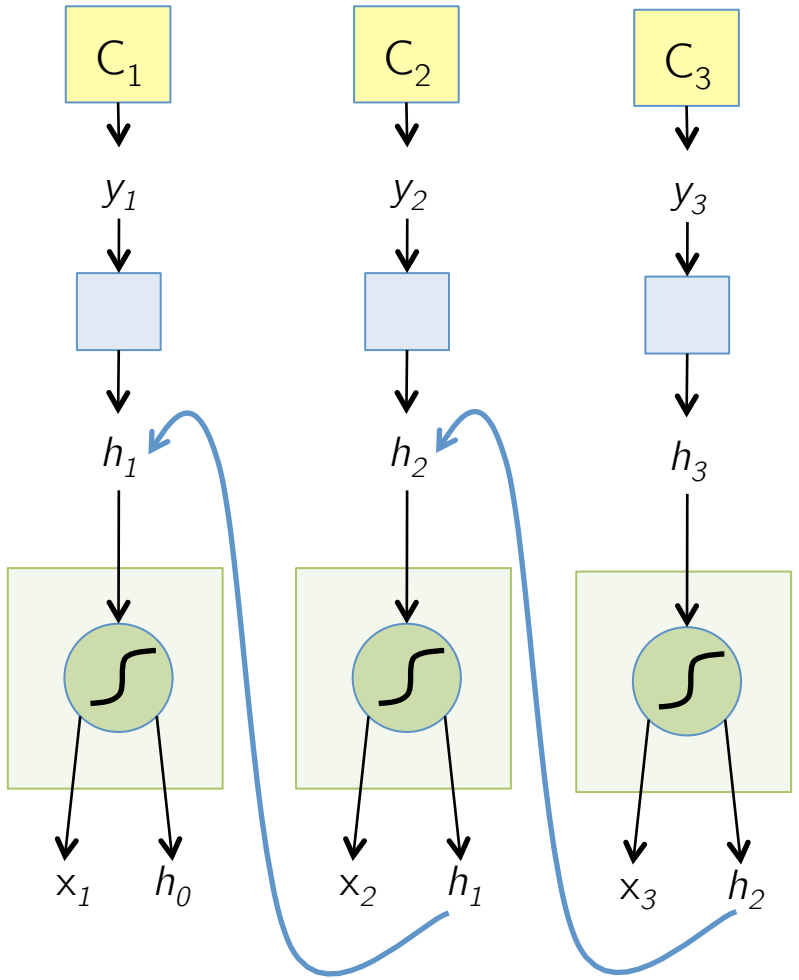
Example: Character Prediction/Word Generation



RNNs +CNNs: Image Captioning



Computing the gradient in an RNN



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

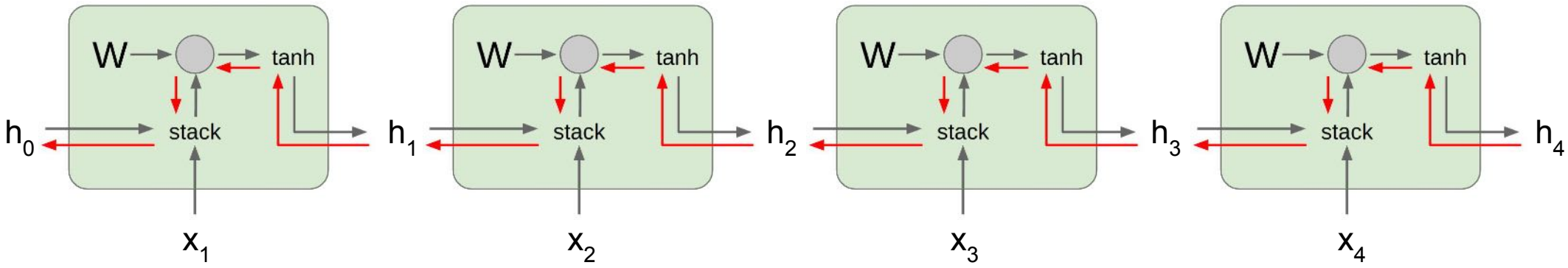
$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)$$

Computing the gradient in an RNN



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Problems with gradient computation

- In the same way a product of k real numbers can shrink to zero or explode to infinity, so can a product of matrices

- It is sufficient for $\lambda_1 < 1/\gamma$, where λ_1 is the largest singular value of W, for the **vanishing gradients** problem to occur and it is necessary for **exploding gradients** that $\lambda_1 > 1/\gamma$, where $\gamma = 1$ for the tanh non-linearity and $\gamma = 1/4$ for the sigmoid non-linearity [1]

- Exploding gradients are often controlled with gradient element-wise or norm clipping

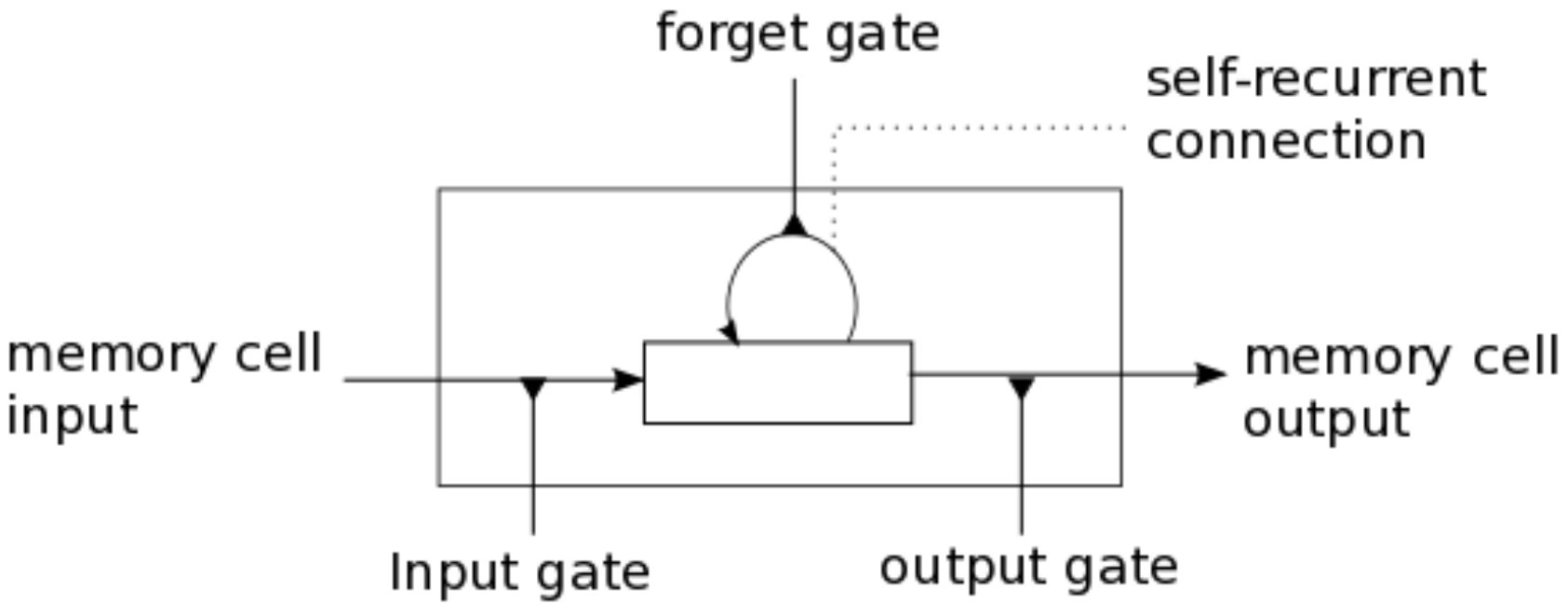
Solution: Long Short Term Memory 8LSTM)

- The LSTM uses so called “Constant Error Flow” for RNNs to ensures that gradients don’t decay!
  - (expalanation not part of this lecture !)

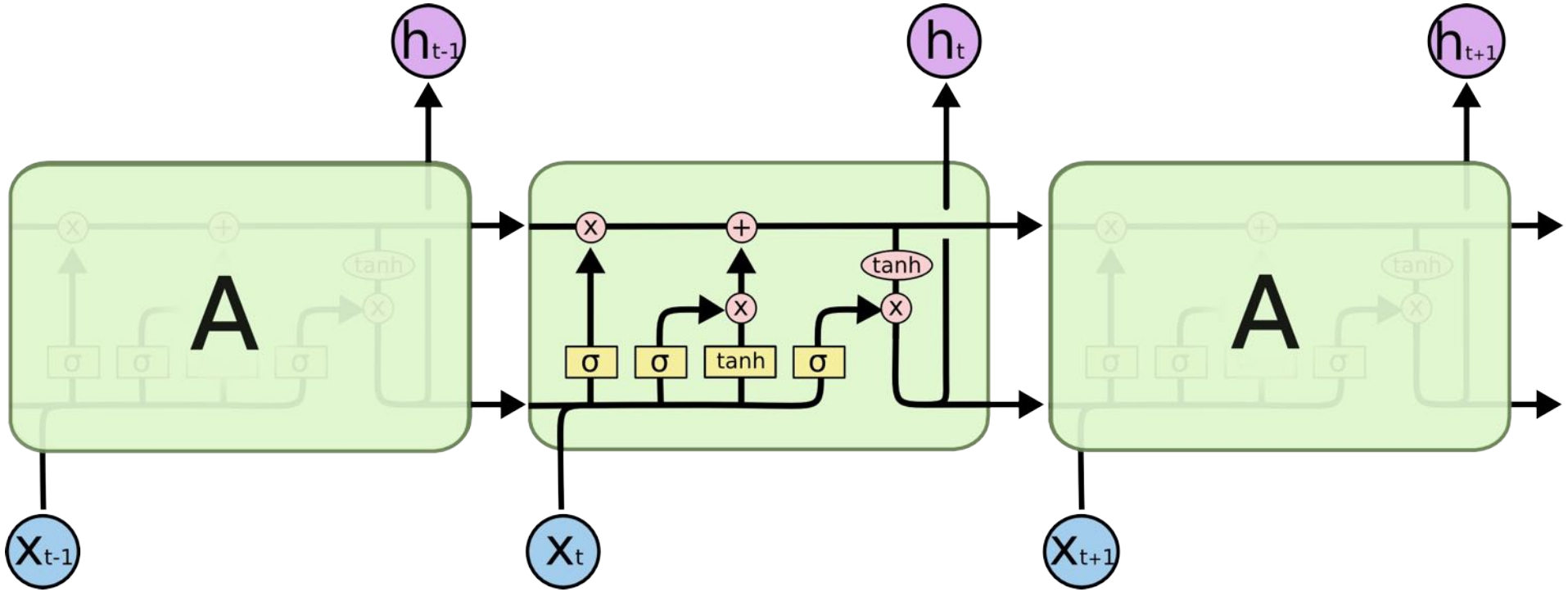
- The key component is a memory cell that acts like an accumulator (contains the identity relationship) over time!

- Expressivity is the same, but gradients are better behaved!

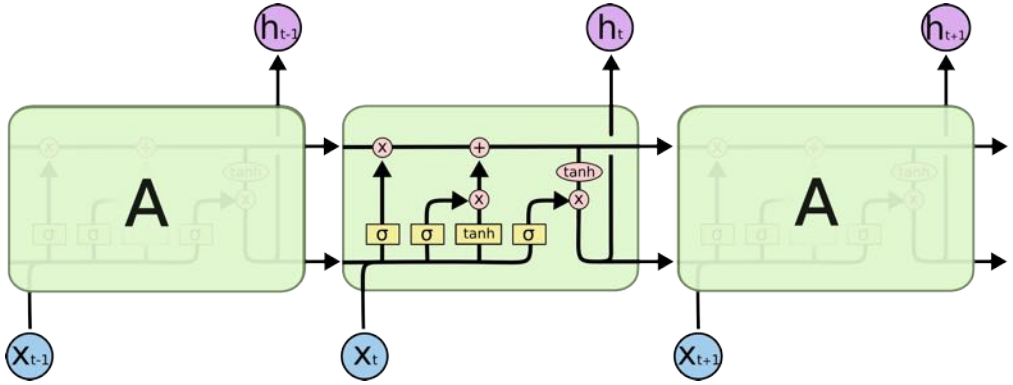
LSTM Idea

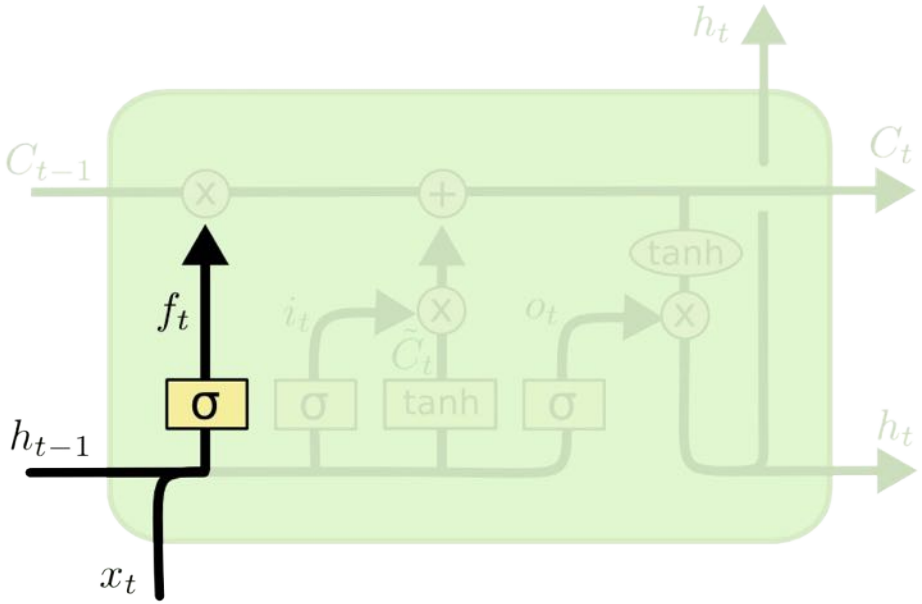

forget gate
self-recurrent
connection
memory cell
input
memory cell
output
Input gate
output gate

LSTM in detail



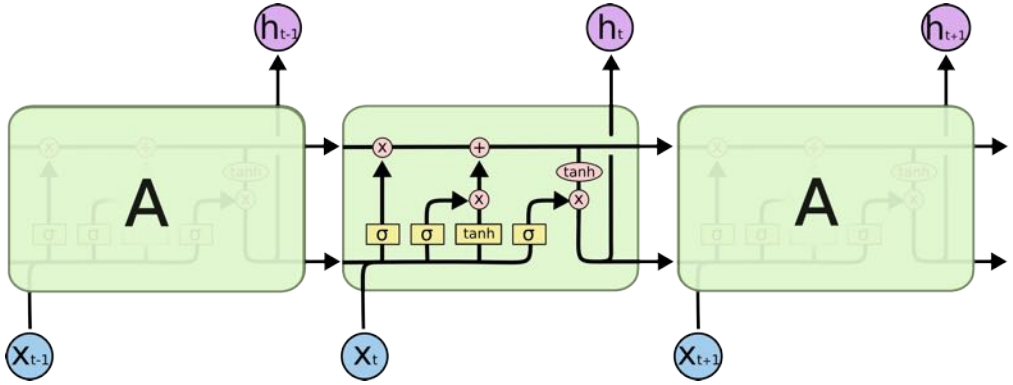
Forget gate „layer“

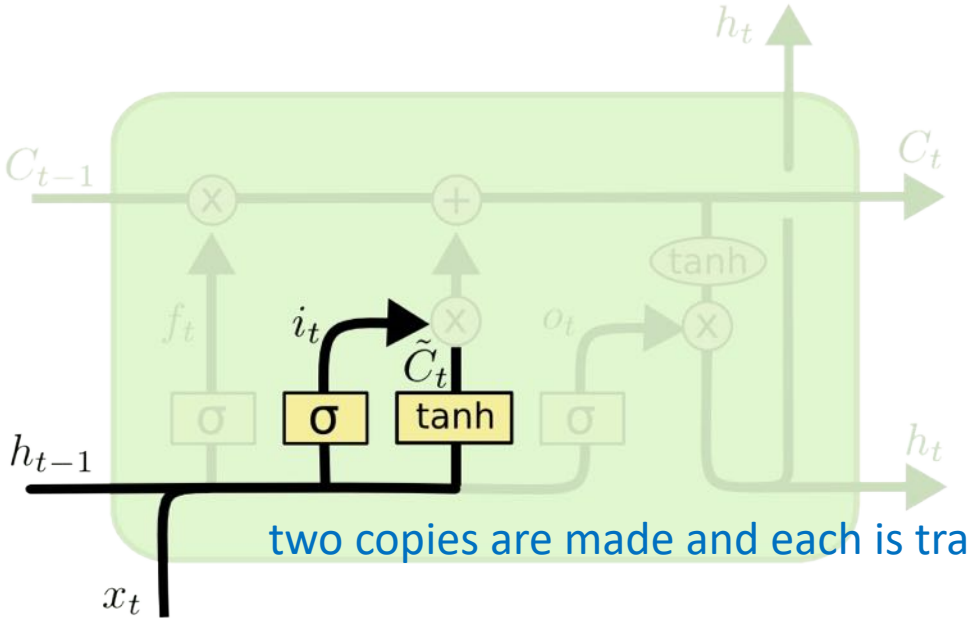




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Input gate „layer“



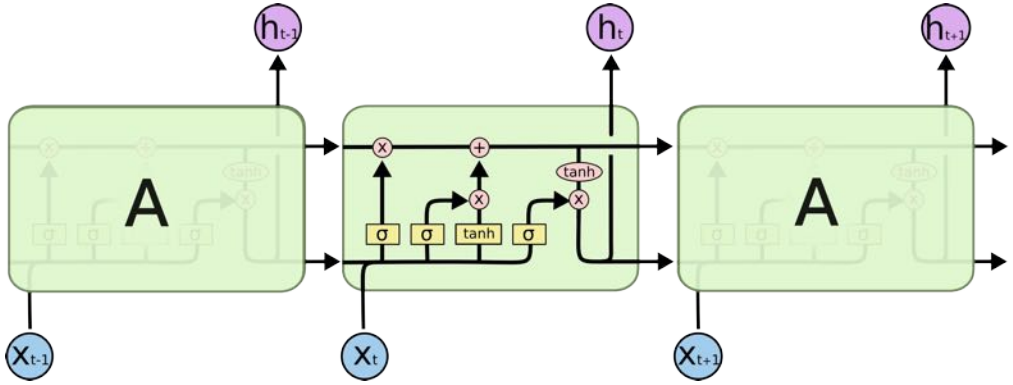


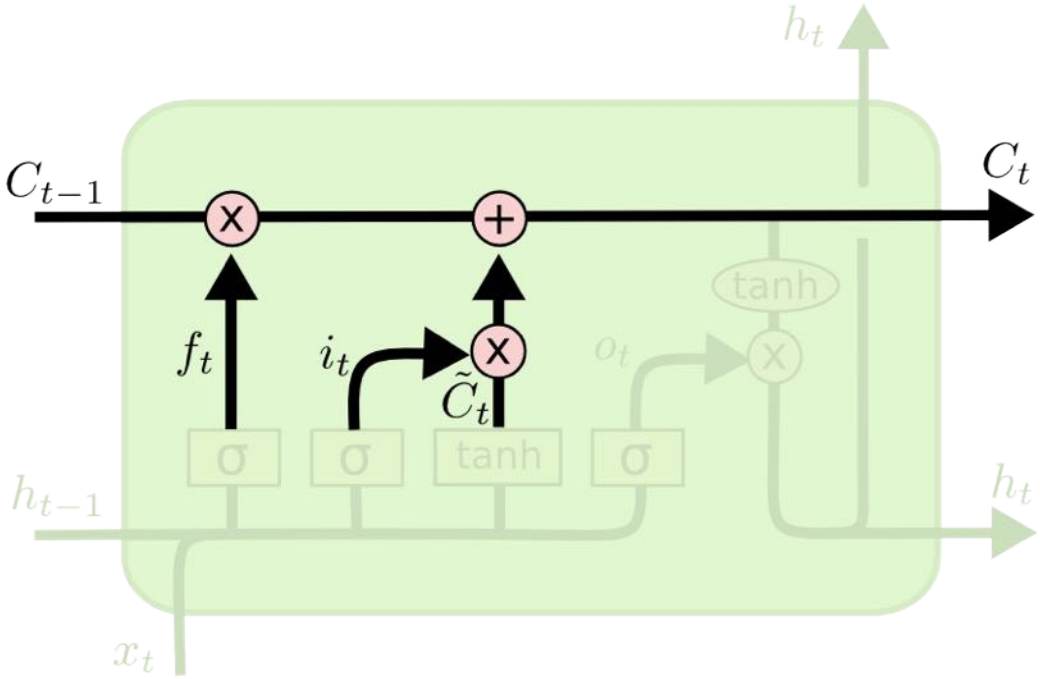
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

two copies are made and each is transformed onto a different value !

The current state



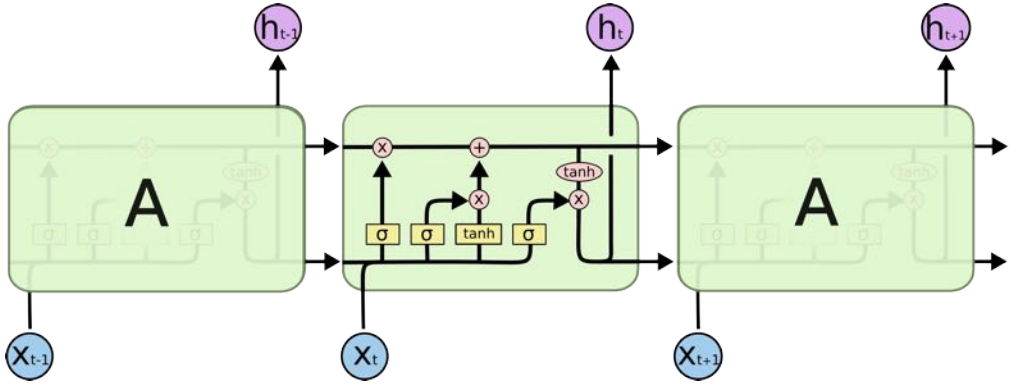


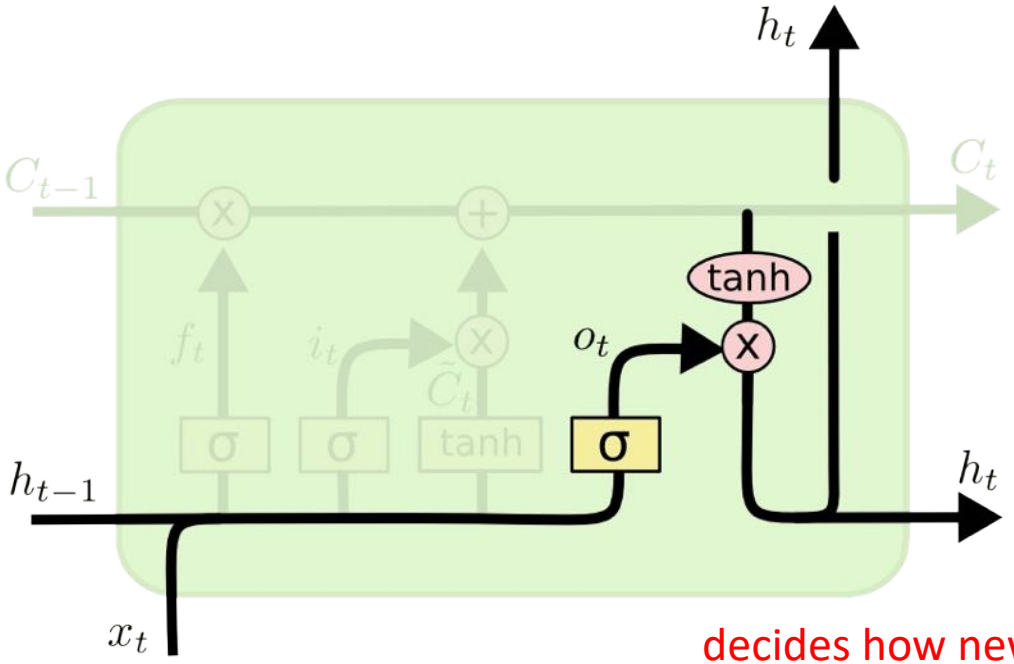
decides if previous state is forgotten or used

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

decides if previous output and current input are forgotten or used

Output„layer“

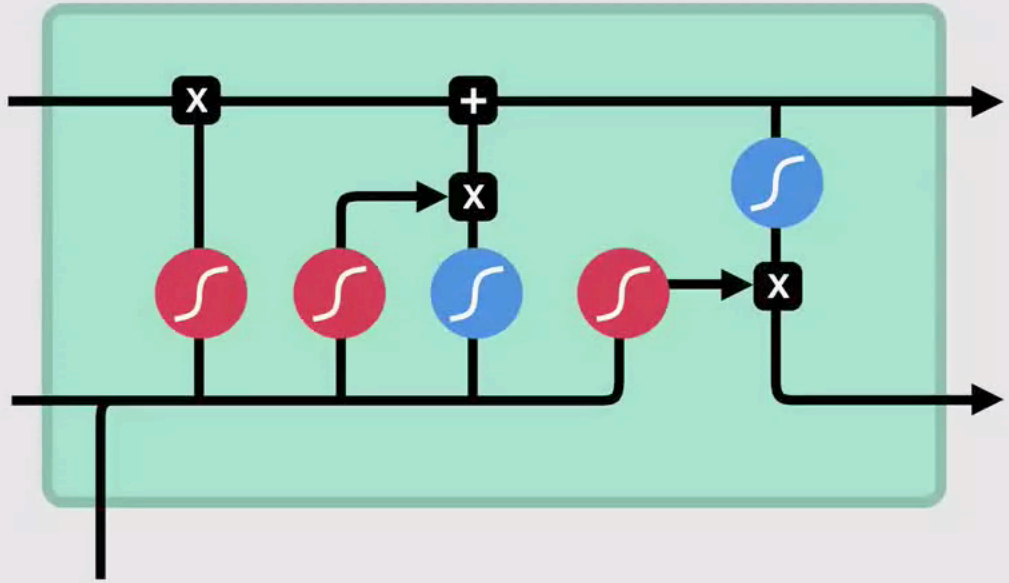




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

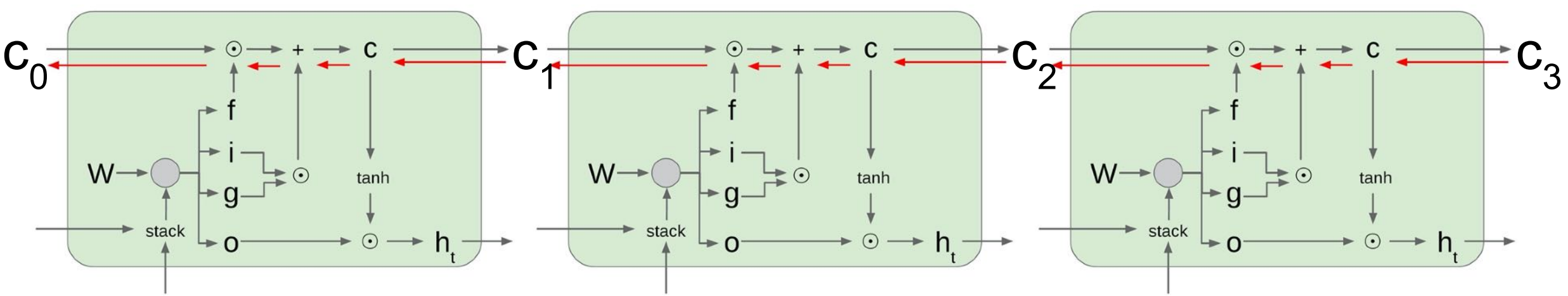
decides how new cell state contributed to the output


X
+
X
X

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

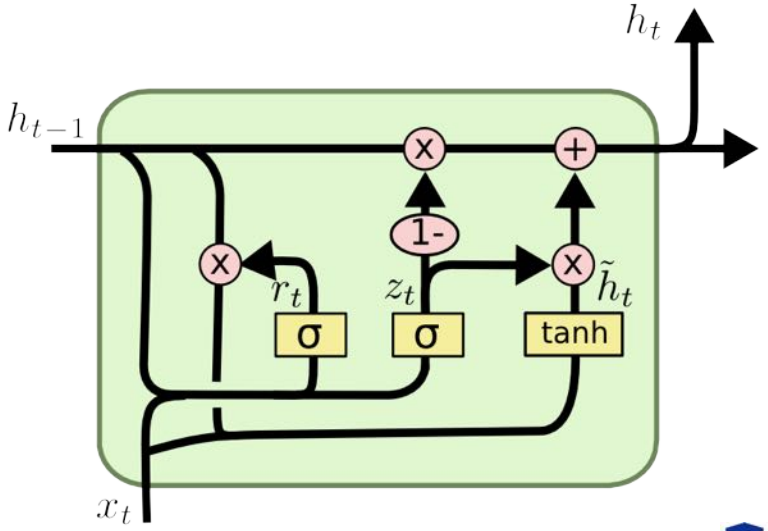
Uninterrupted gradient flow!



The gradient involves no repeated tanh and W multipication !

Gated Recurrent Unit (GRU)

- Alternative RNN to LSTM that uses fewer gates (Cho, et al., 2014)
  - Combines forget and input gates into “update” gate.
  - Eliminates cell state vector



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

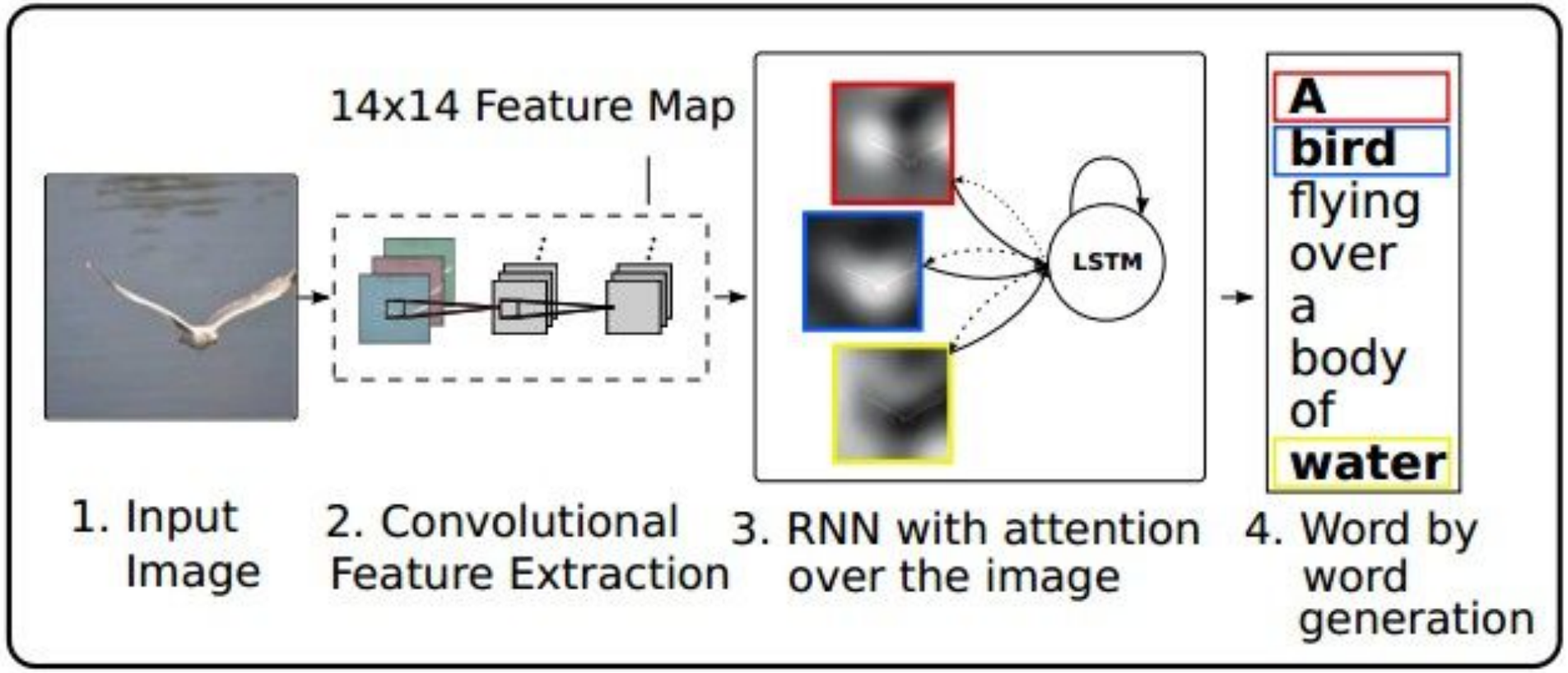
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM Image captioning with attention

RNN focuses its attention at a different spatial location when generating each word



Xu et al, “Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image captioning with attention

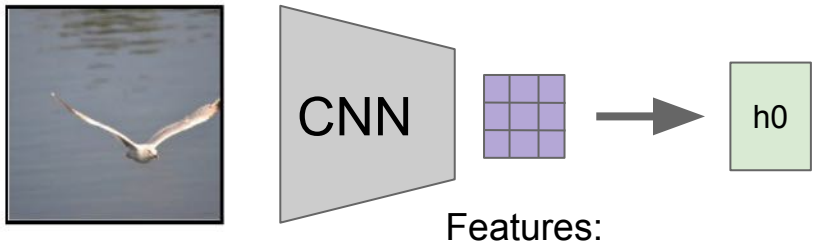


Image captioning with attention

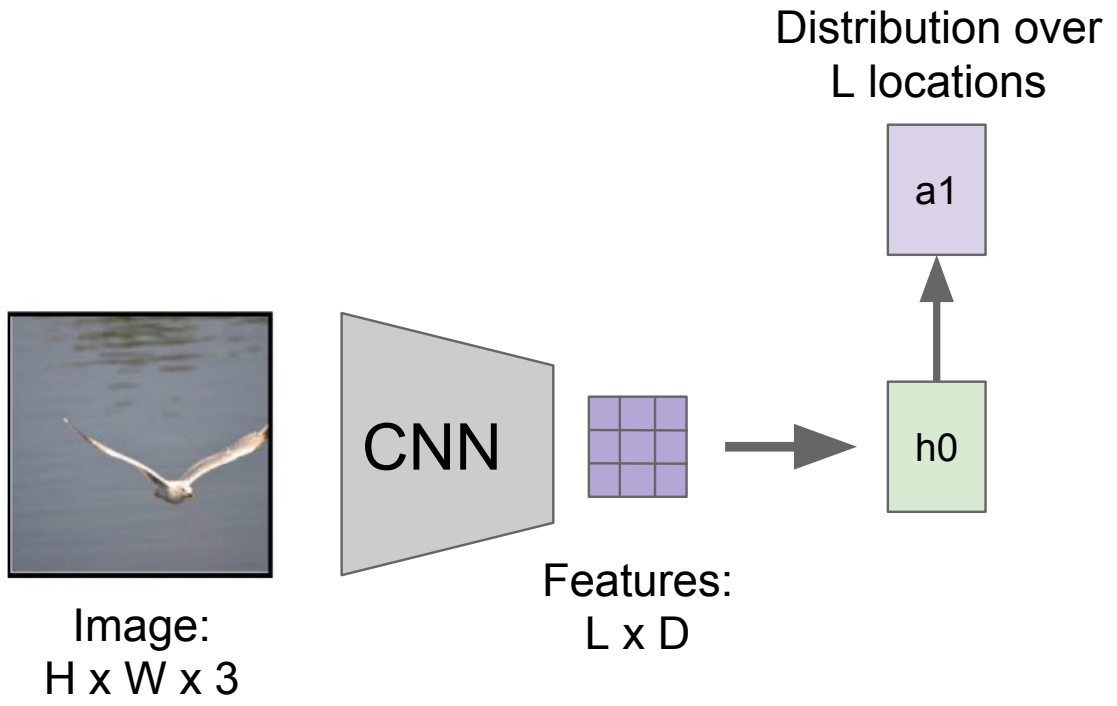


Image captioning with attention

Image:
H x W x 3

CNN

Features:
L x D

Distribution over
L locations

a1

h0

Weighted
features: D

z1

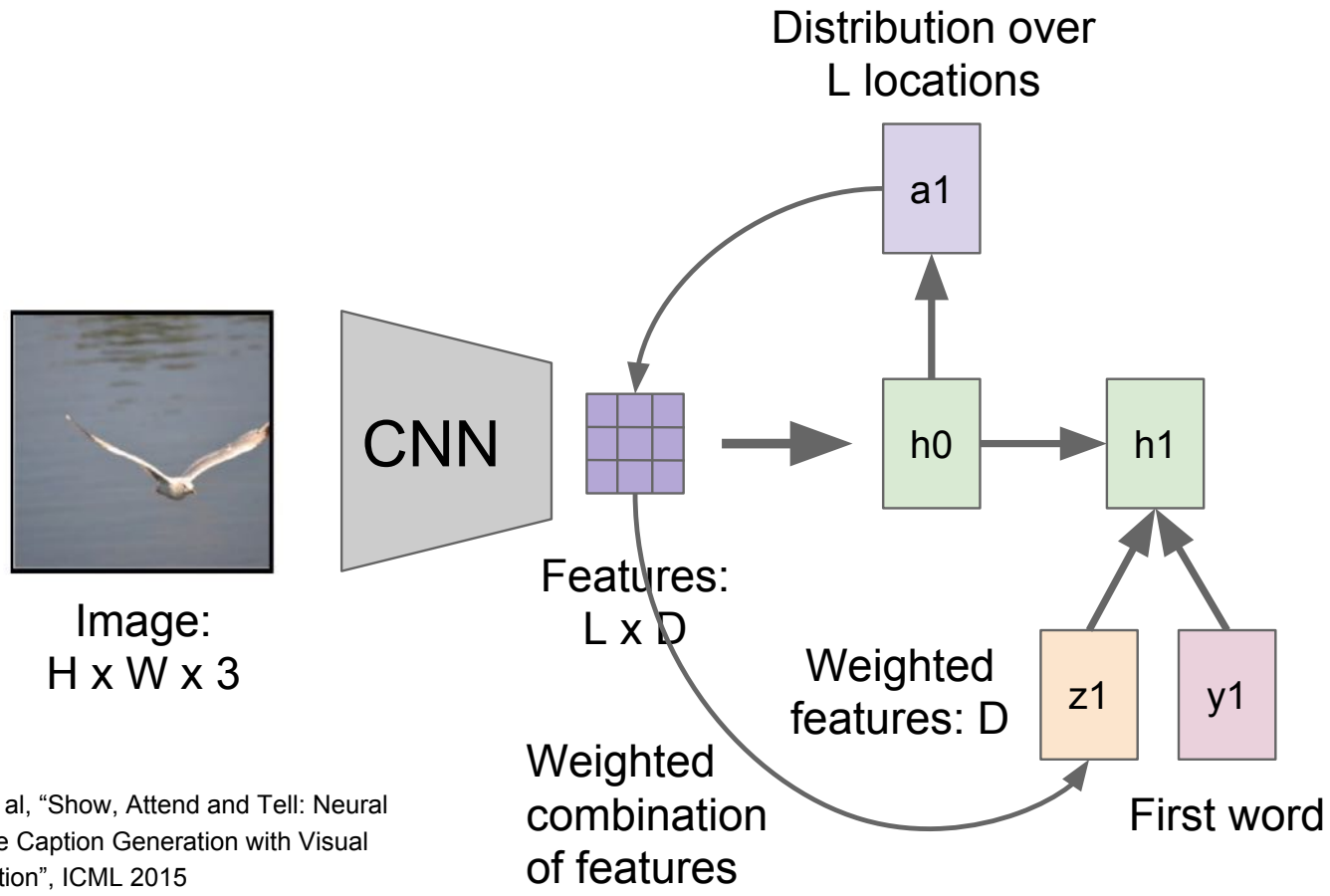
Weighted
combination
of features

$$z = \sum_{i=1}^{L} p_i v_i$$

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

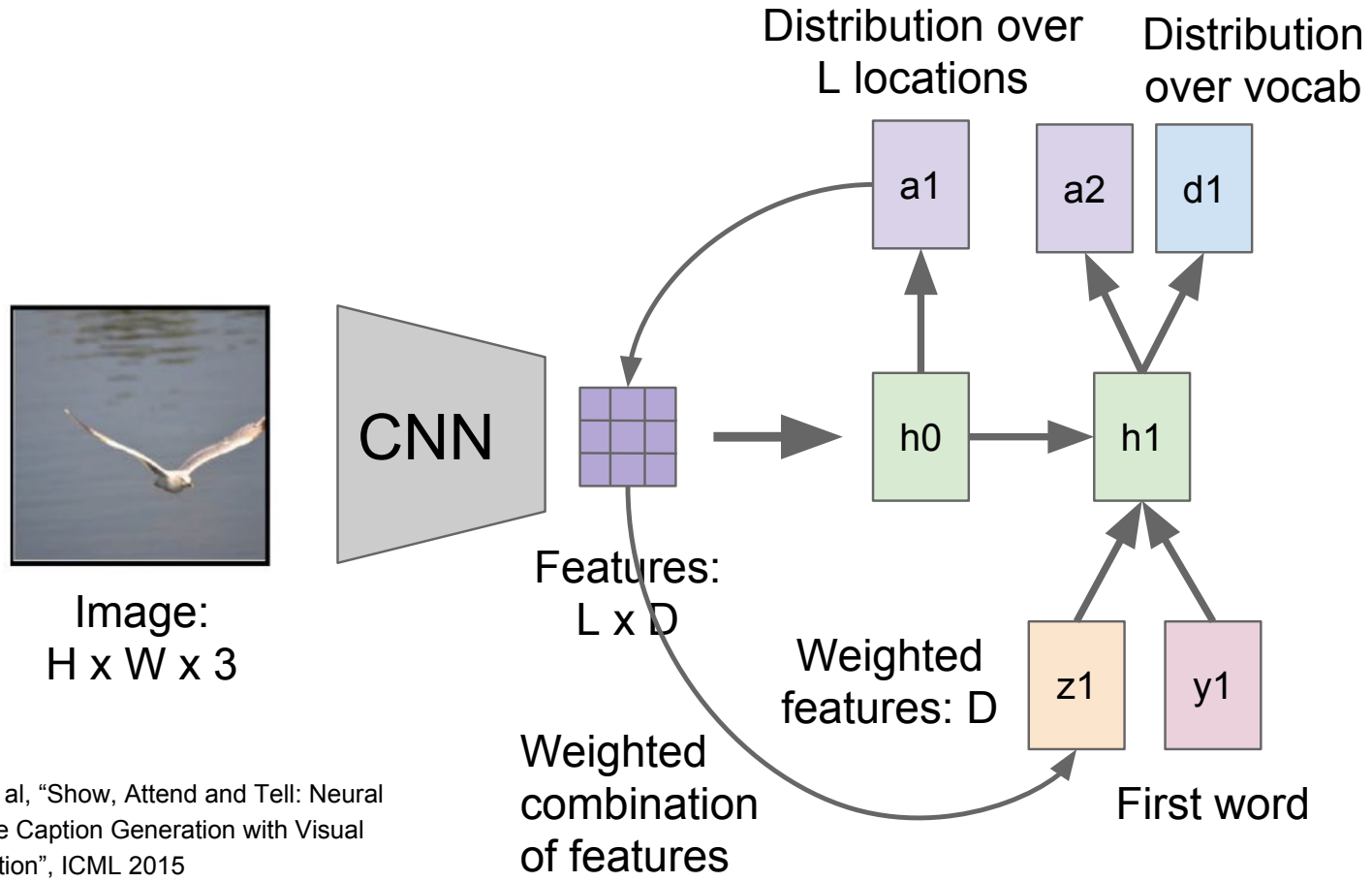
Embedded Intelligence

Image captioning with attention



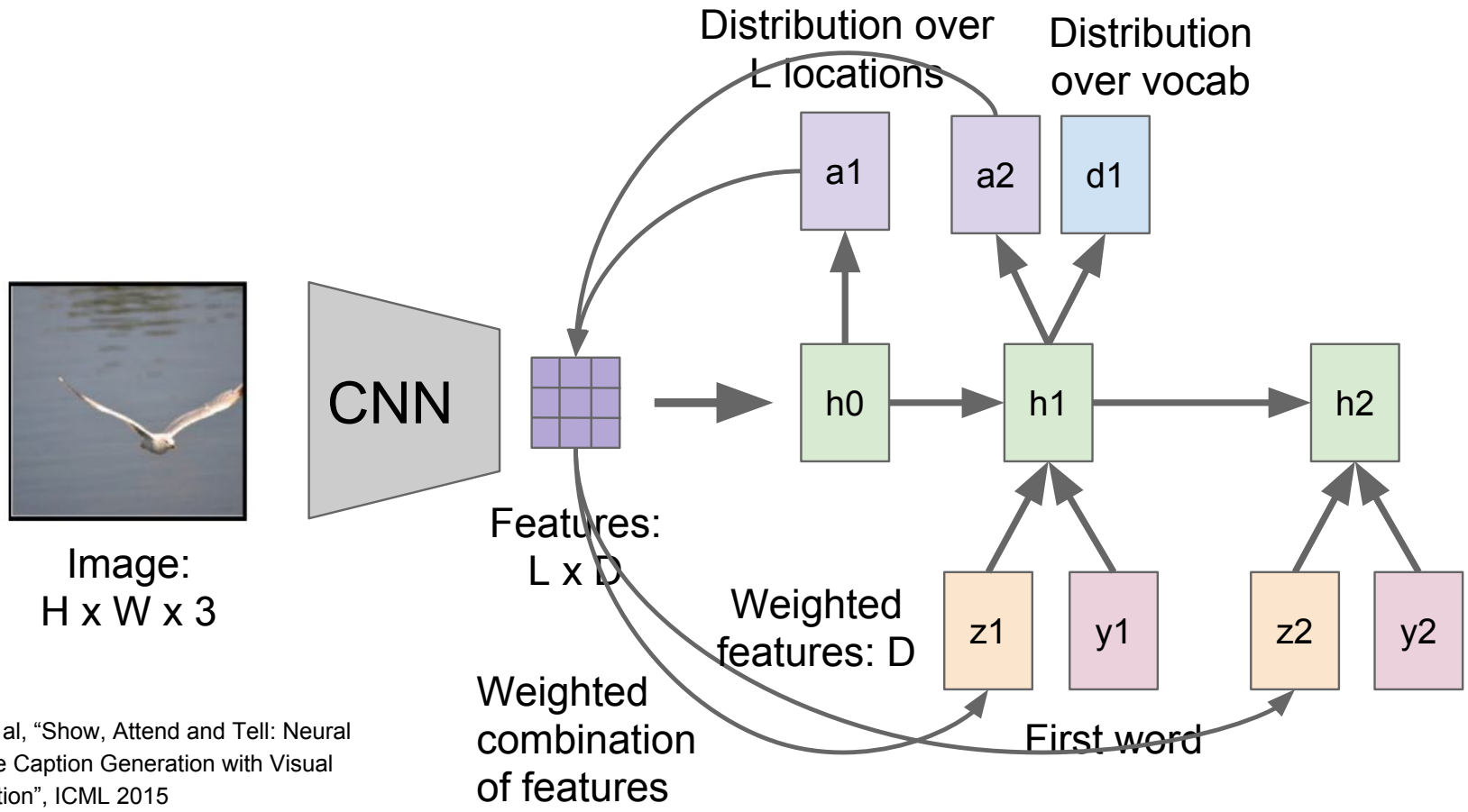
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image captioning with attention



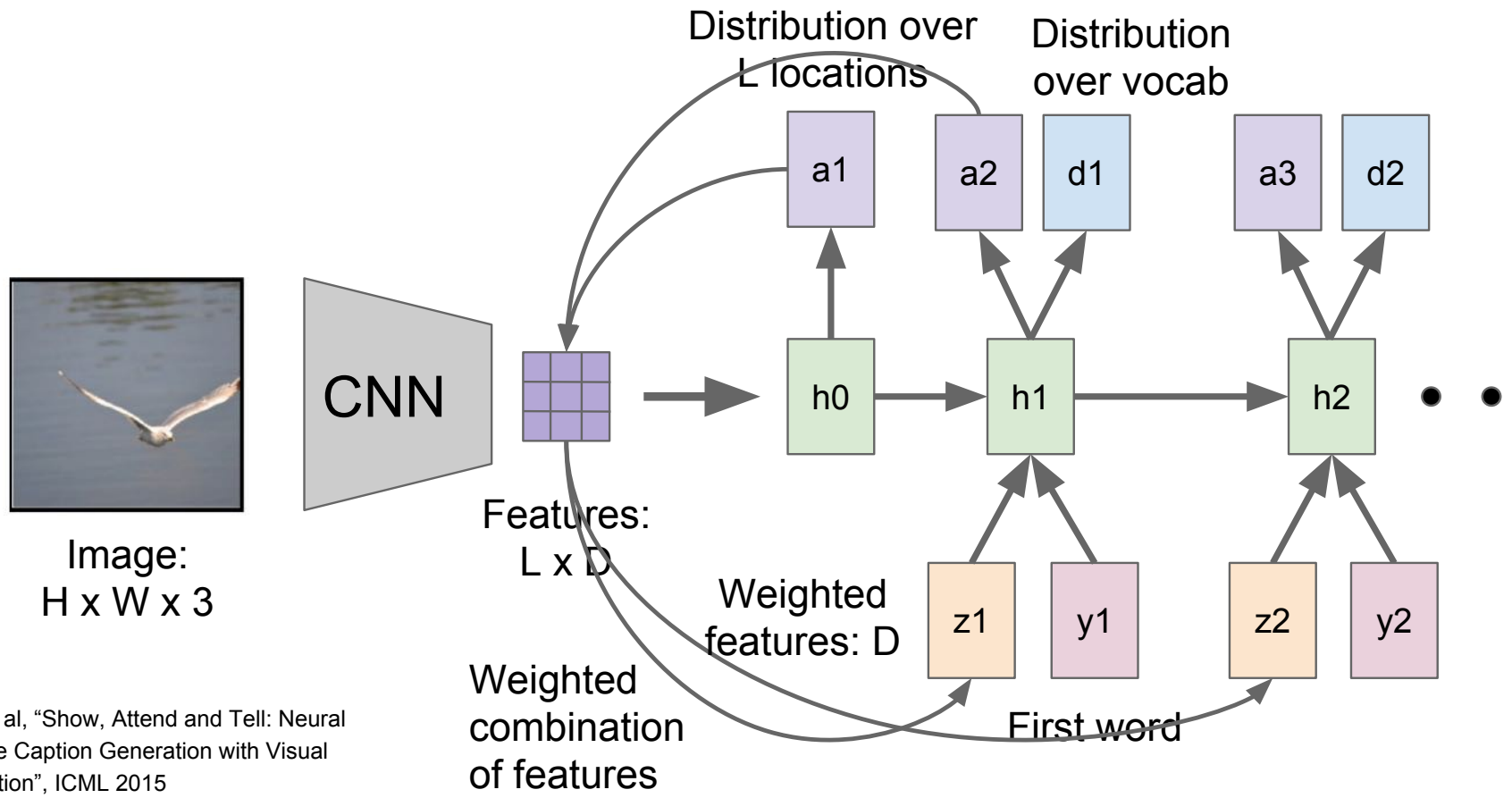
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image captioning with attention



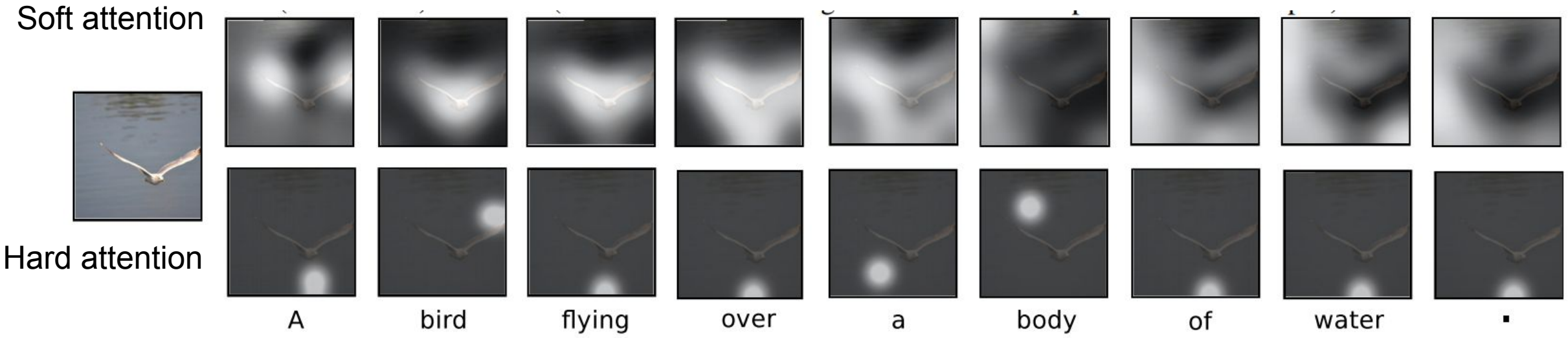
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image captioning with attention



Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

LSTM Image captioning with attention



LSTM Image captioning with attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

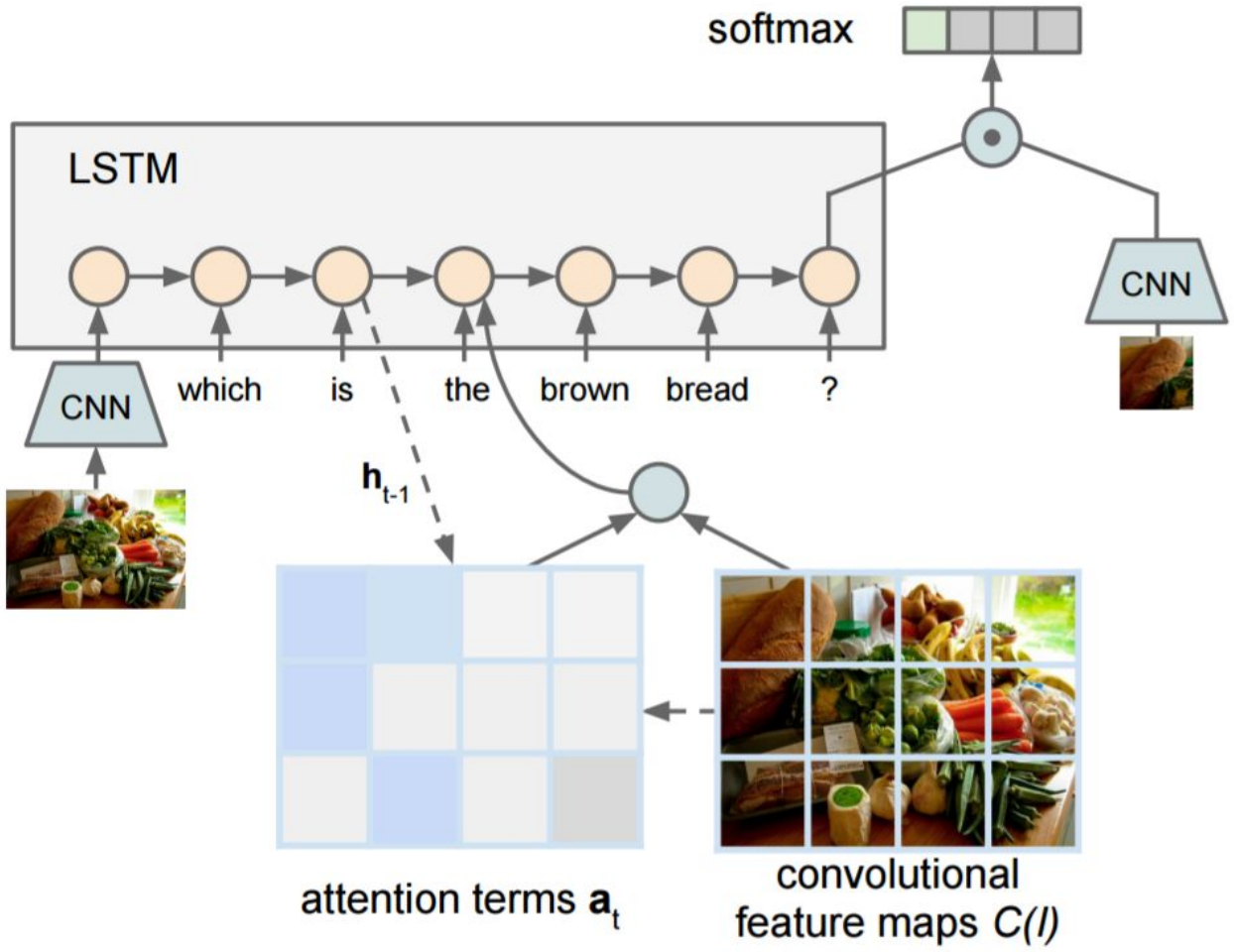
A stop sign is on a road with a mountain in the background.

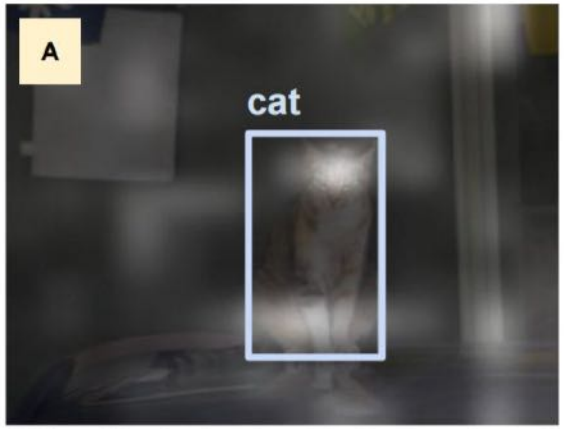
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Visual Question Answering: RNNs with Attention





What kind of animal is in the photo?

A **cat**.

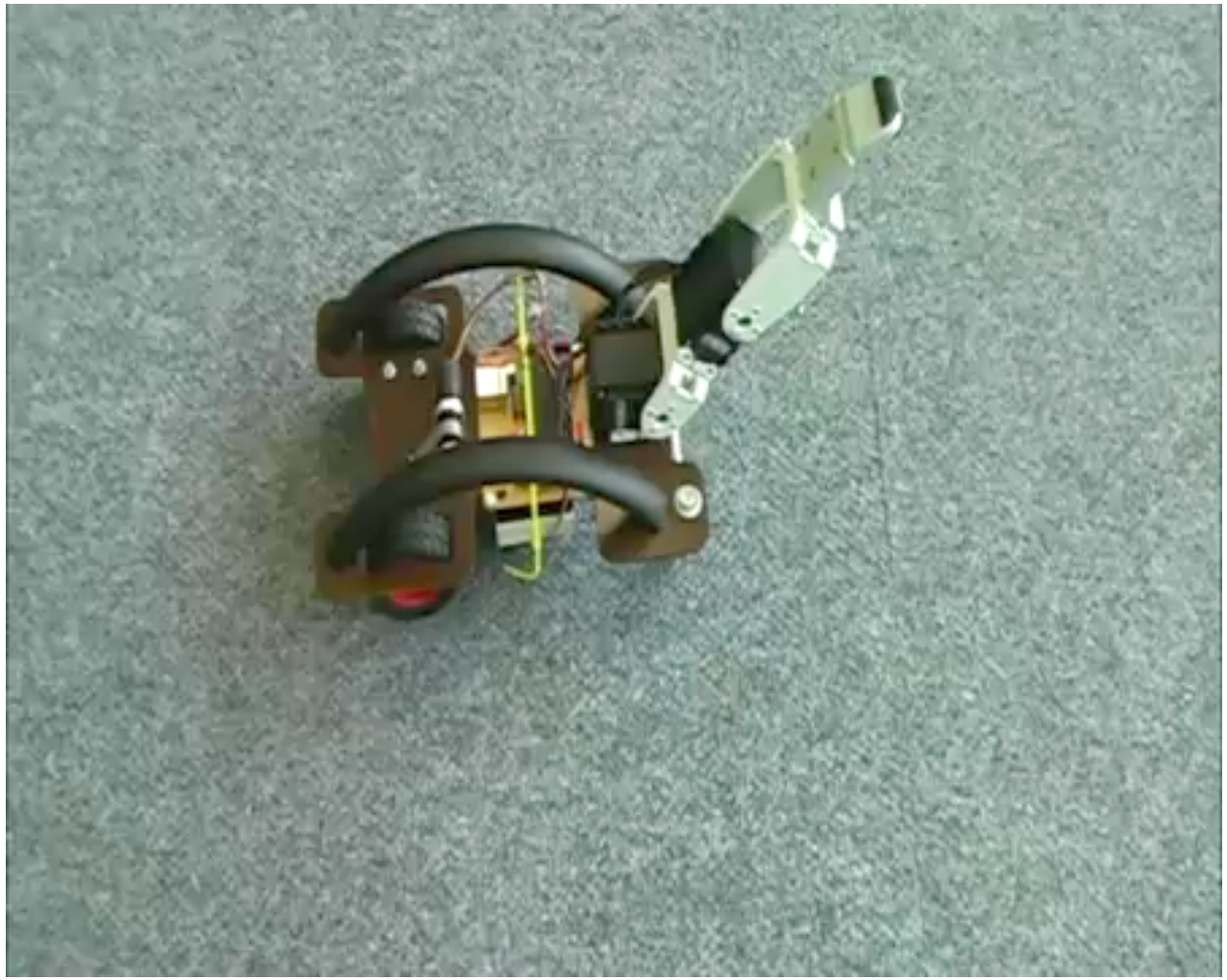
Why is the person holding a knife?

To cut the **cake** with.

Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016

A different type of time dependent learning..

Learning Temporal Control Sequences

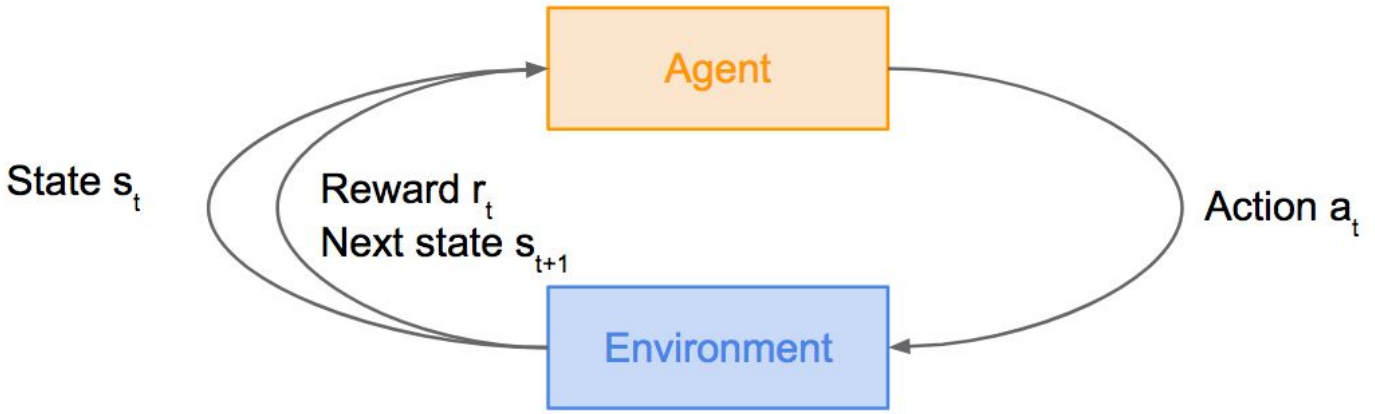


finding out how to control actuators by sensing effects of actions

Reinforcement Learning

- Problems involving an **agent** interacting with an **environment**, which provides numeric **reward** signals

- **Goal**: Learn how to take actions in order to maximize reward

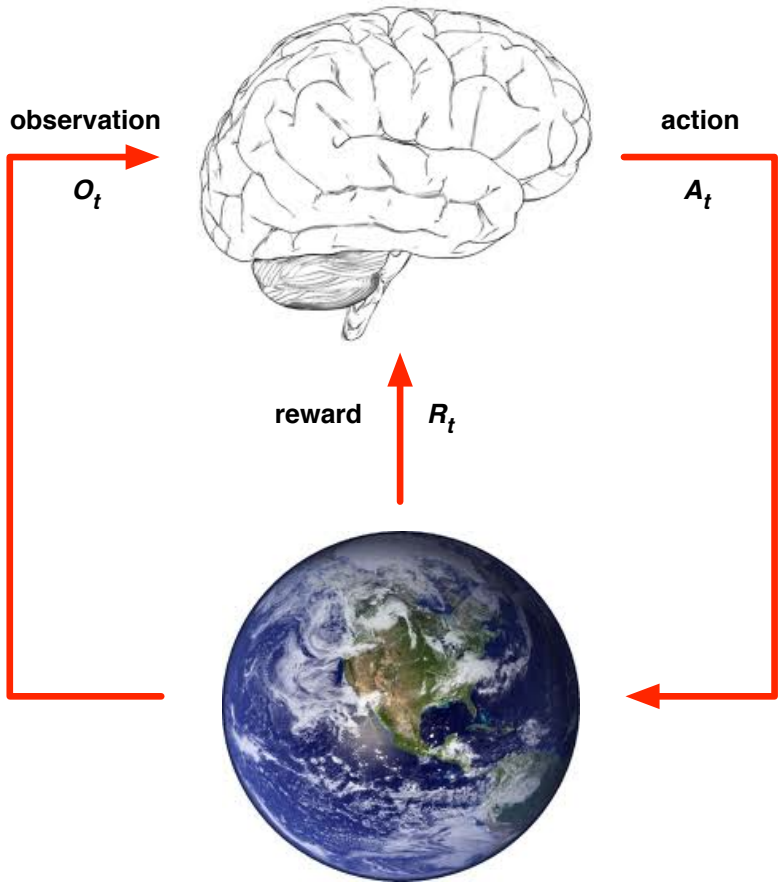


Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms?

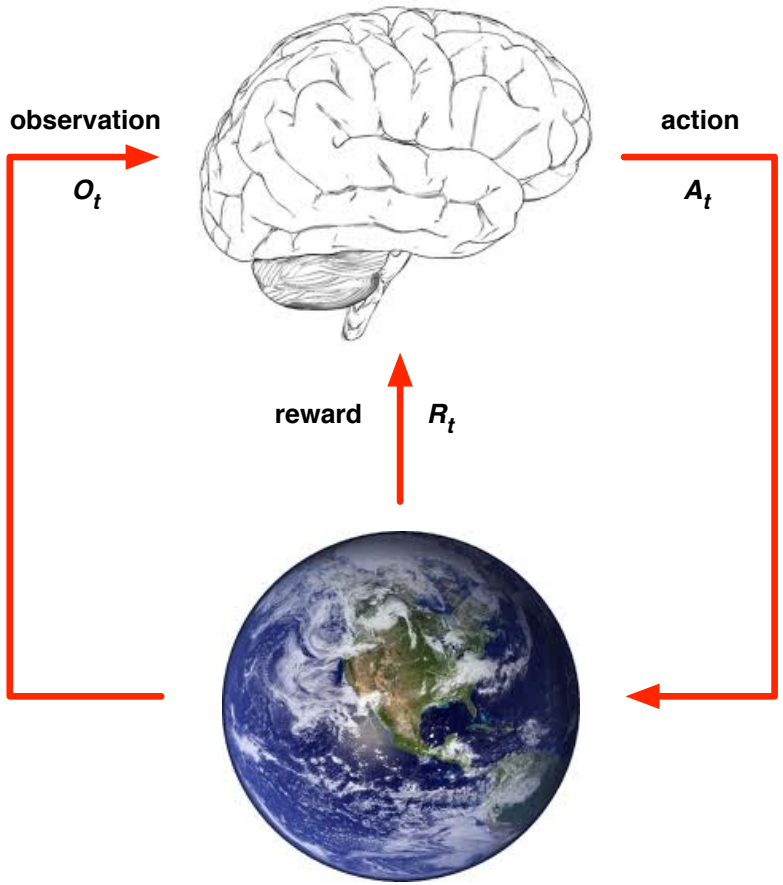
- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent’s actions affect the subsequent data it receives

Reinforcement Learning



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

Reinforcement Learning



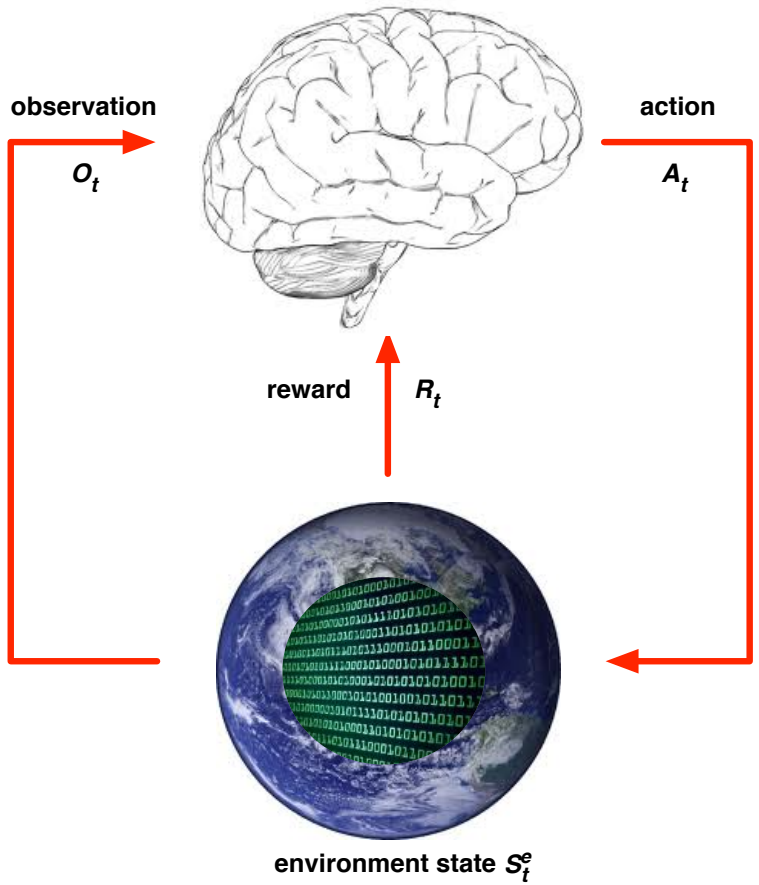
- The history is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

- i.e. all observable variables up to time t
- i.e. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
- Formally, state is a function of the history:

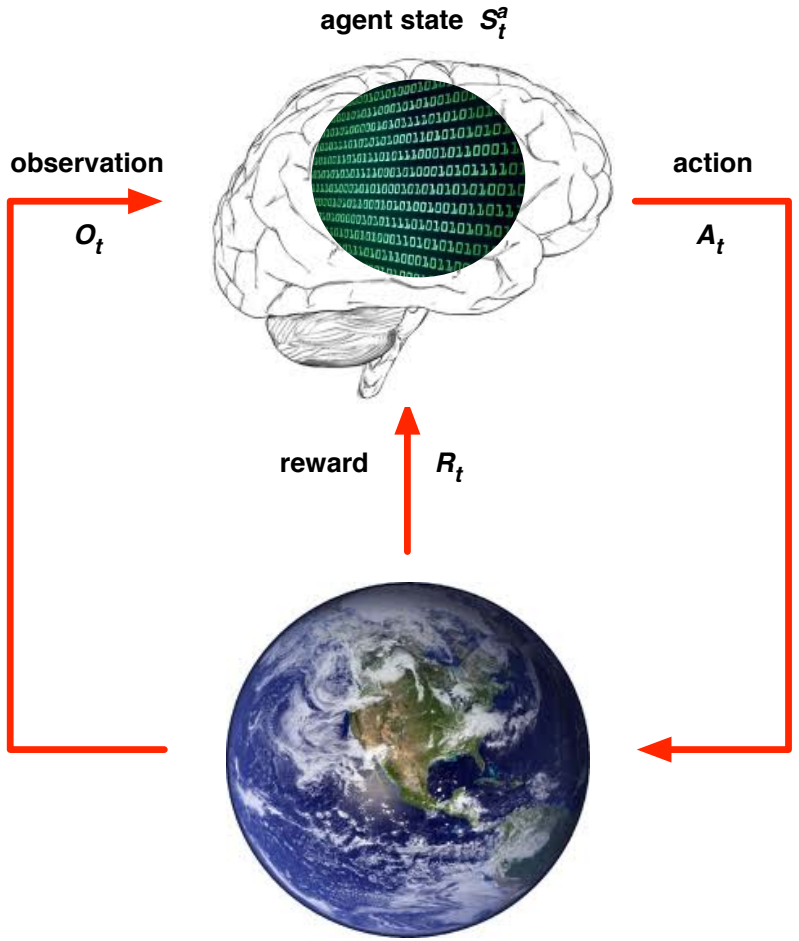
$$S_t = f(H_t)$$

Reinforcement Learning



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

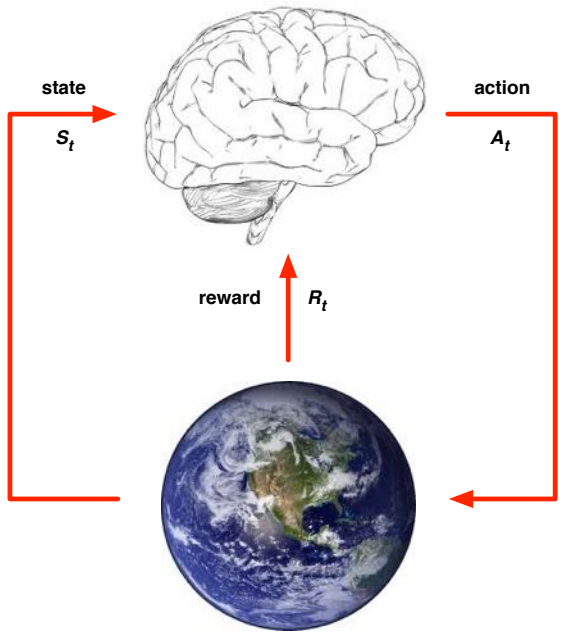
Reinforcement Learning



- The agent state S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

Reinforcement Learning



Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

- Agent state = environment state = information state
- Formally, this is a Markov decision process (MDP)

Reinforcement Learning

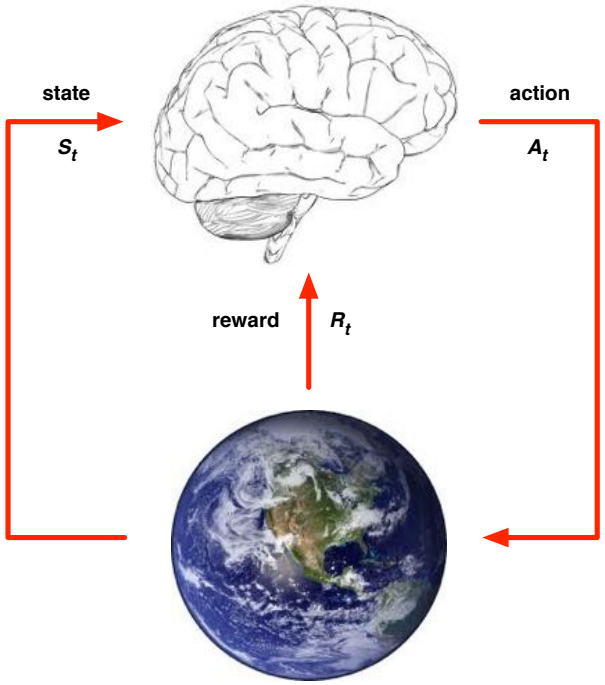
- An RL agent may include one or more of these components:

1. Policy: agent's behaviour function

2. Value function: how good is each state and/or action

3. Model: agent's representation of the environment

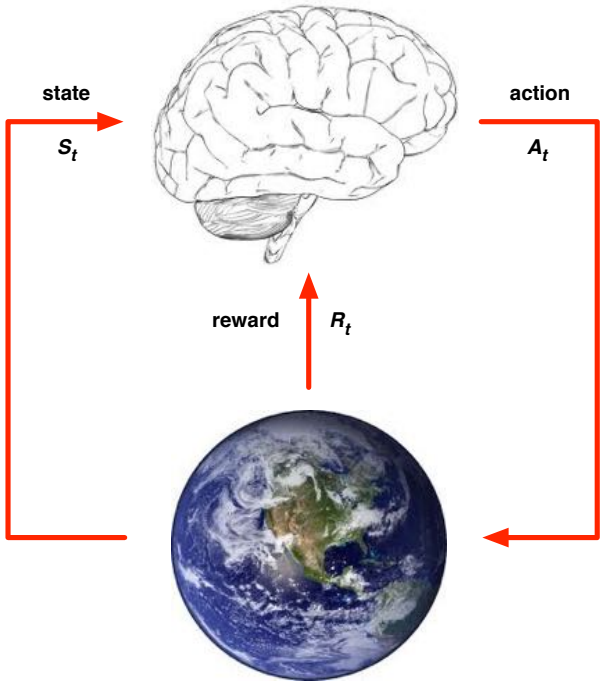
Reinforcement Learning



- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Reinforcement Learning



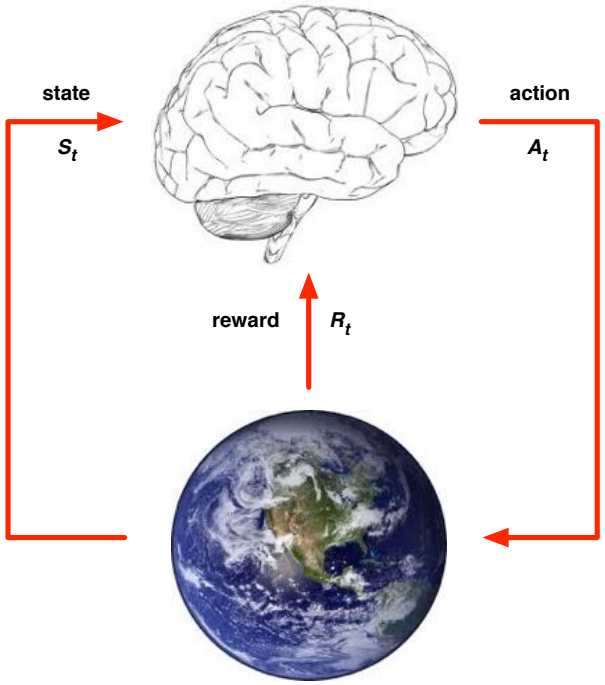
- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

Expectated value
(Erwartungswert)

Reinforcement Learning



- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

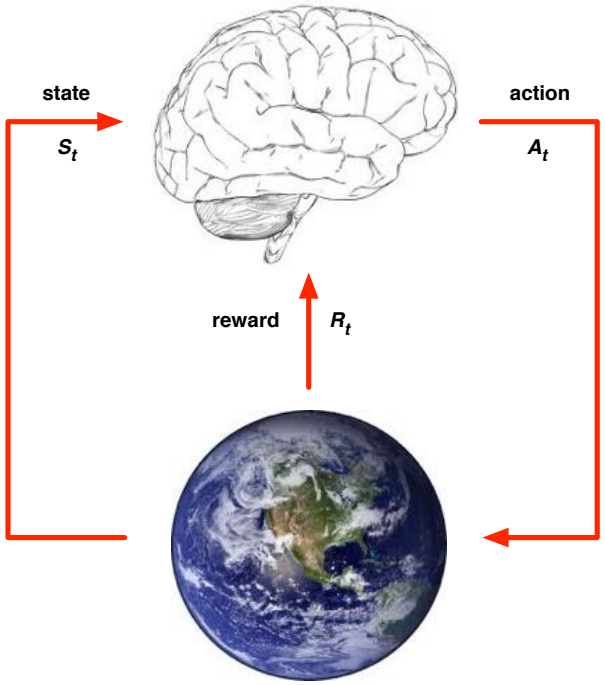
- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t R_t | s_0 = s, a_0 = a, \pi \right]$$

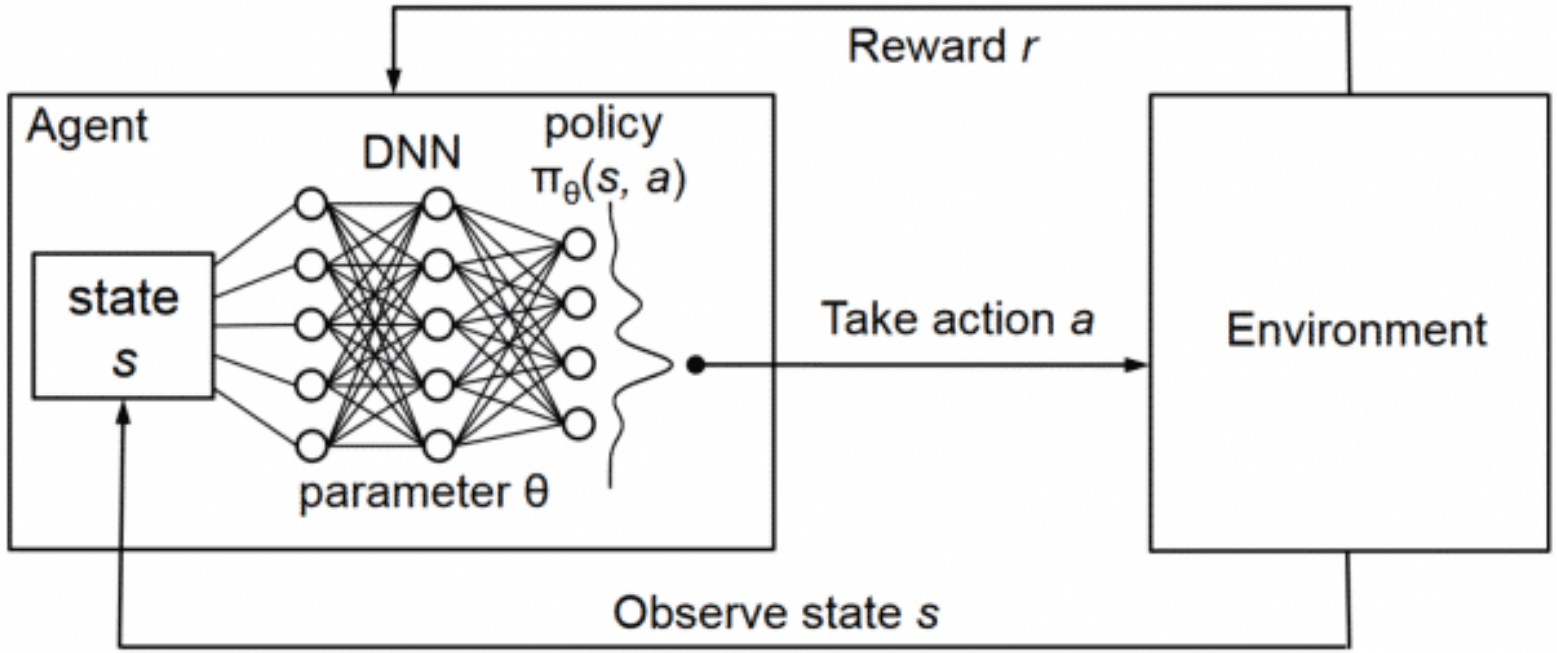
Reinforcement Learning: Value Iteration



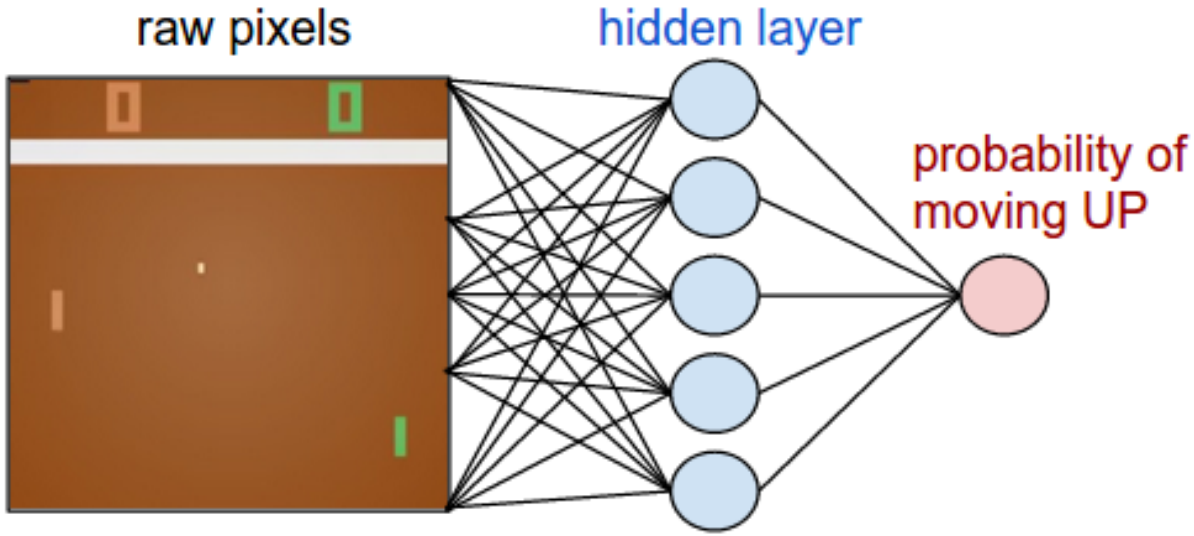
Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in \mathcal{S}$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[R|s,a] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a)V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

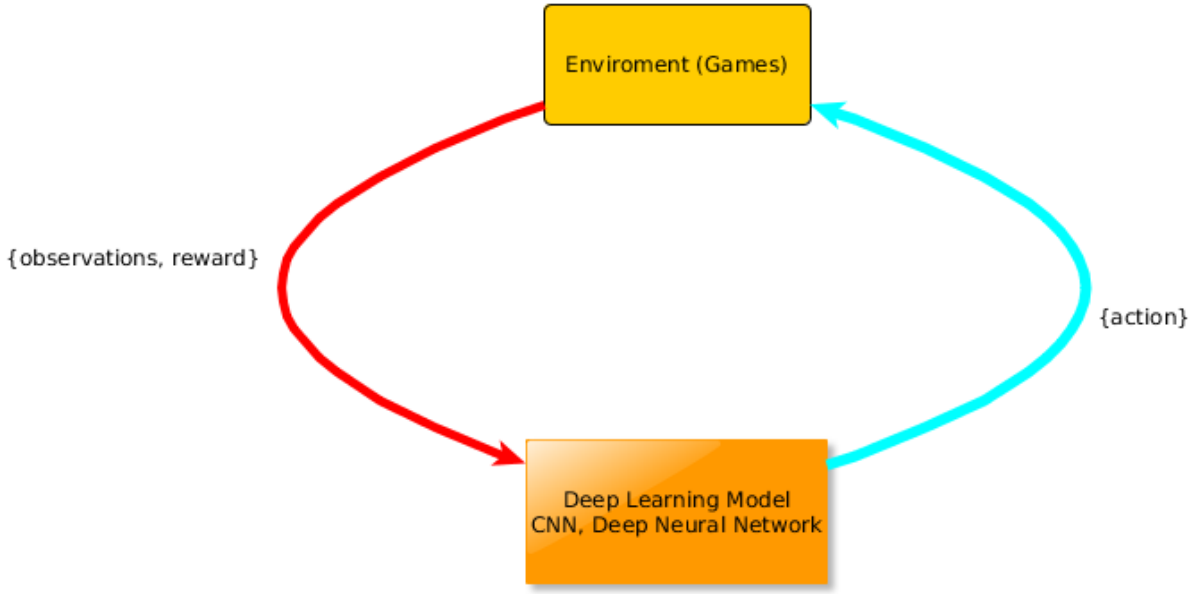
Renforcment Learning with Neural Networks

Reward can be used as error/gain function/gradient for training



Renforcment Learning wth Neural Networks

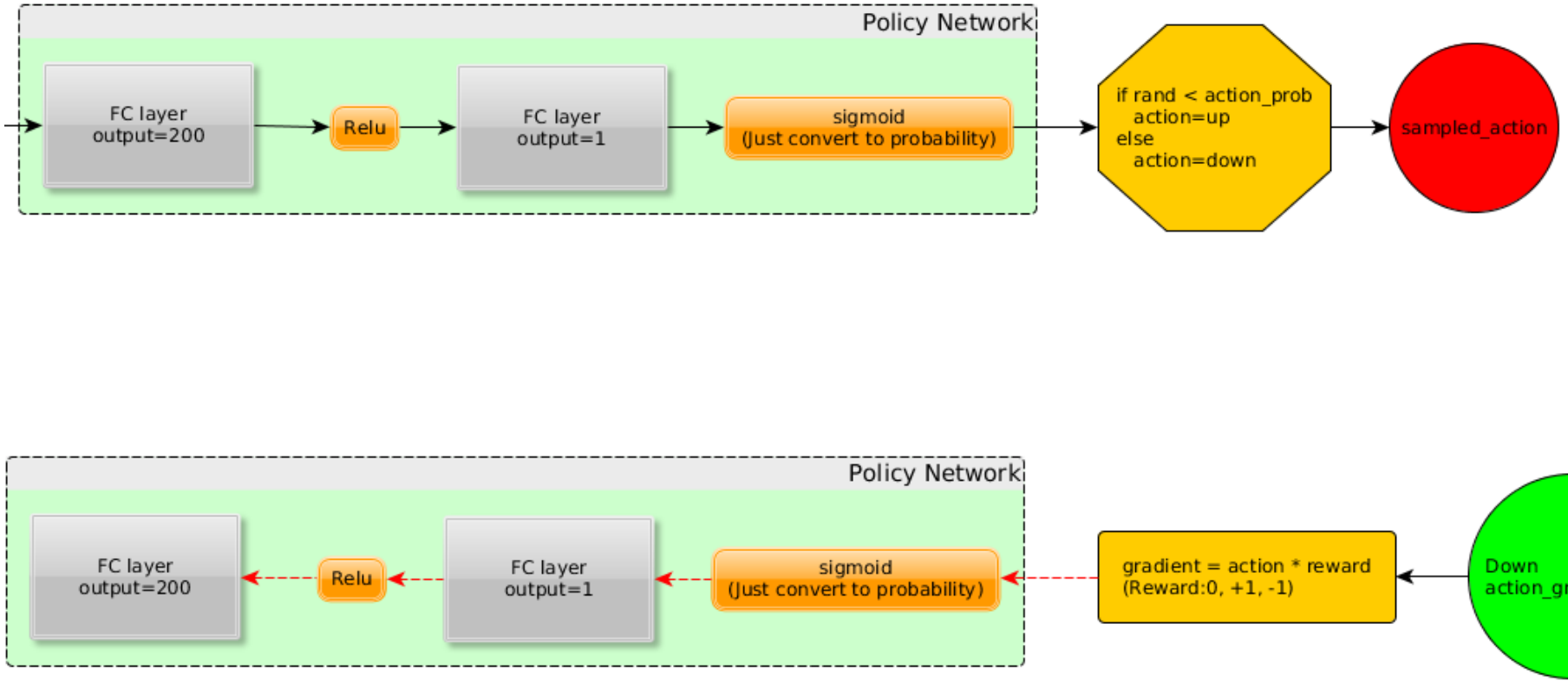




The model will do actions that maximize the rewards.

The problems that are actions that takes a lot of time to give rewards (late rewards)

Policy Gradient



Policy/Action Gradient:

Reward Zero: Our weights will remain the same

Reward Positive: Make the network more likely to repeat this action on the future

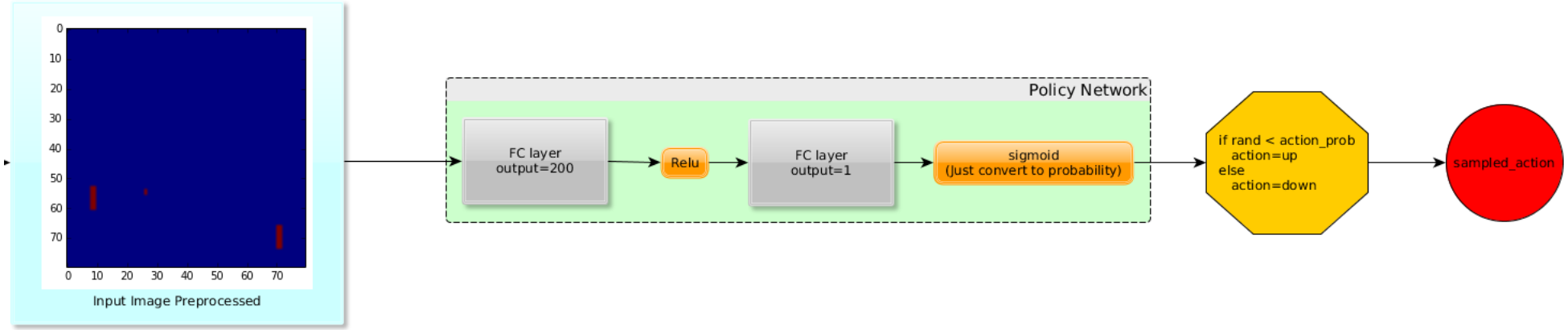
Reward Negative: Make the network less likely to repeat this action on the future

Renforcment Learning with Neural Networks: Training

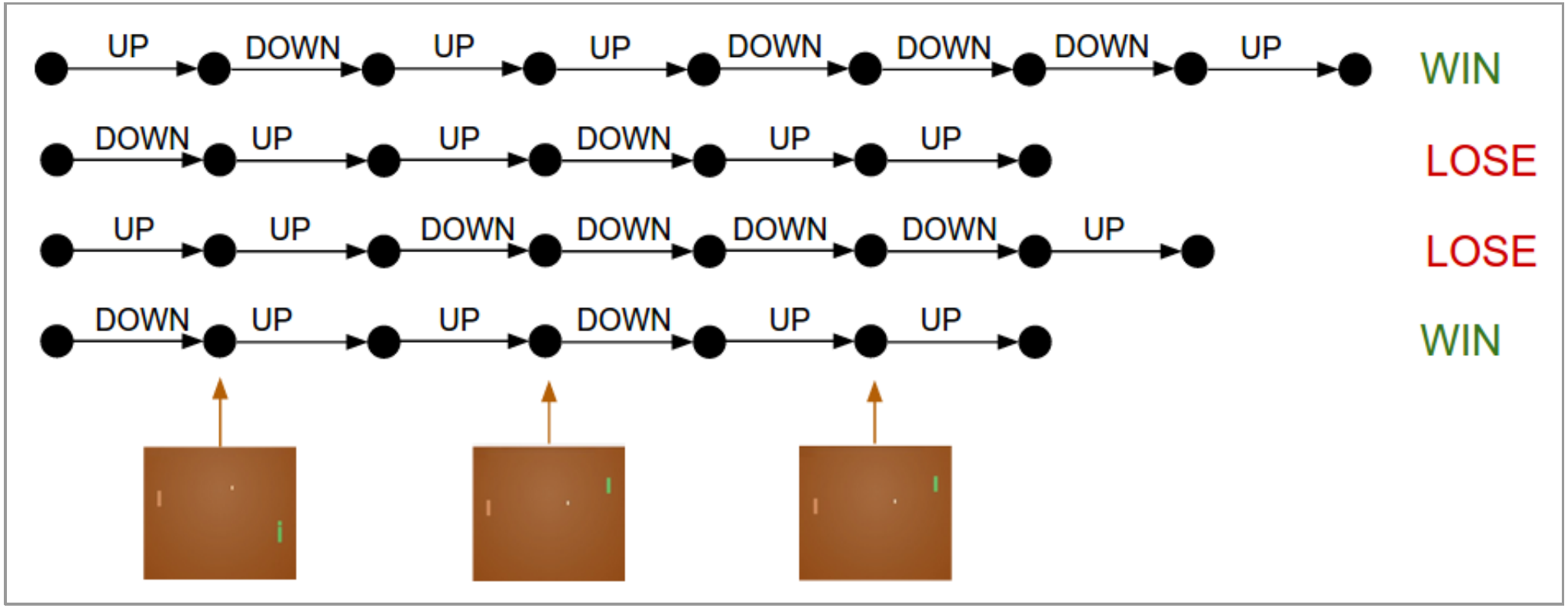
1. First we will initialize randomly W1,W2.
2. Then we will play 20 games (one episode).
3. Keep track of all games and their result (win/loose)
4. After a configured number of episodes we update our policy network

Assuming that each game has 200 frames, and each episode has 20 games, we have to make 4000 decisions (up/down) per episode. Suppose that we run 100 games 12 we win (+1) and 88 we loose(-1).

- We need to take all the 12x200=2400 decisions and do a positive(+1) update.
- Now all the 88x200=17600 decisions that make us loose the game we do a negative(-1) update.

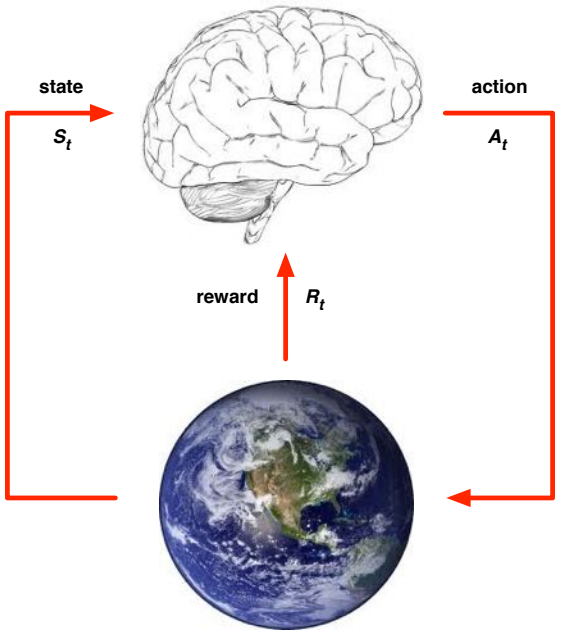


Example



Cartoon diagram of 4 games. Each black circle is some game state (three example states are visualized on the bottom), and each arrow is a transition, annotated with the action that was sampled. In this case we won 2 games and lost 2 games. With Policy Gradients we would take the two games we won and slightly encourage every single action we made in that episode. Conversely, we would also take the two games we lost and slightly discourage every single action we made in that episode.

More on Reinforcement Learning

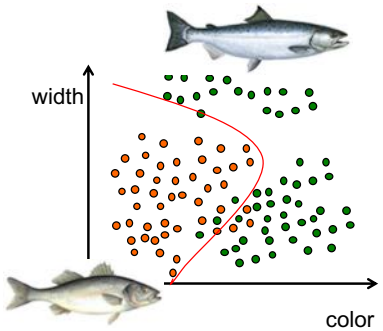


- Partial observability: agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now agent state $\neq$ environment state
- Formally this is a partially observable Markov decision process (POMDP)
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], ..., \mathbb{P}[S_t^e = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

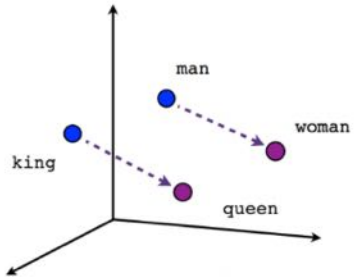
There is actually much much more to Reinforcement Learning (but not in this lecture)

Result: A target function derived from the sample
to do „something usefull“

on data points that are related to the original space but as such have not been seen before !!!

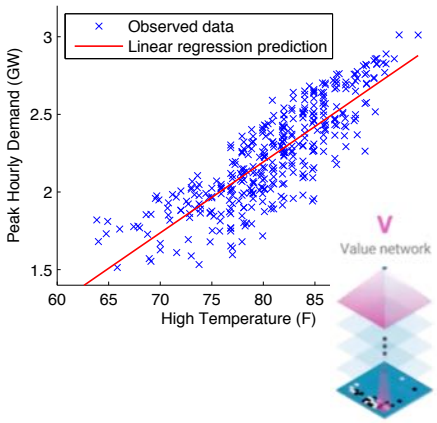


classification



Male-Female

(mapping to)
new representation



value predicton

generation of new
samples (completion)

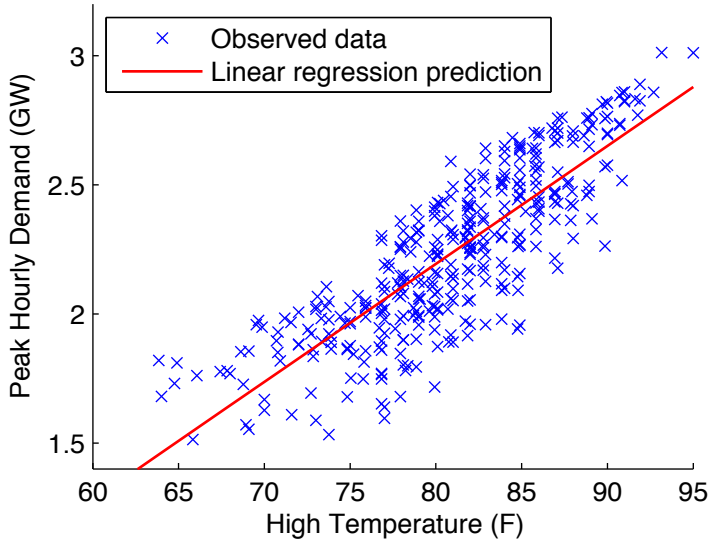
Learning in Linear Regression

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



$$\text{predicted peak demand} = \theta_1 \cdot (\text{high temperature}) + \theta_2$$

Parameters of model: $\theta_1, \theta_2 \in \mathbb{R}$

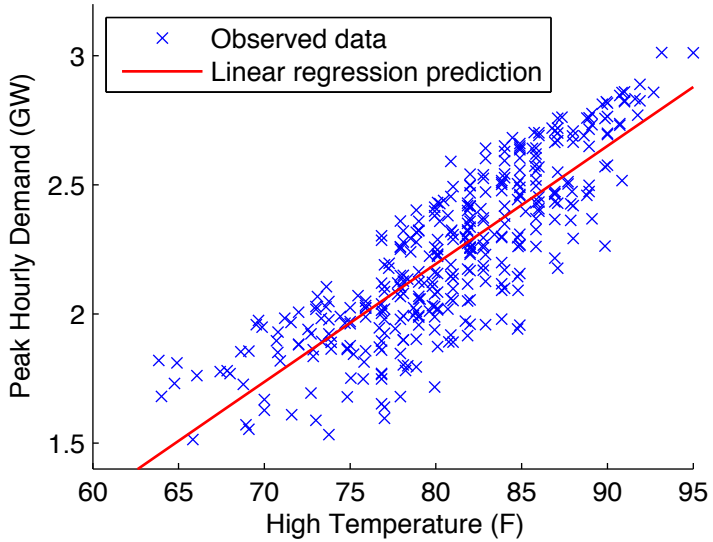
Learning in Linear Regression

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



Search algorithm: Start with an initial guess for θ. Keep changing θ (by a little bit) to reduce $J(\theta)$

$$J(\theta) = \sum_{i=1}^{m} \ell(\hat{y}_i, y_i) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2$$

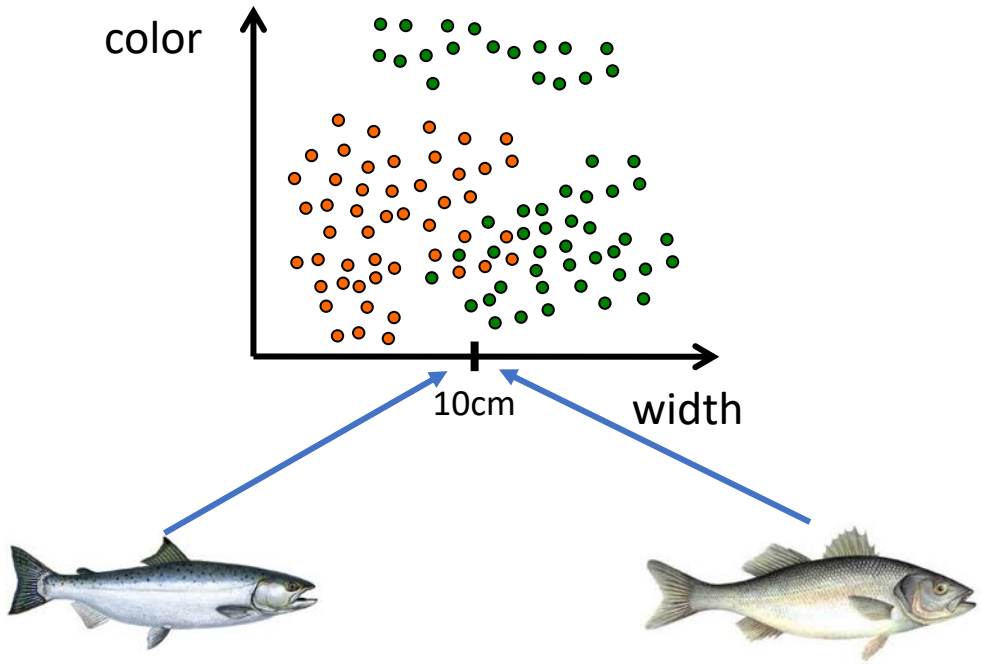
Learning Probabilities

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

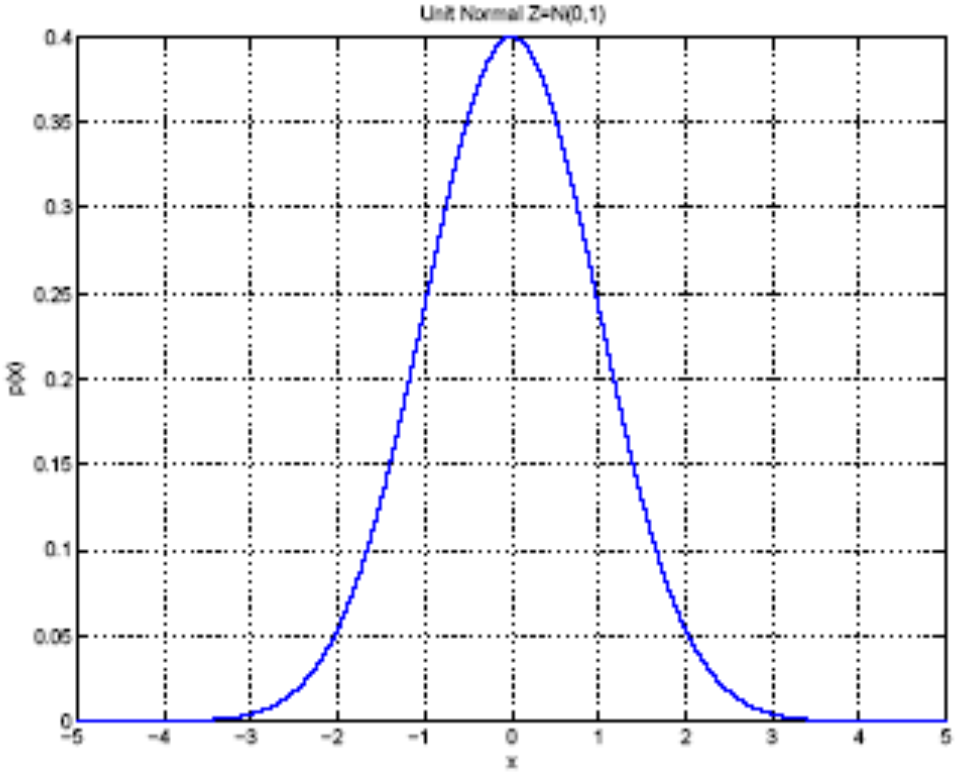
3. use an appropriate optimization techniques to find parameter values that minimize the error function



What is the probability of a randomly caught fish having a width of 10cm ?

- obviously different for each class

Background:Gaussian (Normal) Distribution



- $p(x) = \mathcal{N}(\mu, \sigma^2)$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

Parameter

- *μ : Mittelwert*
- und σ^2: Varianz

Many distributions seen in the real world can be approximated by a Gaussian

- gaussians are so easy to handle that they are often used even if they do not fit well

Given a data set, how can we estimate what Guassian (or other standard distribution) fits it best ?

Beispiel 2D Gaus

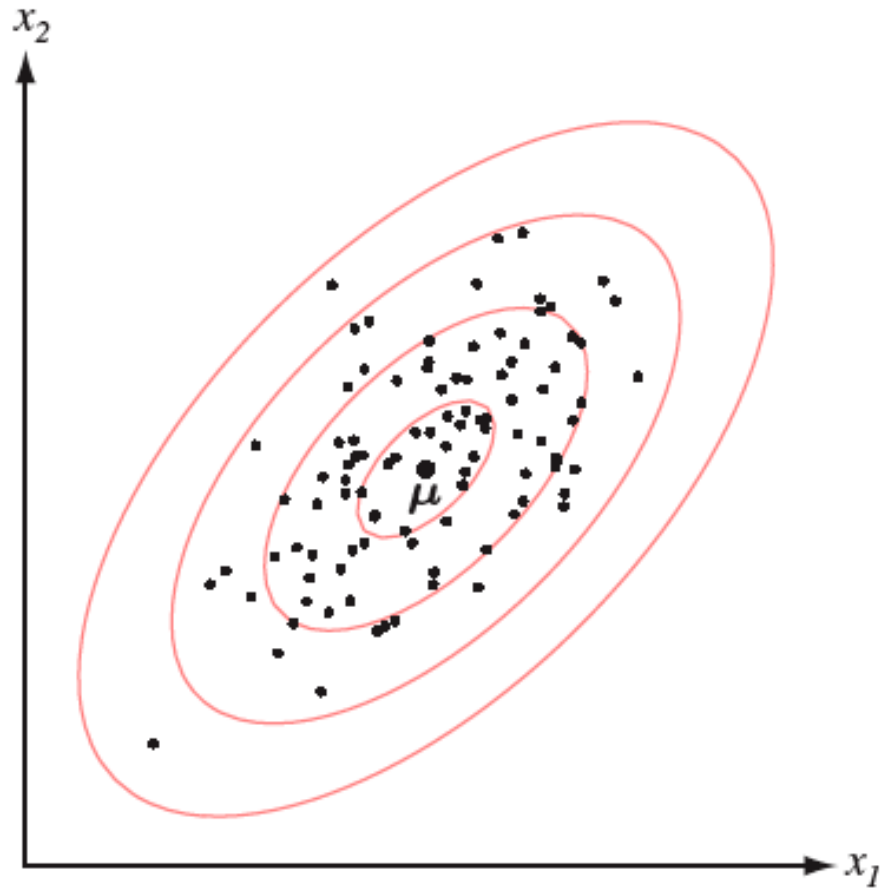
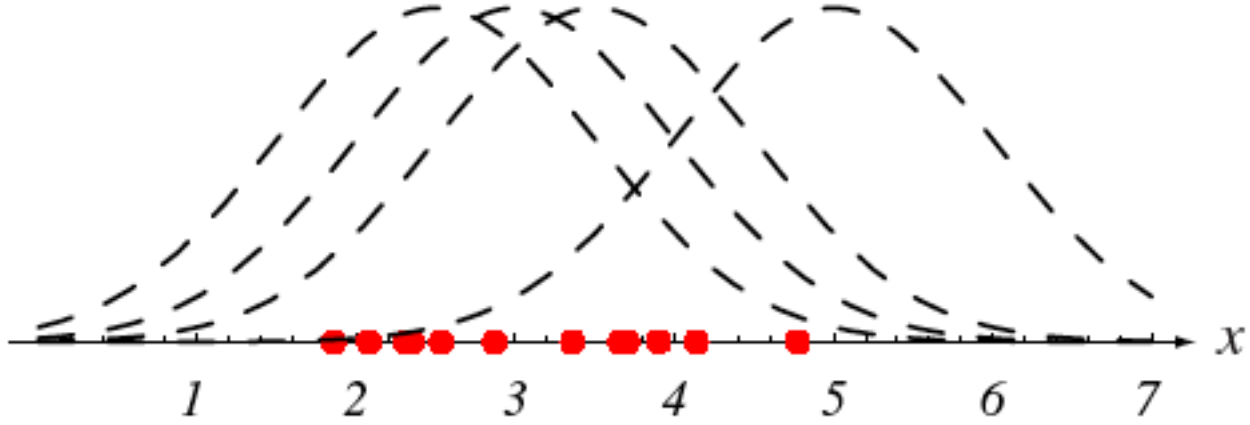


FIGURE 2.9. Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean μ. The ellipses show lines of equal probability density of the Gaussian. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Beispiel Gauss



- Mehrere Gausverteilungen gleicher Varianz aber mit unterschiedlichen Mittelwert
- Stichprobe gezogen aus einer Verteilung mit derselben Varianz

Welche Kurve „passt“ am besten ???

Learning Probabilities

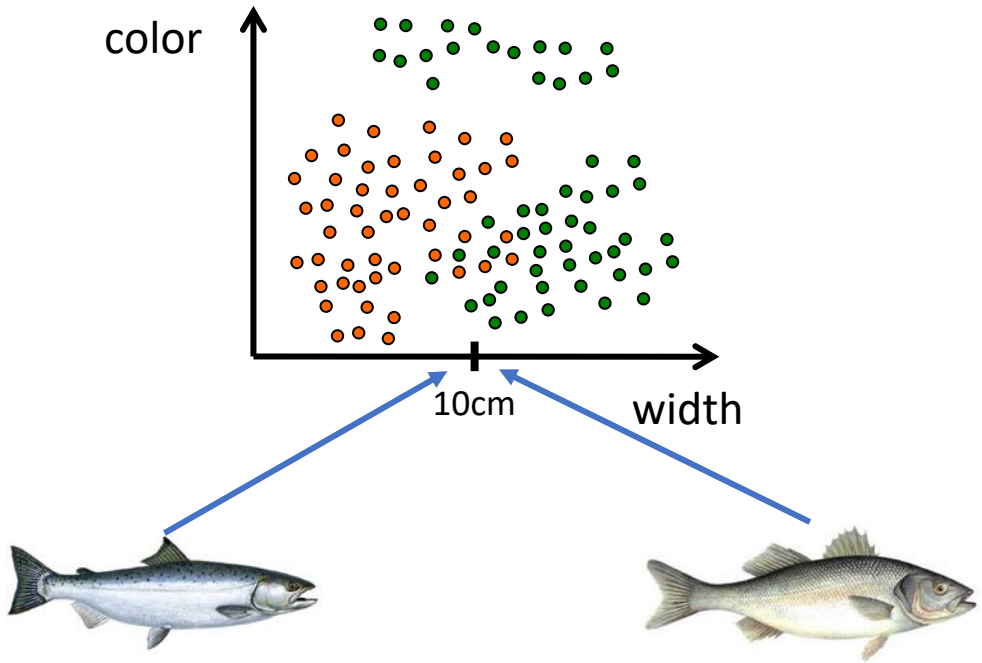
Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$



Parameter: $p(x) = \mathcal{N}(\mu, \sigma^2)$

- μ : *Mittelwert*
- und σ^2: Varianz

Parameterschätzung

$$\mathcal{X} = \{ x^t \}_t \text{ mit } x^t \sim p\,(x)$$

- Parameterschätzung:

  Nehme die Form von $p\,(x \mid \theta)$ an und schätze θ mit Stichprobe $\mathcal{X}$

  z.B., $\mathcal{N}\,(\mu, \sigma^2)$ mit $\theta = \{ \mu, \sigma^2 \}$

- Likelihood von θ aus der Stichprobe $\mathcal{X}$

  $$l\,(\theta|\mathcal{X}) = p\,(\mathcal{X}\,|\theta) = \prod_t p\,(x^t|\theta)$$

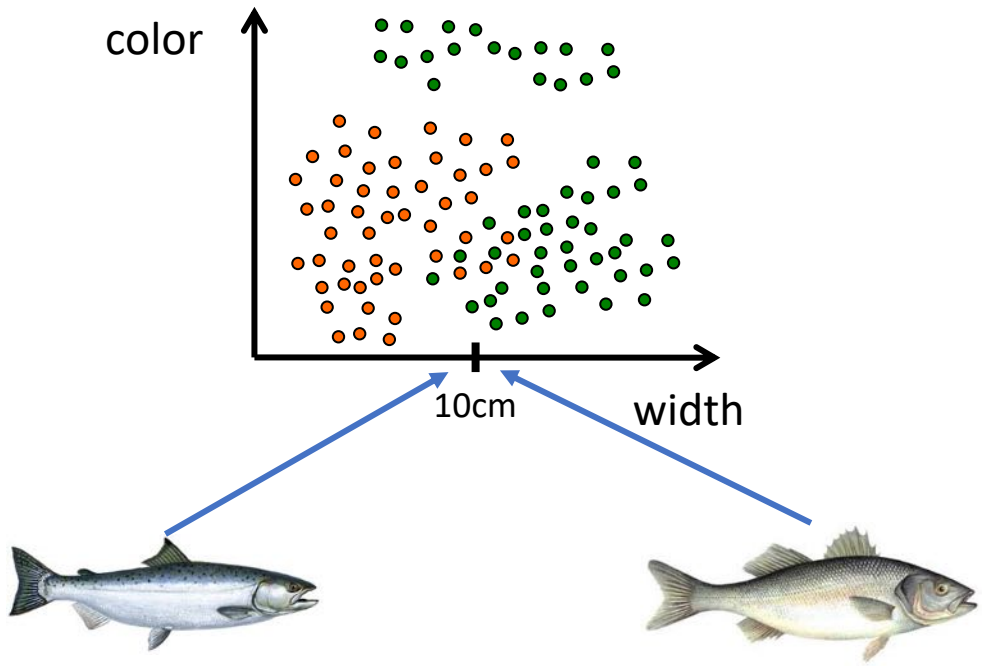
Learning Probabilities

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



Parameter: $p(x) = \mathcal{N}(\mu, \sigma^2)$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- Likelihood von θ aus der Stichprobe $\mathcal{X}$

$$l(\theta|\mathcal{X}) = p(\mathcal{X}|\theta) = \prod_t p(x^t|\theta)$$

Learning Probabilities

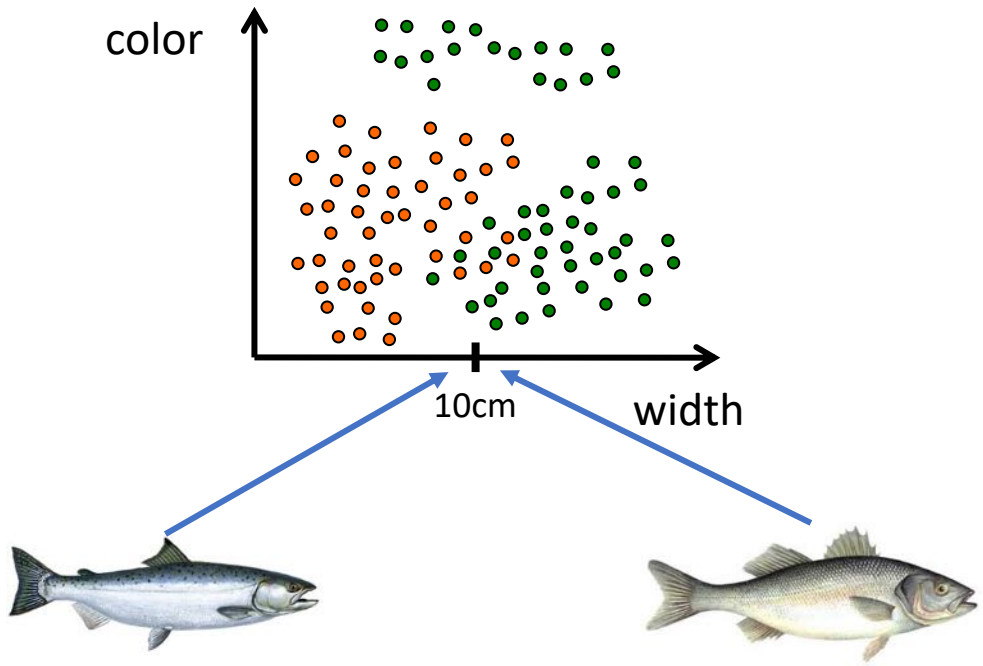
Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function

???



Parameter: $p(x) = \mathcal{N}(\mu, \sigma^2)$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- Likelihood von θ aus der Stichprobe $\mathcal{X}$

$$l(\theta|\mathcal{X}) = p(\mathcal{X}|\theta) = \prod_t p(x^t|\theta)$$

Maximum Likelihood Schätzung

- Likelihood von θ aus der Stichprobe $\mathcal{X}$

  $$l(\theta|\mathcal{X}) = p(\mathcal{X}|\theta) = \prod_t p(x^t|\theta)$$

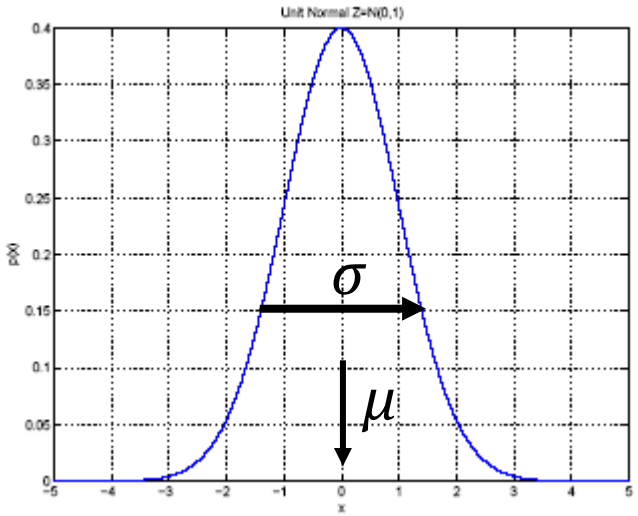
- Log likelihood

  $$\mathcal{L}(\theta|\mathcal{X}) = \log l(\theta|\mathcal{X}) = \sum_t \log p(x^t|\theta)$$

- Maximum likelihood Schätzer (MLE)

  $$\theta^* = \text{argmax}_\theta \mathcal{L}(\theta|\mathcal{X})$$

Gaussian (Normal) Distribution



- $p(x) = \mathcal{N}(\mu, \sigma^2)$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- Maximum Likelyhood Estimation MLE für μ und σ^2:

$$\mathcal{L}(\mu, \sigma^2|\mathcal{X}) = \log p(\mathcal{X}|\mu, \sigma^2) = \sum_t \log p(x^t|\mu, \sigma^2) = -\frac{N}{2}\log 2\pi - N\log\sigma - \frac{\sum_t (x^t-\mu)^2}{2\sigma^2}$$

$$m = \frac{\sum_t x^t}{N}, s^2 = \frac{\sum_t (x^t-m)^2}{N}$$

Example

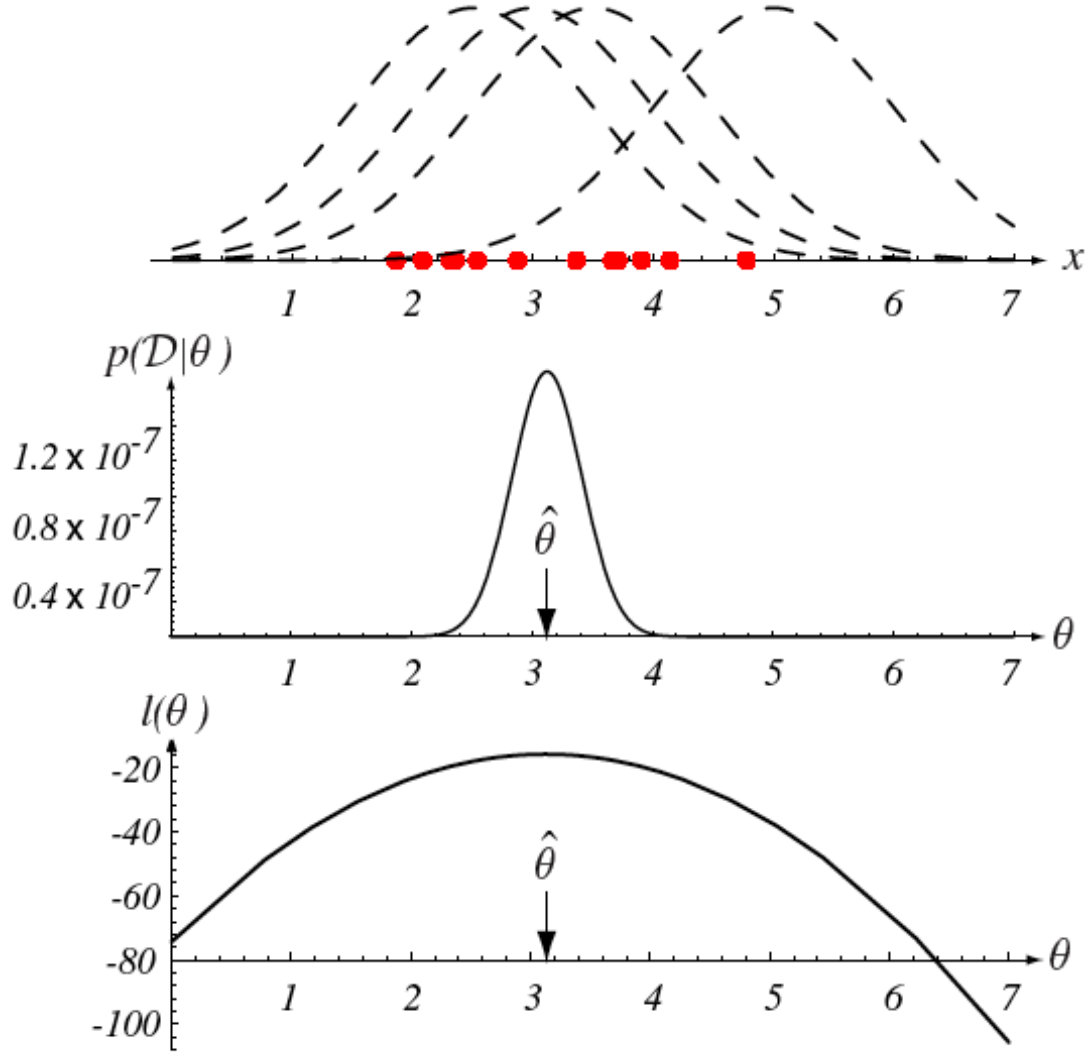
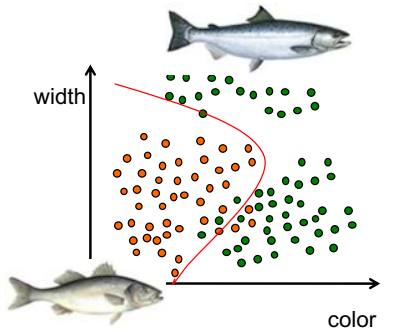


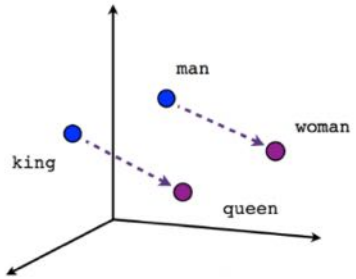
FIGURE 3.1. The top graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four of the infinite number of candidate source distributions are shown in dashed lines. The middle figure shows the likelihood $p(\mathcal{D}|\theta)$ as a function of the mean. If we had a very large number of training points, this likelihood would be very narrow. The value that maximizes the likelihood is marked $\hat{\theta}$; it also maximizes the logarithm of the likelihood—that is, the log-likelihood $l(\theta)$, shown at the bottom. Note that even though they look similar, the likelihood $p(\mathcal{D}|\theta)$ is shown as a function of θ whereas the conditional density $p(x|\theta)$ is shown as a function of x. Furthermore, as a function of θ, the likelihood $p(\mathcal{D}|\theta)$ is not a probability density function and its area has no significance. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*.

Result: A target function derived from the sample
to do „something usefull“

on data points that are related to the original space but as such have not been seen before !!!

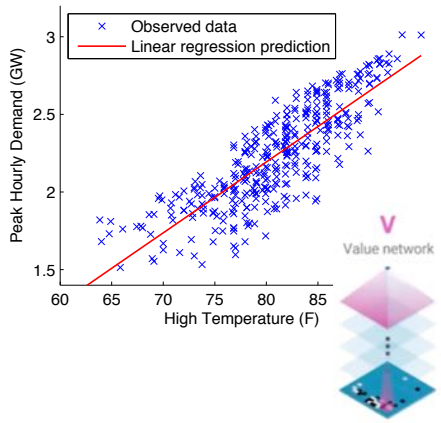


classification



Male-Female

(mapping to)
new representation



value prediction

generation of new
samples (completion)

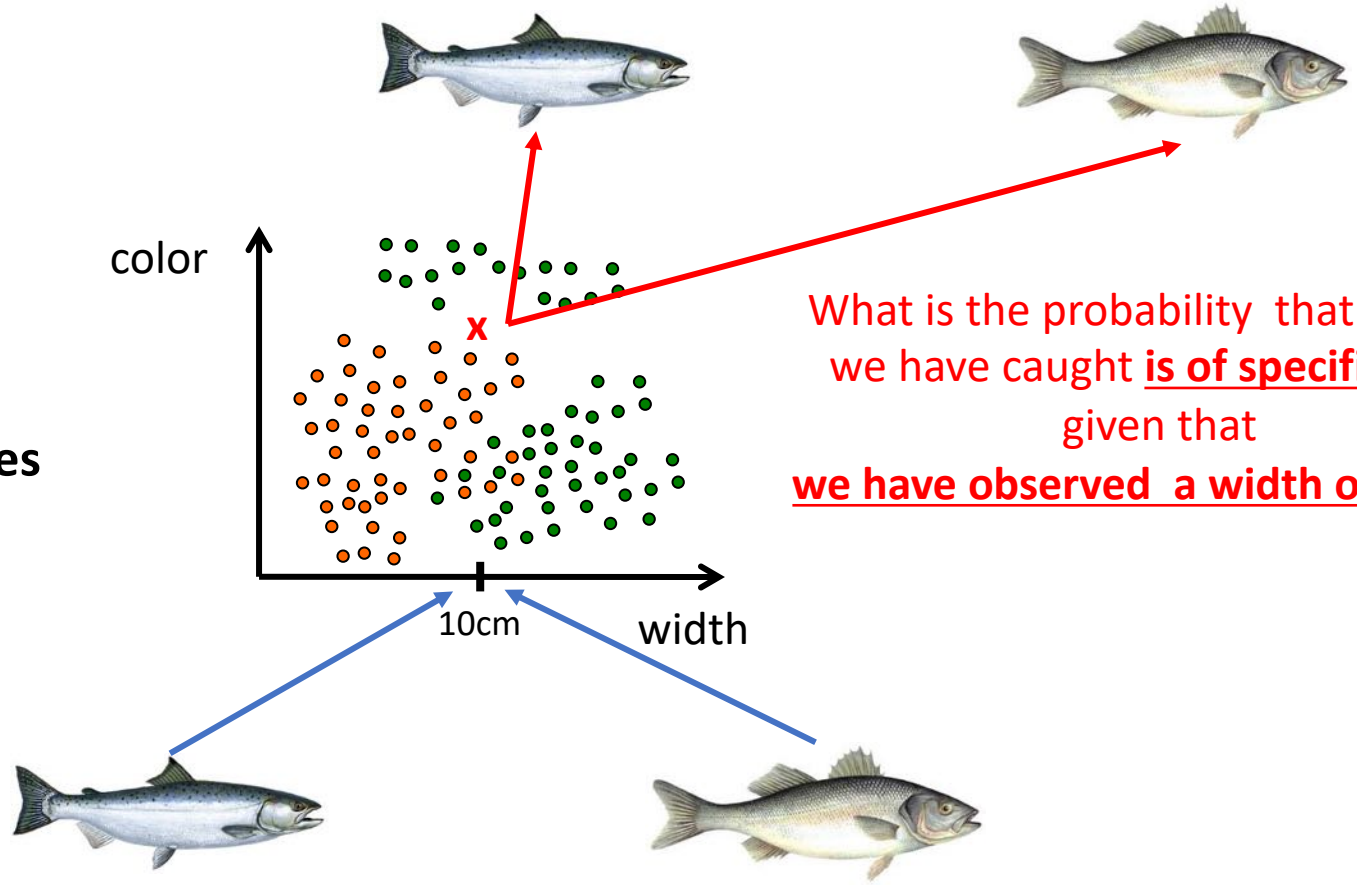
Probability Estimation and Classification

Machine Learning

1. describe all possible shapes through parameters (opening, angle, position,...)

2. define a function that, given a set of paramters and the training data computes an error value (error/cost function)

3. use an appropriate optimization techniques to find parameter values that minimize the error function



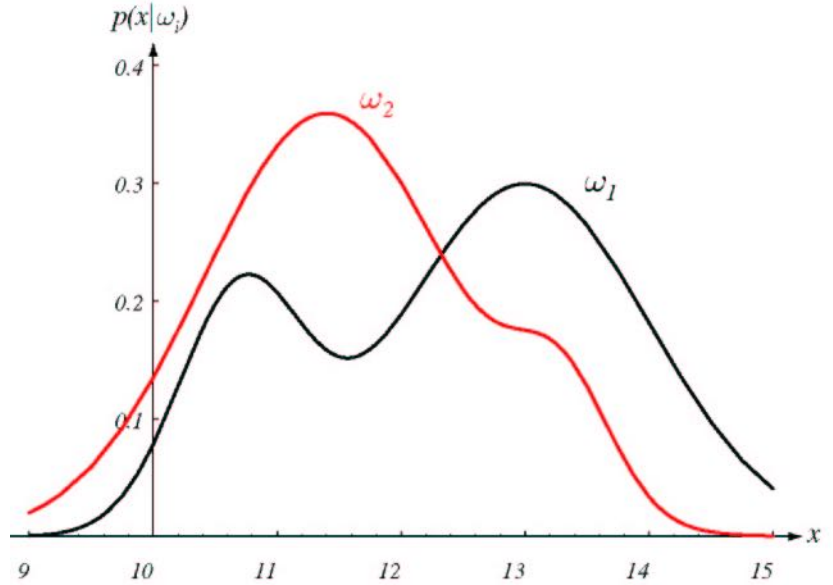
What is the probability that the fish we have caught **is of specific class** given that **we have observed a width of 10cm ?**

What is the probability of a randomly caught fish having a width of 10cm ?

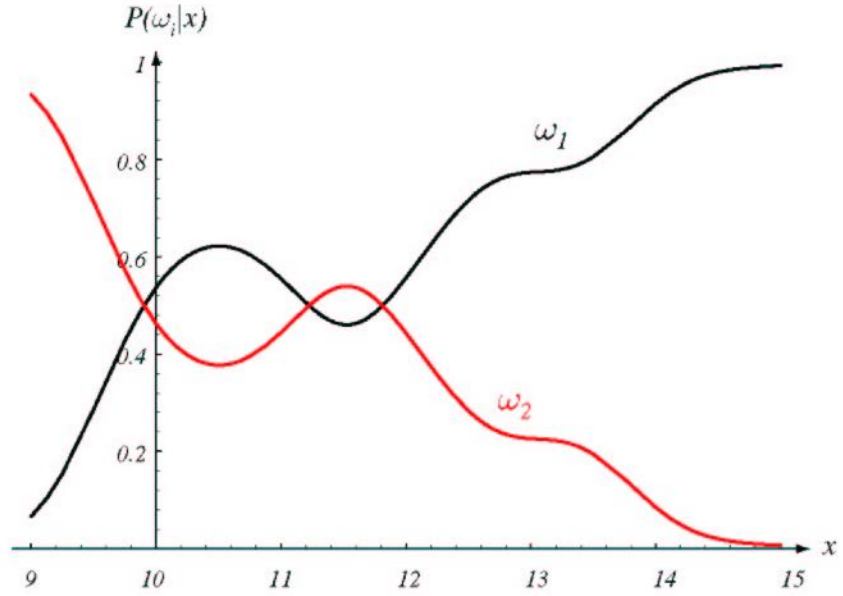
- obviously different for each class

Beispiel: Bayesische Regel

- Gegeben: die Wahrscheinlichkeiten für das Beobachten verschiedener Werte von x für zwei verschiedene Klassen



- Gesucht: die Wahrscheinlichkeiten für das Vorliegen der jeweiligen Klassen bei Beobachtung verschiedener Werte von x
  - Beide Klassen gleich wahrscheinlich



A priori und a posteriori

- A priori: vor der Messung
  - Wahrscheinlichkeit einen bestimmten Messwert zu Beobachten, wenn eine bestimmte Klasse vorliegt
  - z.B. Wahrscheinlichkeit eine Zahl zu beobachten, wenn eine falsche bzw. Echte Münze geworfen wird
- A posteriori:
  - Wahrscheinlichkeit, dass eine bestimmte Klasse vorliegt, wenn ein bestimmter Wert gemessen wurde
  - z.B Wahrscheinlichkeit für falsche Münze wenn zwei mal Zahl beobachtet wurde
  - Hängt von der grundsätzliche Wahrscheinlichkeit, dass es eine falsche Münze gibt, ab !

Bayesische Entscheidungstheorie

- Bekannt sind in der Regel die a priori Verteilungen
  - $P(\omega_i)$ Wahrscheinlichkeit für das Auftreten der Klasse
  - $P(\vec{x} \mid \omega_i)$ Wahrscheinlichkeit für Vektor $\vec{x}$, wenn Klasse ω_i vorliegt

Die Bayesische Regel erlaubt die Berechnung der a posteriori Wahrscheinlichkeit $P(\omega_j \mid \vec{x})$ für Klasse ω_j , wenn Vektor $\vec{x}$ vorliegt

$$P(\omega_i \mid \vec{x}) = \frac{P(\vec{x} \mid \omega_i) P(\omega_i)}{P(\vec{x})}$$

Ziel des Klassifikator Designs

Es soll der Erwartungswert der Gesamtkosten für Fehlklassifikation minimiert werden.

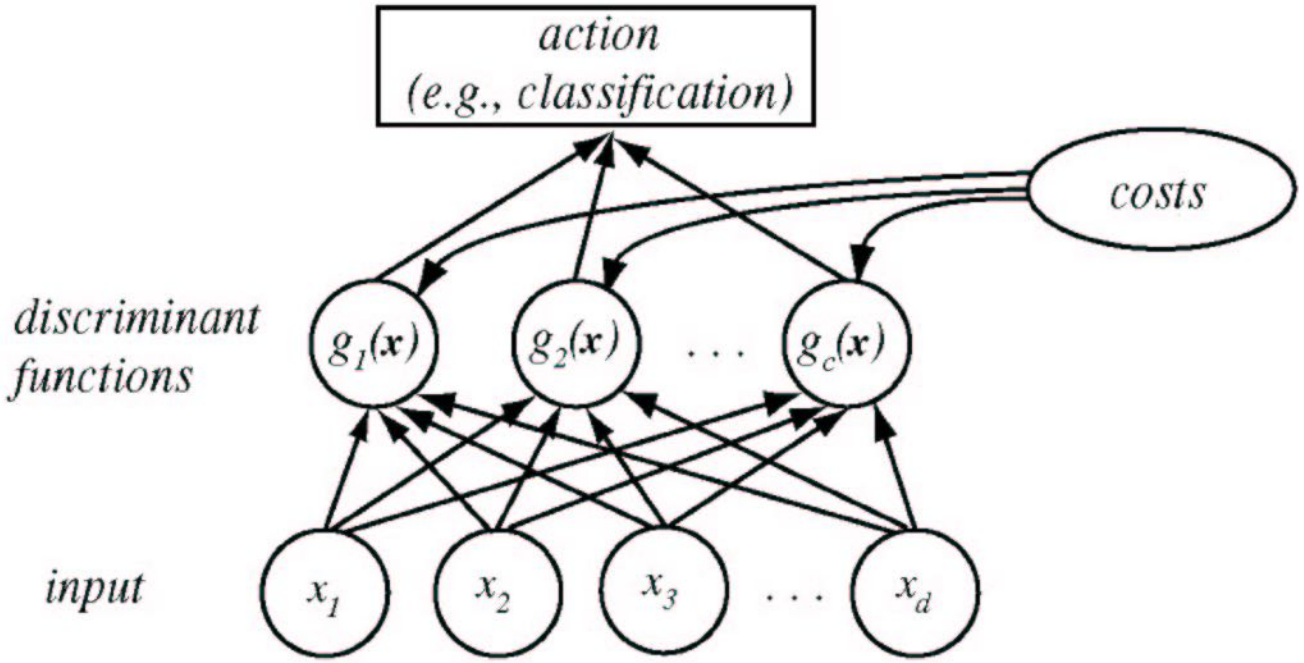
- nicht alle Fehlklassifikationen sind gleich ‚schlimm'
  - z.B. wird das EKG eines kranken Patienten als gesund erkannt, so ist es wesentlich gravierender als anderes herum !
- Zu minimieren ist die Risikofunktion

$$R(c_i \mid \vec{x}) = \sum_{j=1}^{c} \lambda(c_i \mid \omega_j) P(\omega_j \mid \vec{x})$$

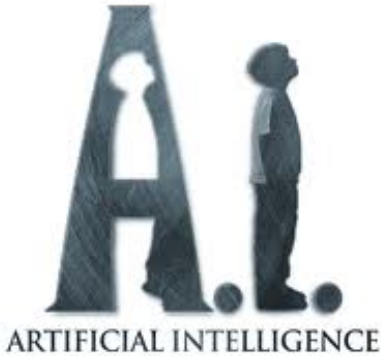
  - $\lambda(c_i \mid \omega_j)$ Kosten für die Erkennung der Klasse c_i, wenn in Wirklichkeit die Klasse ω_j vorliegt
  - $P(\omega_j \mid \vec{x})$ Wahrscheinlichkeit für Klasse ω_j, wenn Vektor $\vec{x}$ vorliegt

Unterscheidungsfunktionen (discriminant functions)

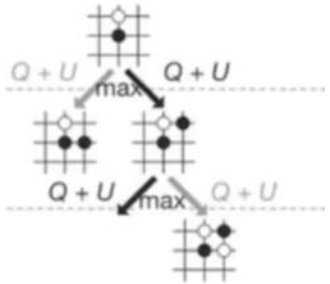
- Die Klassifikation kann durch Unterscheidungsfunktionen g(x) im folgenden Netzwerk durchgeführt werden
  - Für jede Klasse existiert eine eigene Funktion (im allgemeinen gleich der Risikofunktion)
  - Es wird die Klasse gewählt, für die die Funktion den kleinsten Wert hat

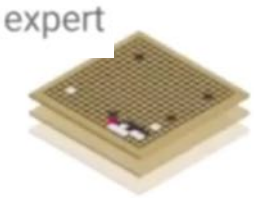


That‘s all



highly efficient complex (1) search and (2) knowledge representation





(3) learning: statistical analysis and optimization