

CLASSIFICATION— ADVANCED

Hari Sundaram

hs1@illinois.edu

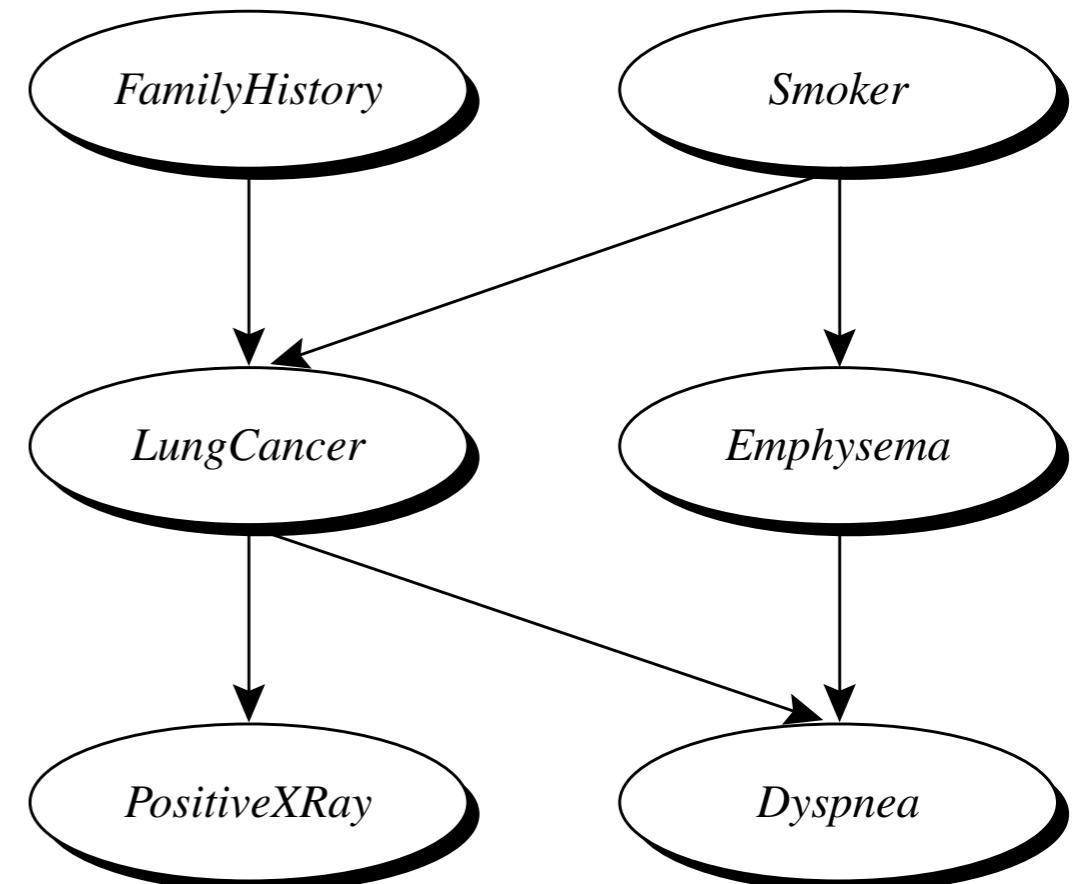
<http://sundaram.cs.illinois.edu>

adapted from slides by Jiawei Han and Kevin Chang

BAYESIAN BELIEF NETWORKS

Support Vector Machines Neural Networks Etc

Summary

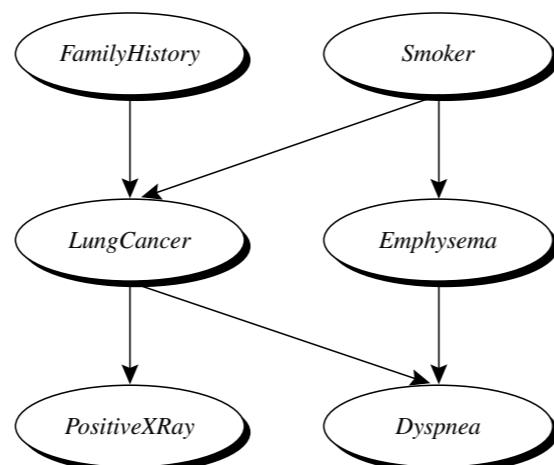


a set of conditional
probability tables (CPTs)

	FH, S	$FH, \sim S$	$\sim FH, S$	$\sim FH, \sim S$
LC	0.8	0.5	0.7	0.1
$\sim LC$	0.2	0.5	0.3	0.9

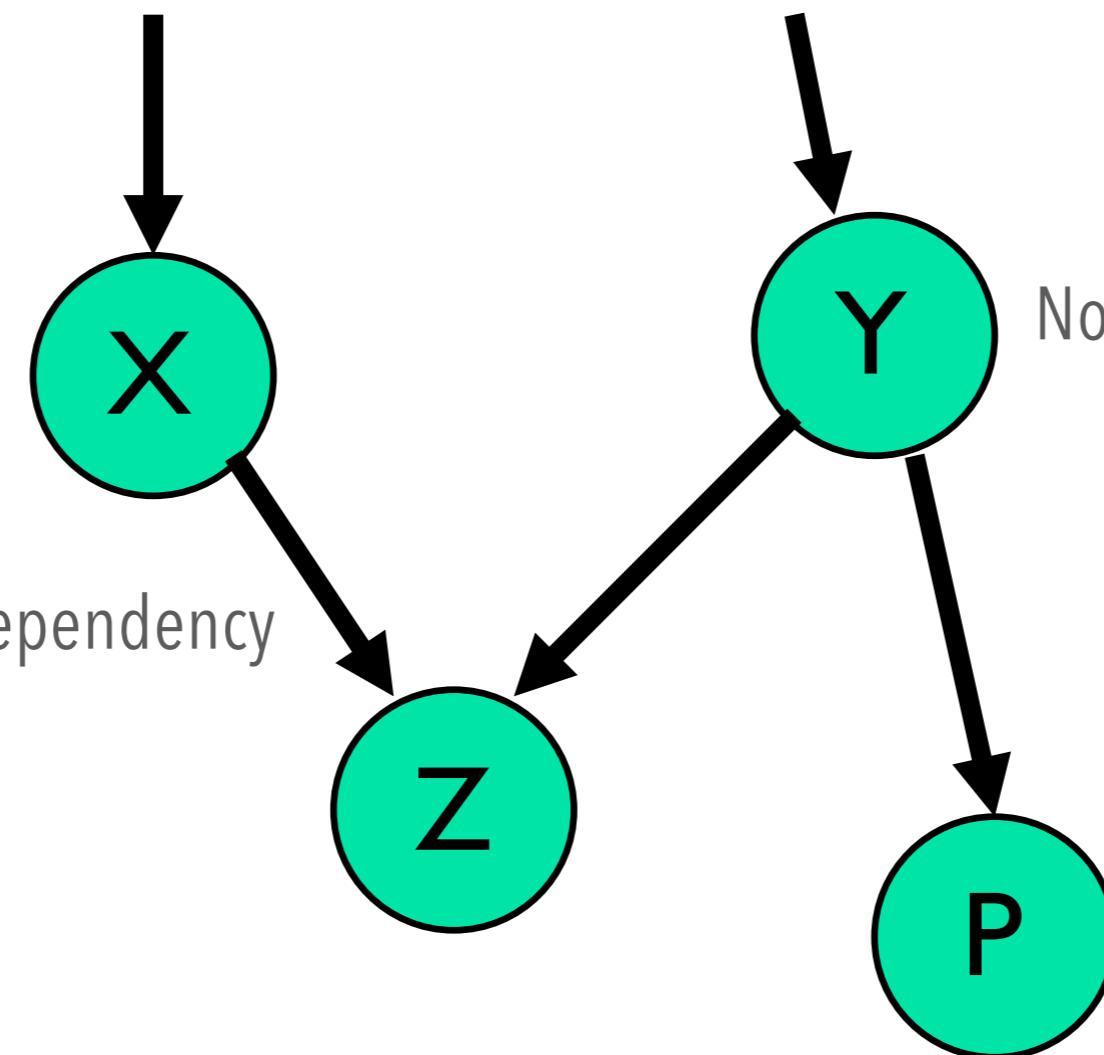
bayesian belief networks

A directed acyclic
graph (called a
structure)



Bayesian belief networks
allow class conditional
independencies between
subsets of variables

X and Y are the parents of Z, and Y is the parent of P

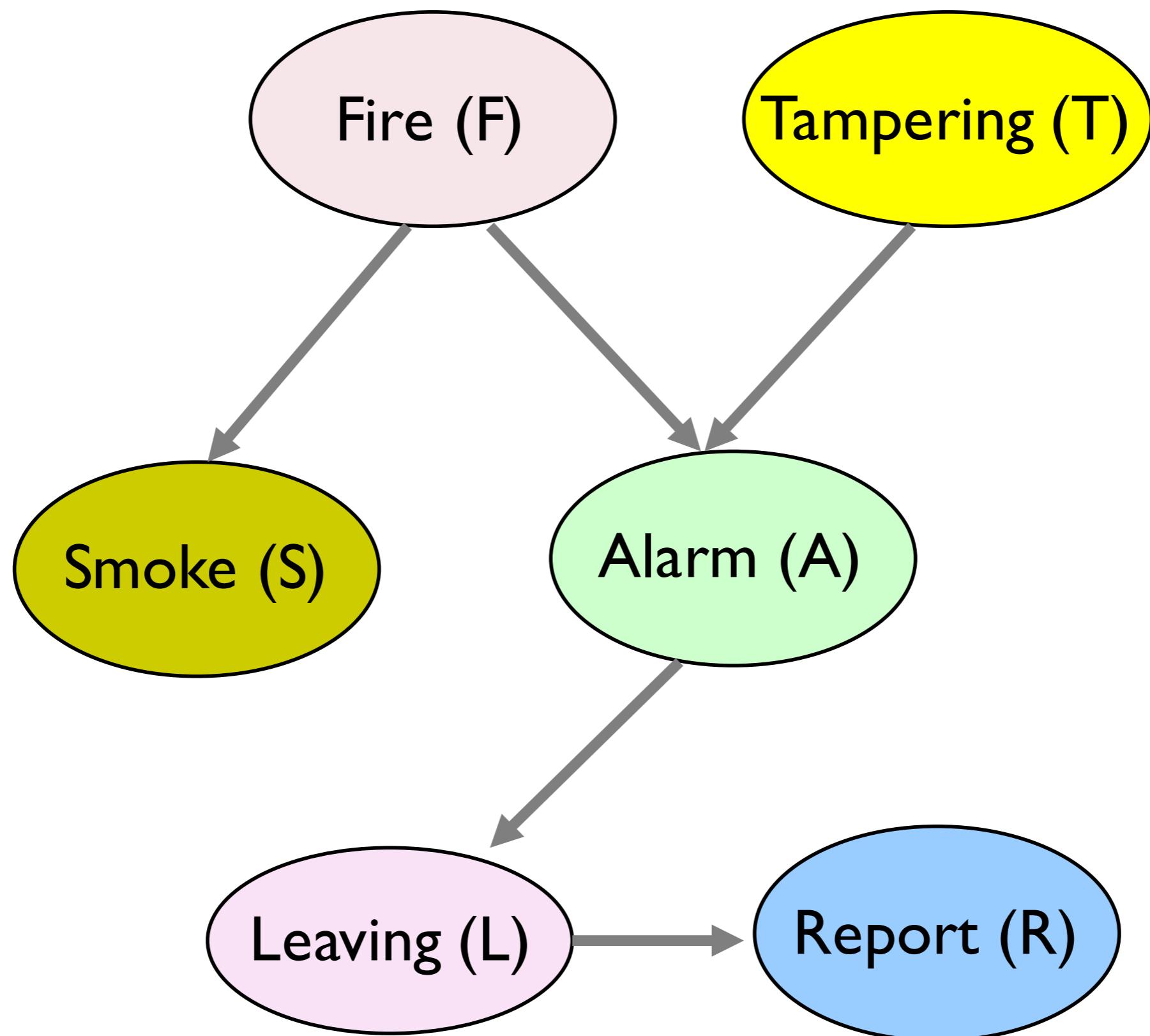


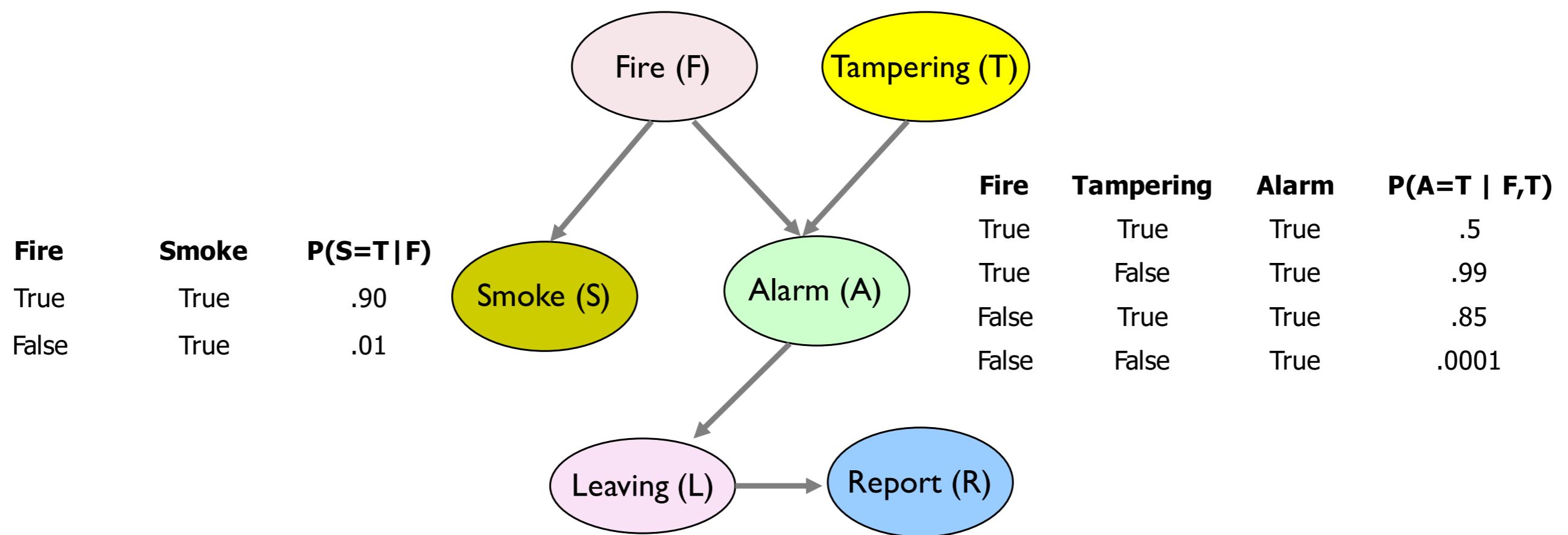
Links: dependency

Nodes: random variables

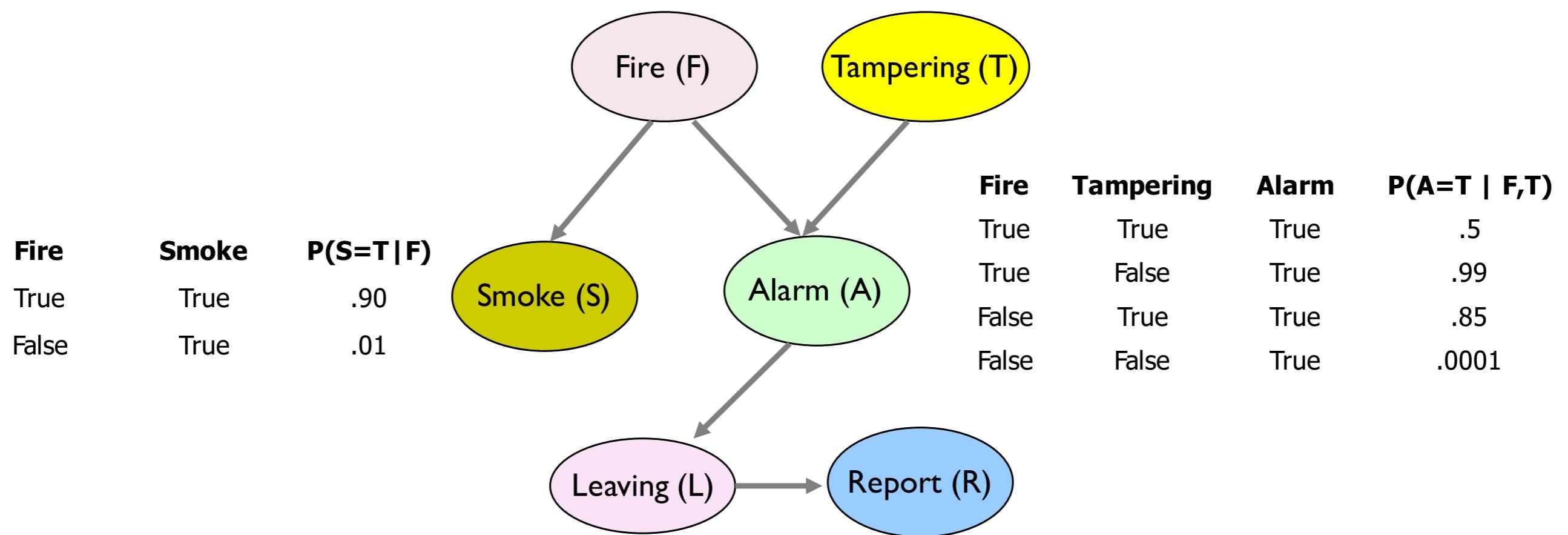
no loops/cycles

No dependency between Z and P





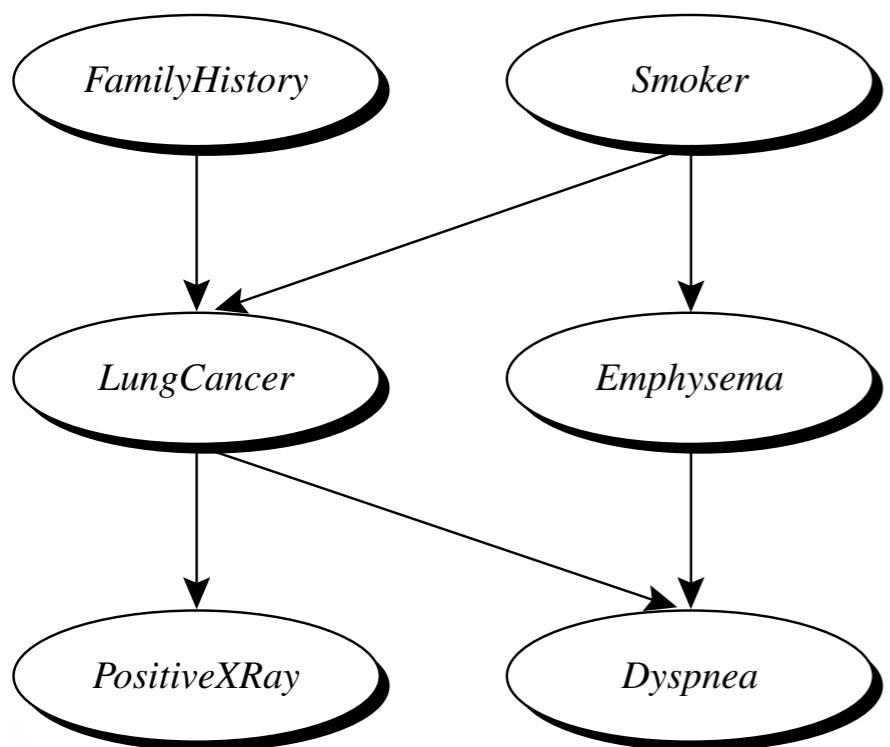
CPT shows the conditional probability for each possible combination of its parents



CPT shows the conditional probability for each possible combination of its parents

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | Parents(x_i))$$

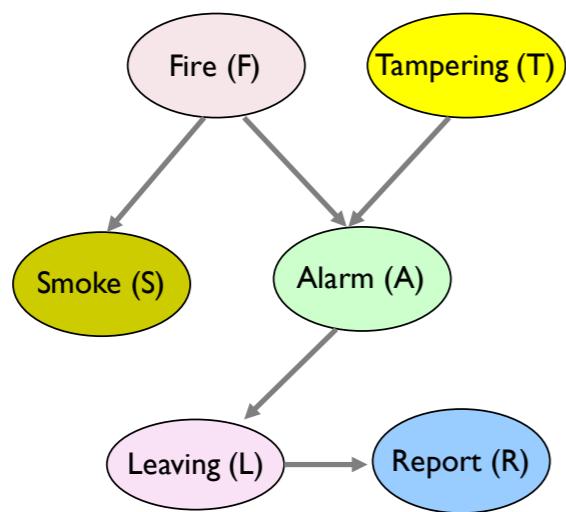
does a belief network allow for causal inference?



**how do we construct
these networks?**



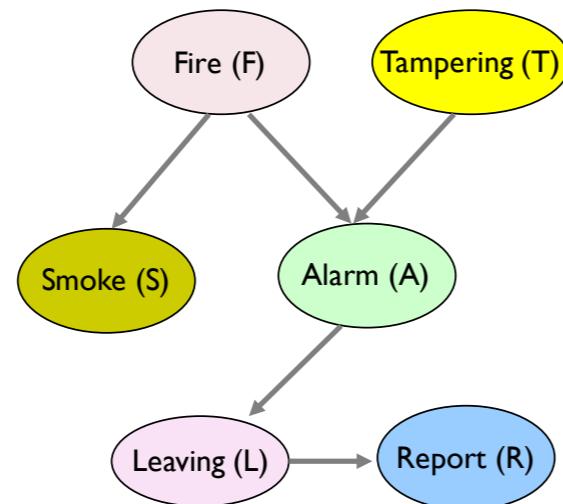
Subjective construction



Identification of (direct) causal structure

People are quite good at identifying direct causes from a given set of variables & whether the set contains all relevant direct causes

Markovian assumption

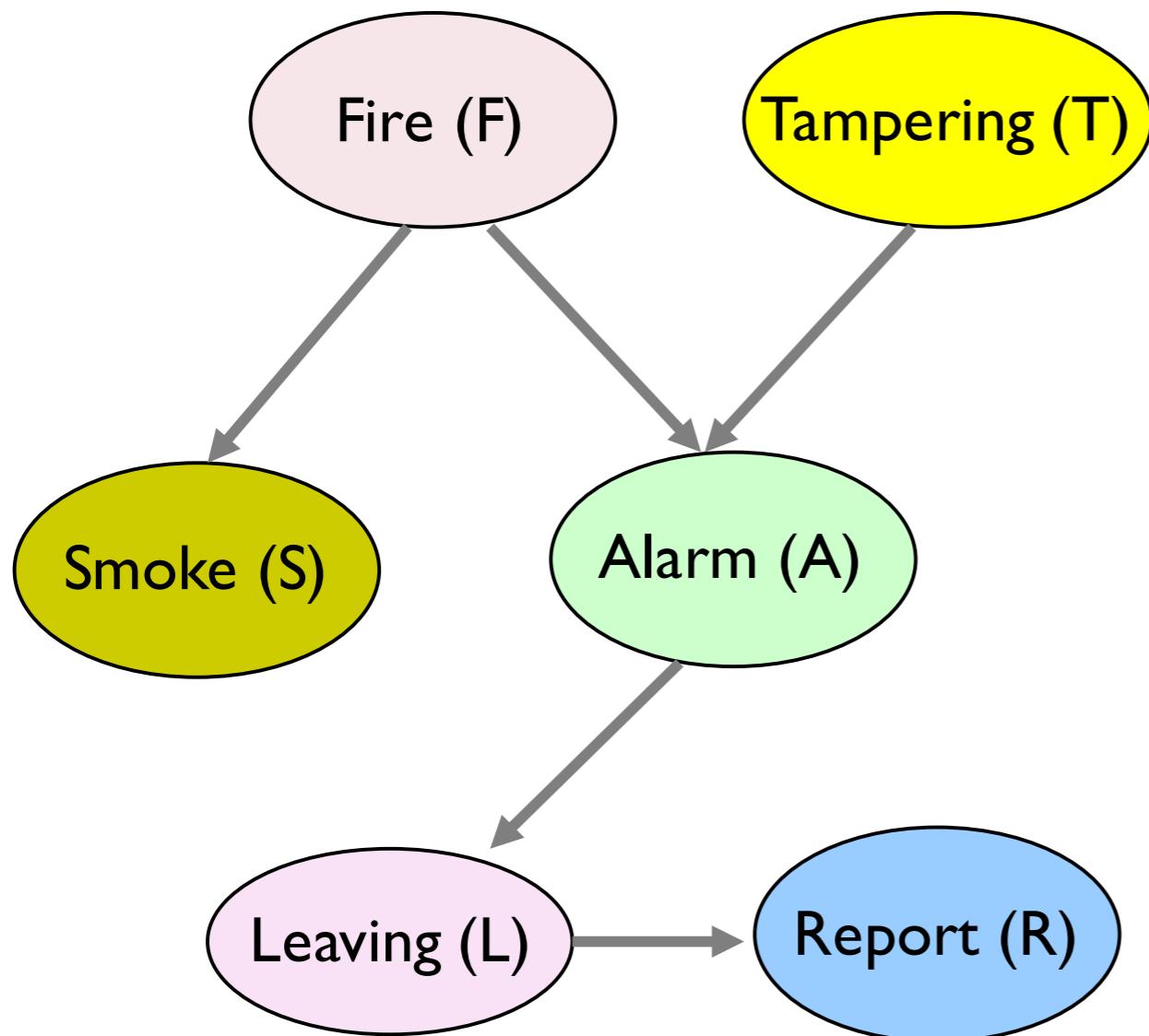


Each variable becomes independent of its non-effects once its direct causes are known

ID	LOC_NBR	FIRST_NAME	LAST_NAME	SEX	BIRTH_DATE	DEATH_DATE	EDUCATION_LEVEL	EDUCATION_YEAR	COUNTRY	AGE	WEIGHT	HEIGHT	CHOLESTEROL	BLOOD_PRESSURE	SMOKING_STATUS	DIABETES
10001	34001	Los Angeles	John	M	1950-01-01		Elementary school	1950	USA	28	105	67	200	100	Yes	No
10002	34002	Santa	Anne	F	1951-02-01		Elementary school	1951	USA	29	106	68	201	101	No	No
10003	34003	Steve	Mark	M	1952-03-01		Elementary school	1952	USA	30	107	69	202	102	No	No
10004	34004	Louis	Karen	F	1953-04-01		Elementary school	1953	USA	31	108	70	203	103	No	No
10005	34005	Jeffrey	Mike	M	1954-05-01		Elementary school	1954	USA	32	109	71	204	104	No	No
10006	34006	John	David	M	1955-06-01		Elementary school	1955	USA	33	110	72	205	105	No	No
10007	34007	Chris	Mark	M	1956-07-01		Elementary school	1956	USA	34	111	73	206	106	No	No
10008	34008	David	Steve	M	1957-08-01		Elementary school	1957	USA	35	112	74	207	107	No	No
10009	34009	Mark	David	M	1958-09-01		Elementary school	1958	USA	36	113	75	208	108	No	No
10010	34010	Mark	Mark	M	1959-10-01		Elementary school	1959	USA	37	114	76	209	109	No	No
10011	34011	Tom	John	M	1960-11-01		Elementary school	1960	USA	38	115	77	210	110	No	No
10012	34012	Tom	John	M	1961-12-01		Elementary school	1961	USA	39	116	78	211	111	No	No
10013	34013	Tom	John	M	1962-01-01		Elementary school	1962	USA	40	117	79	212	112	No	No
10014	34014	Tom	John	M	1963-02-01		Elementary school	1963	USA	41	118	80	213	113	No	No
10015	34015	Tom	John	M	1964-03-01		Elementary school	1964	USA	42	119	81	214	114	No	No
10016	34016	Tom	John	M	1965-04-01		Elementary school	1965	USA	43	120	82	215	115	No	No
10017	34017	Tom	John	M	1966-05-01		Elementary school	1966	USA	44	121	83	216	116	No	No
10018	34018	Tom	John	M	1967-06-01		Elementary school	1967	USA	45	122	84	217	117	No	No
10019	34019	Tom	John	M	1968-07-01		Elementary school	1968	USA	46	123	85	218	118	No	No
10020	34020	Tom	John	M	1969-08-01		Elementary school	1969	USA	47	124	86	219	119	No	No
10021	34021	Tom	John	M	1970-09-01		Elementary school	1970	USA	48	125	87	220	120	No	No
10022	34022	Tom	John	M	1971-10-01		Elementary school	1971	USA	49	126	88	221	121	No	No
10023	34023	Tom	John	M	1972-11-01		Elementary school	1972	USA	50	127	89	222	122	No	No
10024	34024	Tom	John	M	1973-12-01		Elementary school	1973	USA	51	128	90	223	123	No	No
10025	34025	Tom	John	M	1974-01-01		Elementary school	1974	USA	52	129	91	224	124	No	No
10026	34026	Tom	John	M	1975-02-01		Elementary school	1975	USA	53	130	92	225	125	No	No
10027	34027	Tom	John	M	1976-03-01		Elementary school	1976	USA	54	131	93	226	126	No	No
10028	34028	Tom	John	M	1977-04-01		Elementary school	1977	USA	55	132	94	227	127	No	No
10029	34029	Tom	John	M	1978-05-01		Elementary school	1978	USA	56	133	95	228	128	No	No
10030	34030	Tom	John	M	1979-06-01		Elementary school	1979	USA	57	134	96	229	129	No	No
10031	34031	Tom	John	M	1980-07-01		Elementary school	1980	USA	58	135	97	230	130	No	No
10032	34032	Tom	John	M	1981-08-01		Elementary school	1981	USA	59	136	98	231	131	No	No
10033	34033	Tom	John	M	1982-09-01		Elementary school	1982	USA	60	137	99	232	132	No	No
10034	34034	Tom	John	M	1983-10-01		Elementary school	1983	USA	61	138	100	233	133	No	No
10035	34035	Tom	John	M	1984-11-01		Elementary school	1984	USA	62	139	101	234	134	No	No
10036	34036	Tom	John	M	1985-12-01		Elementary school	1985	USA	63	140	102	235	135	No	No
10037	34037	Tom	John	M	1986-01-01		Elementary school	1986	USA	64	141	103	236	136	No	No
10038	34038	Tom	John	M	1987-02-01		Elementary school	1987	USA	65	142	104	237	137	No	No
10039	34039	Tom	John	M	1988-03-01		Elementary school	1988	USA	66	143	105	238	138	No	No
10040	34040	Tom	John	M	1989-04-01		Elementary school	1989	USA	67	144	106	239	139	No	No
10041	34041	Tom	John	M	1990-05-01		Elementary school	1990	USA	68	145	107	240	140	No	No
10042	34042	Tom	John	M	1991-06-01		Elementary school	1991	USA	69	146	108	241	141	No	No
10043	34043	Tom	John	M	1992-07-01		Elementary school	1992	USA	70	147	109	242	142	No	No
10044	34044	Tom	John	M	1993-08-01		Elementary school	1993	USA	71	148	110	243	143	No	No
10045	34045	Tom	John	M	1994-09-01		Elementary school	1994	USA	72	149	111	244	144	No	No
10046	34046	Tom	John	M	1995-10-01		Elementary school	1995	USA	73	150	112	245	145	No	No
10047	34047	Tom	John	M	1996-11-01		Elementary school	1996	USA	74	151	113	246	146	No	No
10048	34048	Tom	John	M	1997-12-01		Elementary school	1997	USA	75	152	114	247	147	No	No
10049	34049	Tom	John	M	1998-01-01		Elementary school	1998	USA	76	153	115	248	148	No	No
10050	34050	Tom	John	M	1999-02-01		Elementary school	1999	USA	77	154	116	249	149	No	No
10051	34051	Tom	John	M	2000-03-01		Elementary school	2000	USA	78	155	117	250	150	No	No
10052	34052	Tom	John	M	2001-04-01		Elementary school	2001	USA	79	156	118	251	151	No	No
10053	34053	Tom	John	M	2002-05-01		Elementary school	2002	USA	80	157	119	252	152	No	No
10054	34054	Tom	John	M	2003-06-01		Elementary school	2003	USA	81	158	120	253	153	No	No
10055	34055	Tom	John	M	2004-07-01		Elementary school	2004	USA	82	159	121	254	154	No	No
10056	34056	Tom	John	M	2005-08-01		Elementary school	2005	USA	83	160	122	255	155	No	No
10057	34057	Tom	John	M	2006-09-01		Elementary school	2006	USA	84	161	123	256	156	No	No
10058	34058	Tom	John	M	2007-10-01		Elementary school	2007	USA	85	162	124	257	157	No	No
10059	34059	Tom	John	M	2008-11-01		Elementary school	2008	USA	86	163	125	258	158	No	No
10060	34060	Tom	John	M	2009-12-01		Elementary school	2009	USA	87	164	126	259	159	No	No
10061	34061	Tom	John	M	2010-01-01		Elementary school	2010	USA	88	165	127	260	160	No	No
10062	34062	Tom	John	M	2011-02-01		Elementary school	2011								

TRAINING A BBN

.....



D. Heckerman. [A Tutorial on Learning with Bayesian Networks](#). In Learning in Graphical Models, M. Jordan, ed. MIT Press, 1999.

Scenario 1: Given both the network structure and all variables observable: compute only the CPT entries

Scenario 2: Network structure known, some variables hidden: gradient descent (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function

Weights are initialized to random probability values

At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking

Weights are updated at each iteration & converge to local optimum

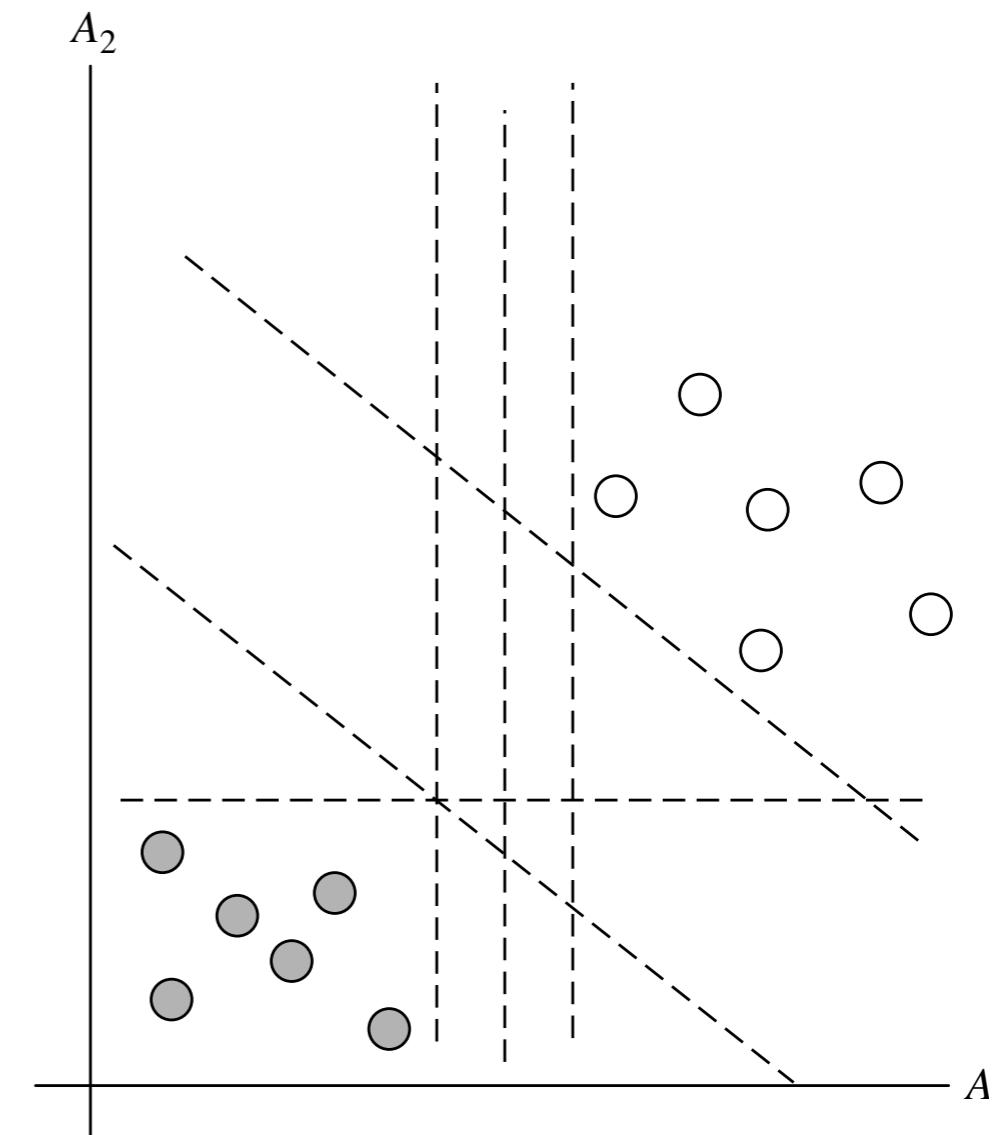
Scenario 3: Network structure unknown, all variables observable: search through the model space to reconstruct network topology

Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose

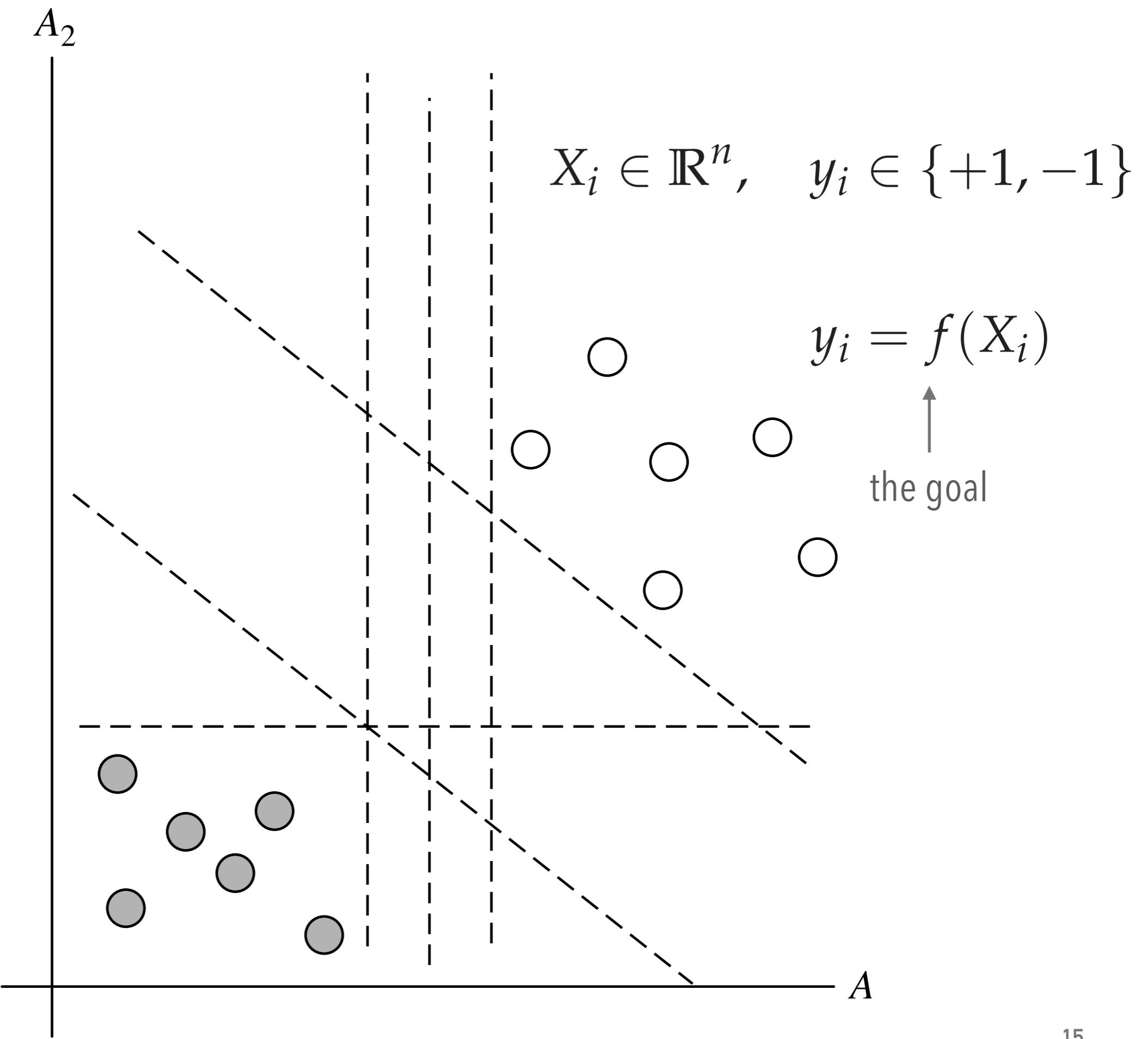
SUPPORT VECTOR MACHINES

.....
Bayesian Belief Networks Neural Networks k-NN

Multiclass Classification Summary

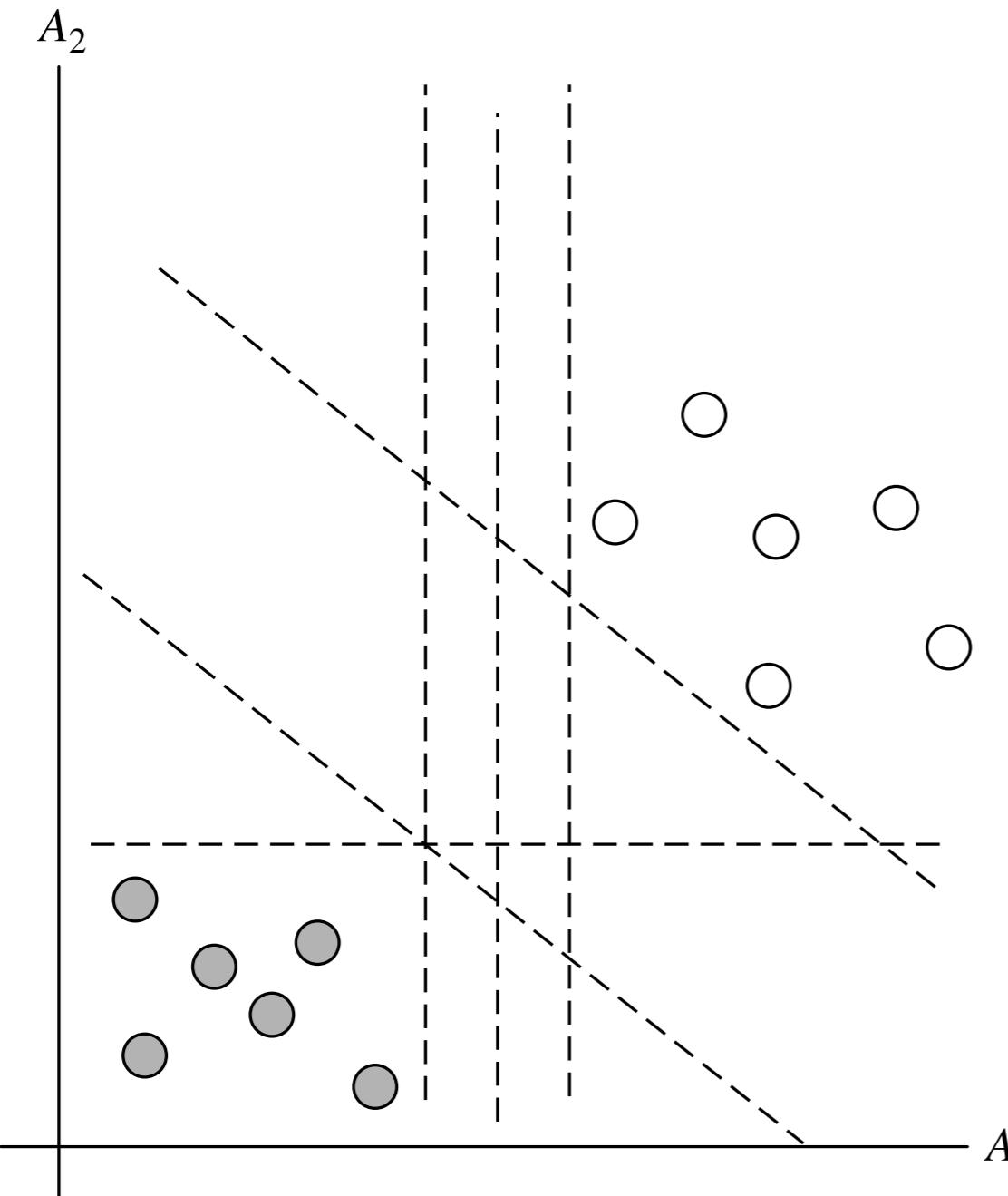


A_2



ADVANTAGES

.....



Prediction accuracy is generally high

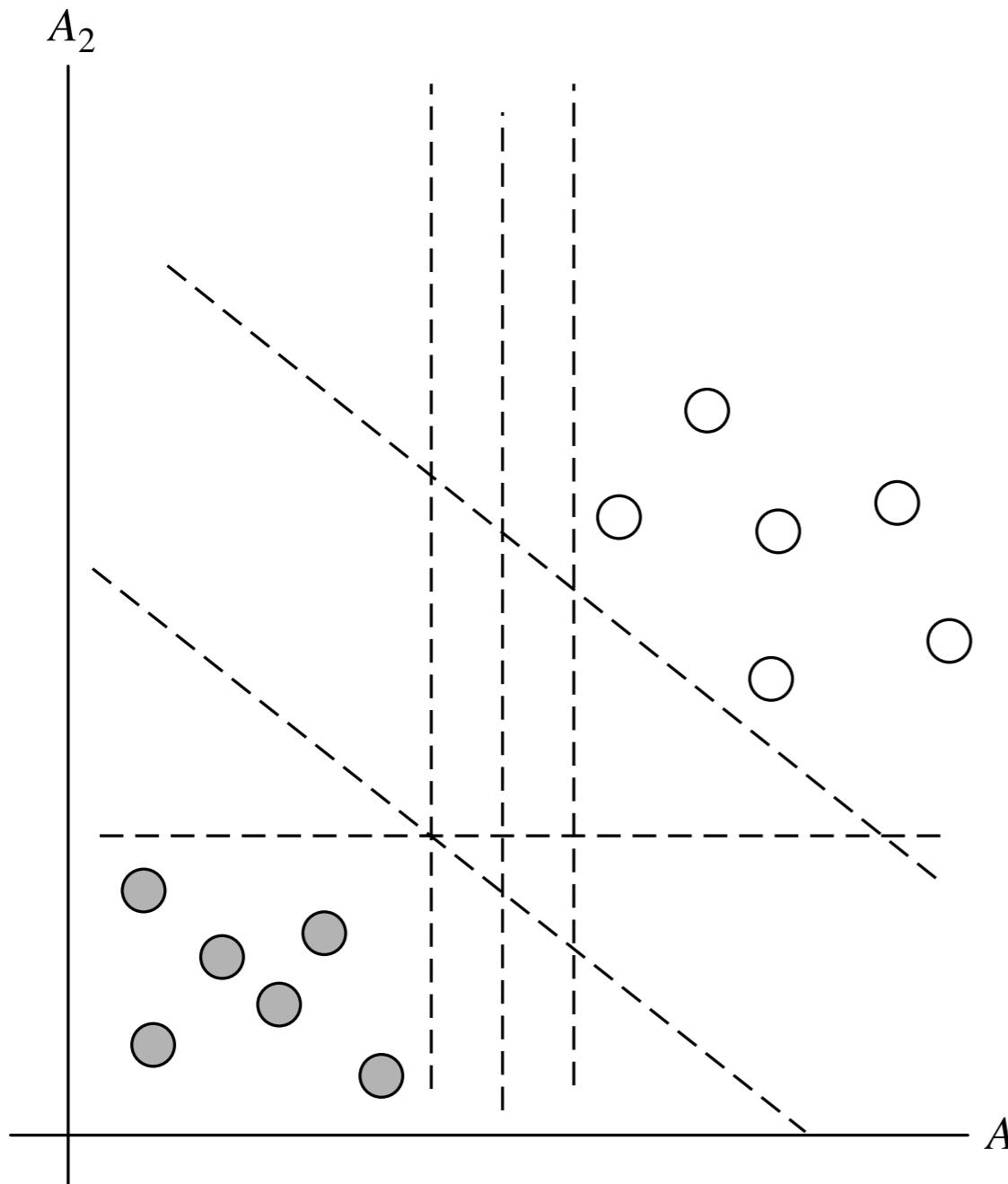
As compared to Bayesian methods – in general

Robust, works when training examples contain errors

Fast evaluation of the learned target function

Bayesian networks are normally slow

CRITICISMS



Long training time

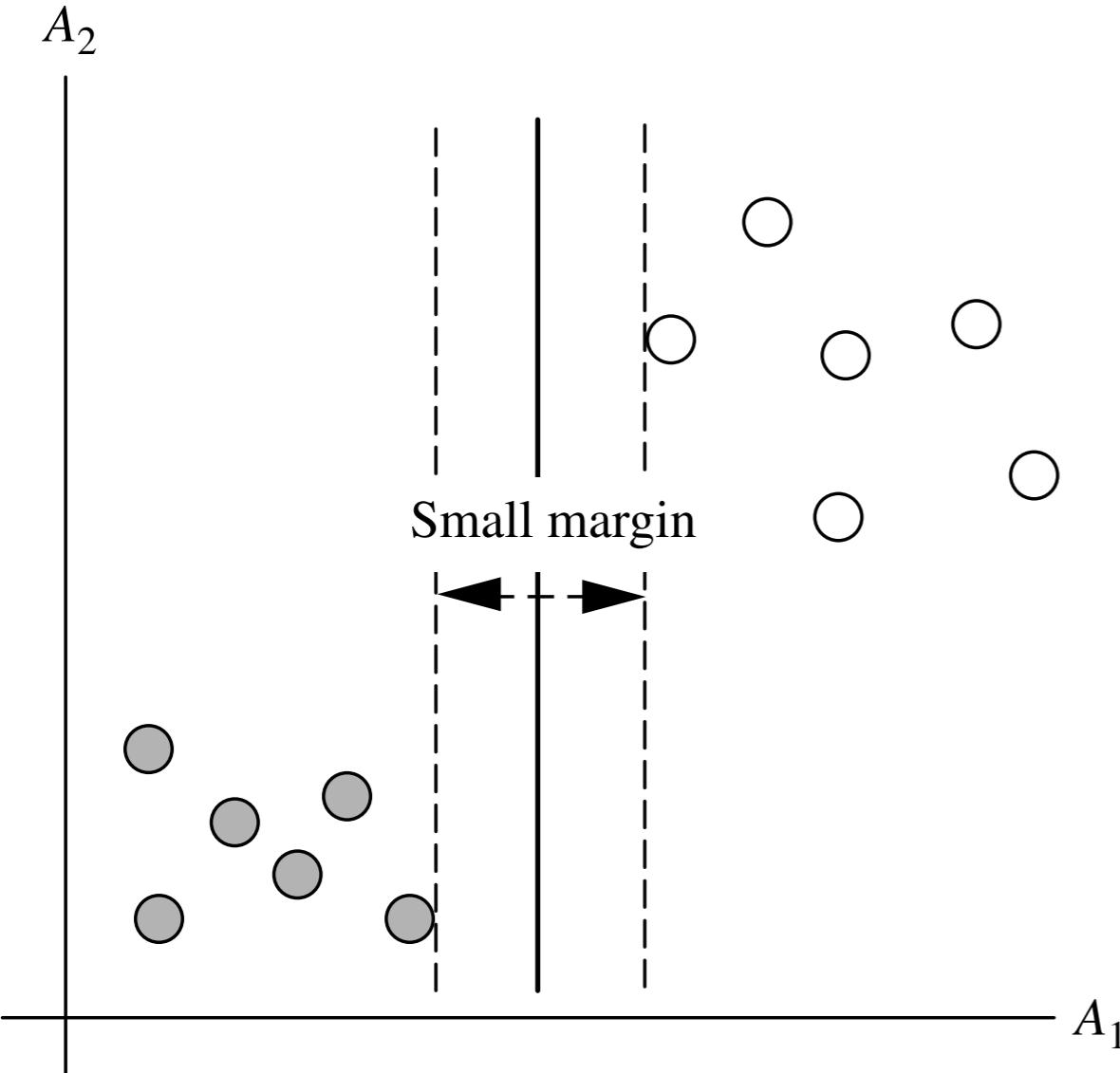
Difficult to understand the learned function (weights)

Bayesian networks can be used easily for pattern discovery

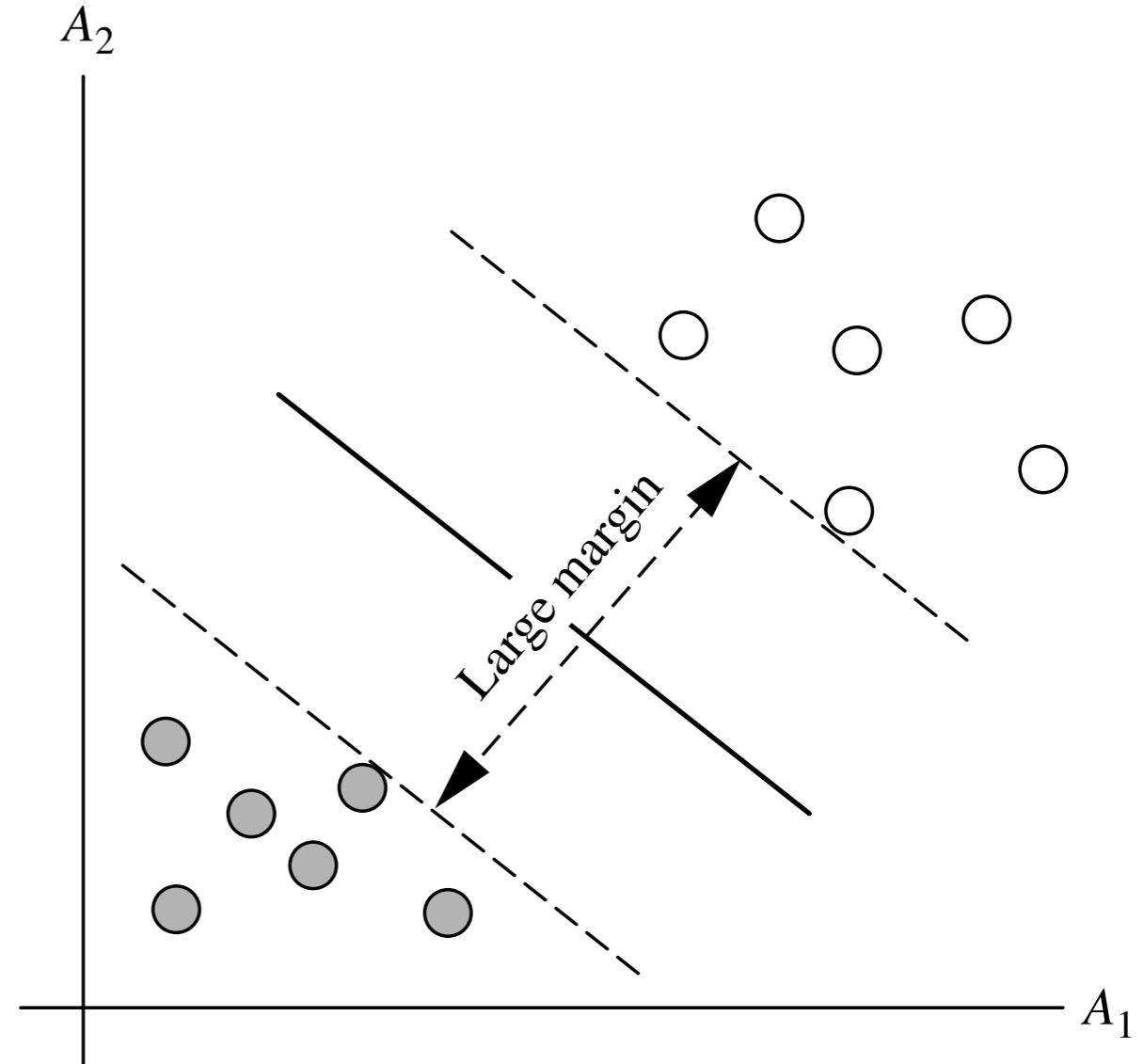
Not easy to incorporate domain knowledge

Easy in the form of priors on the data or distributions

maximum margin hyperplane



- Class 1, $y=+1$ (*buys_computer=yes*)
- Class 2, $y=-1$ (*buys_computer=no*)



- Class 1, $y=+1$ (*buys_computer=yes*)
- Class 2, $y=-1$ (*buys_computer=no*)

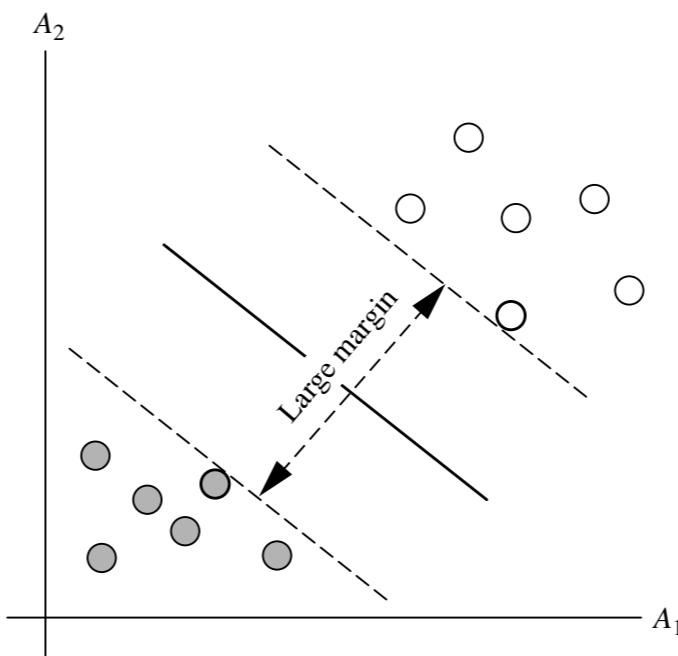
$$W \cdot X + b = 0 \quad \text{equation of a separating hyperplane}$$

It uses a nonlinear mapping to transform the original training data into a higher dimension

With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")

support vector machines

With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane

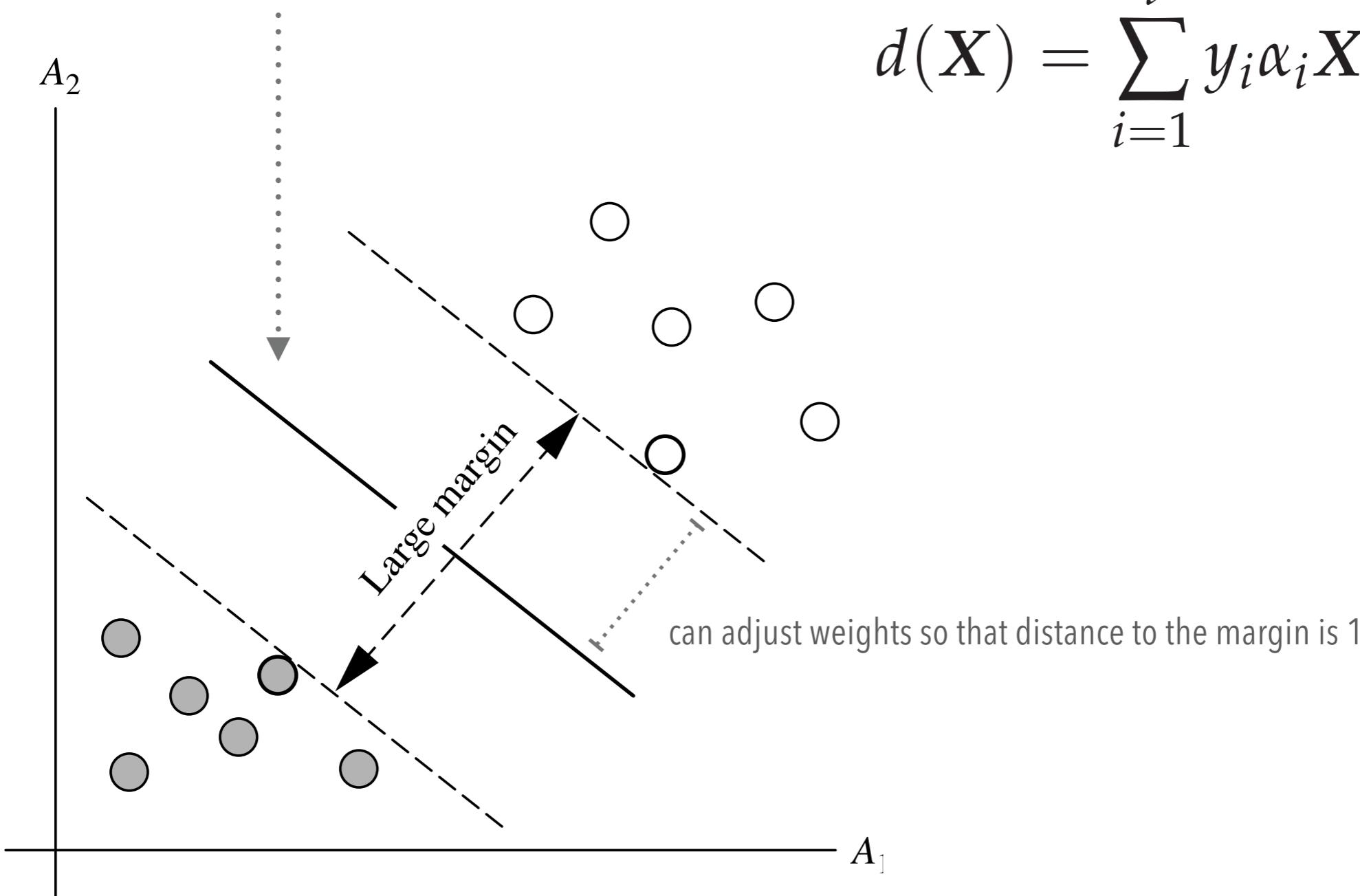


SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)

$$w_1x_1 + w_2x_2 + b_0 = 0$$

decision

$$d(X) = \sum_{i=1}^l y_i \alpha_i X_i X^T + b_0$$

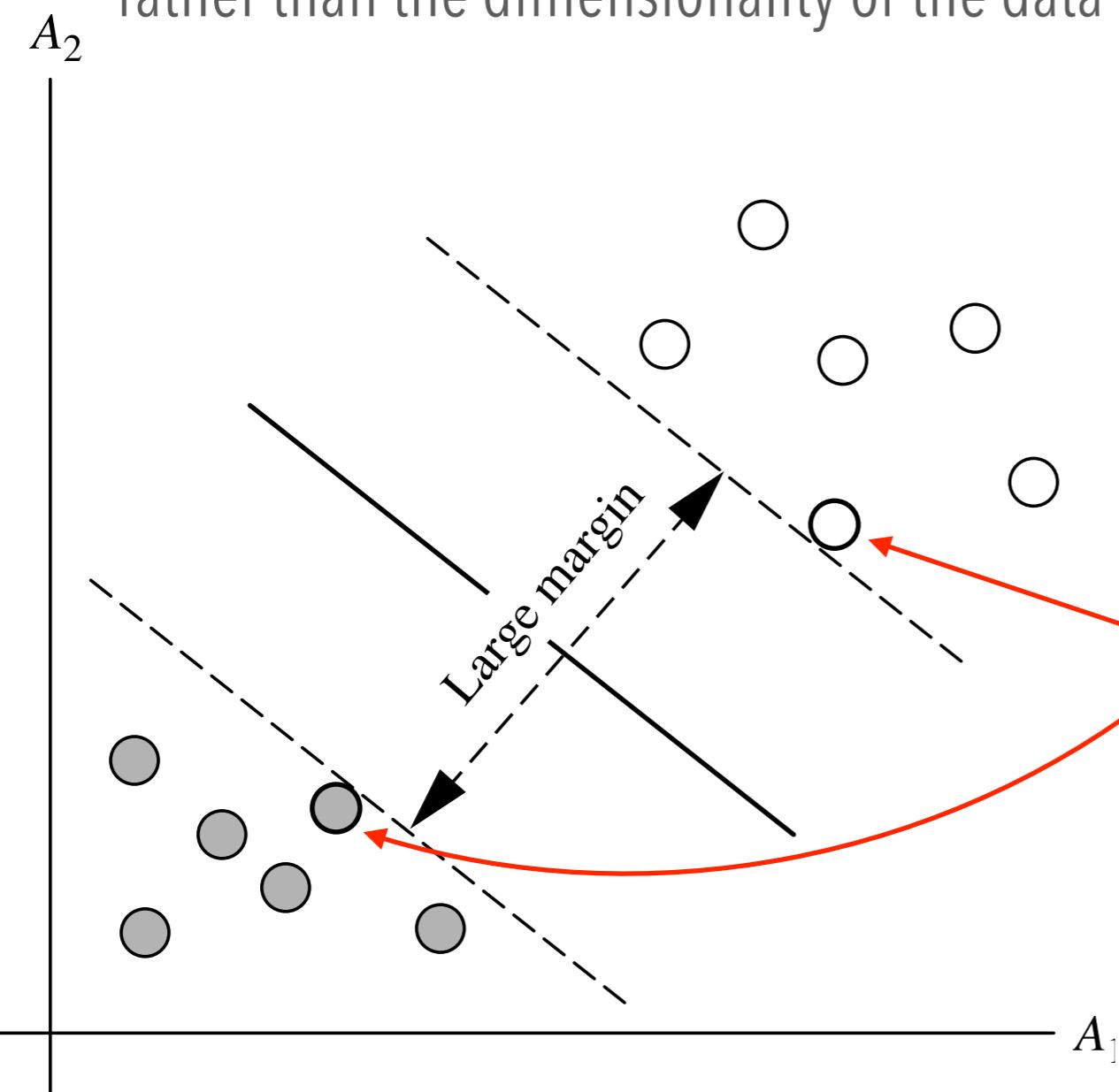


$$\begin{aligned}
 H_1 : w_1 x_{i,1} + w_2 x_{i,2} + b_0 &\geq 1, \\
 H_2 : w_1 x_1 + w_2 x_2 + b_0 &\leq -1,
 \end{aligned}
 \quad \text{general case } y_i = \begin{cases} +1 & \forall i \\ -1 & \end{cases}$$

why are SVM's so effective?

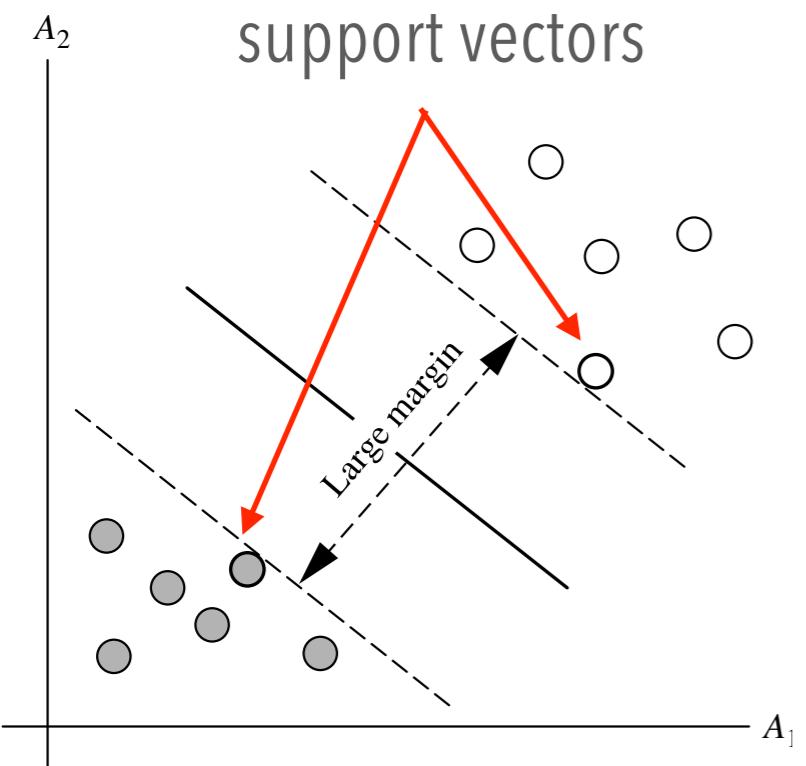


The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data



The support vectors are the essential or critical training examples—they lie closest to the decision boundary ; i.e. the maximum margin hyperplane (MMH)

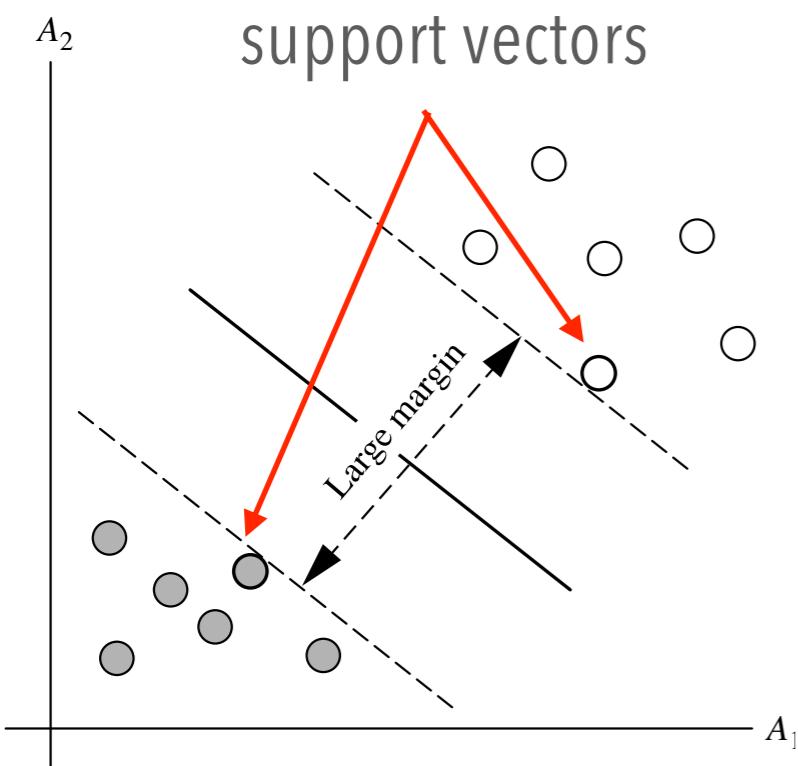
what is so special about the support vectors?



The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality



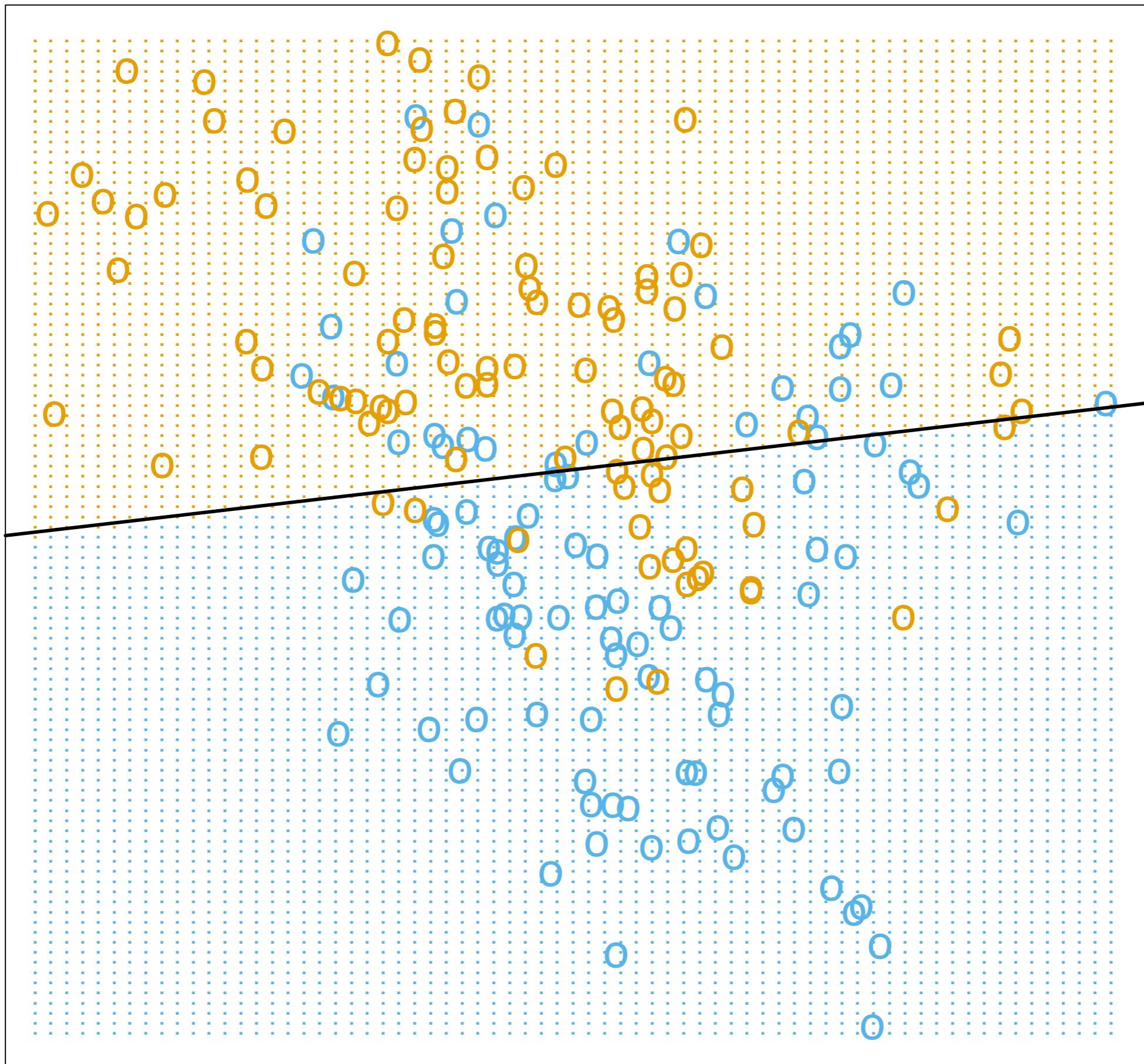
what about generalization?



**but, wait! all the cases
we've discussed are
linearly separable!**

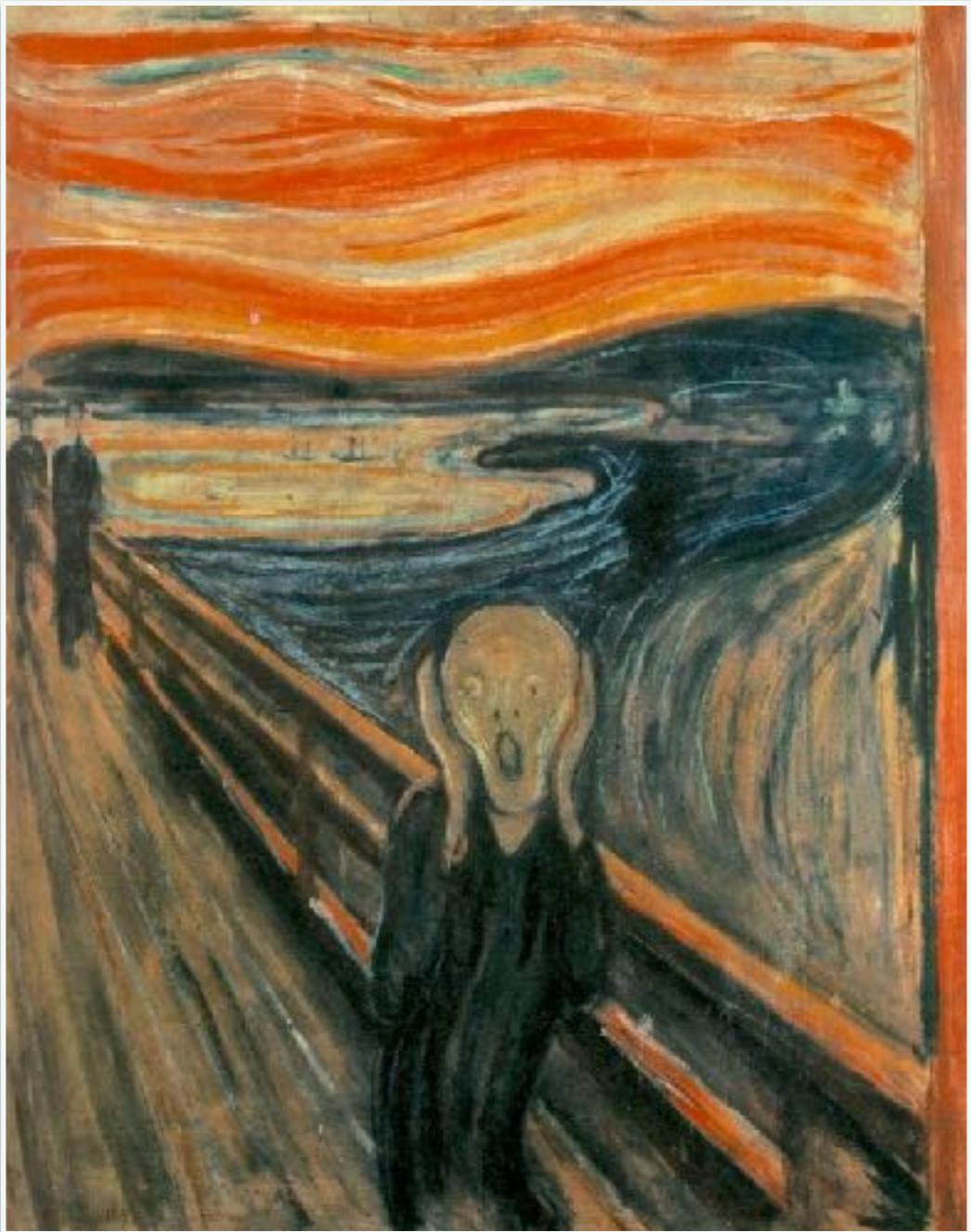
so why are SVM's a big deal?





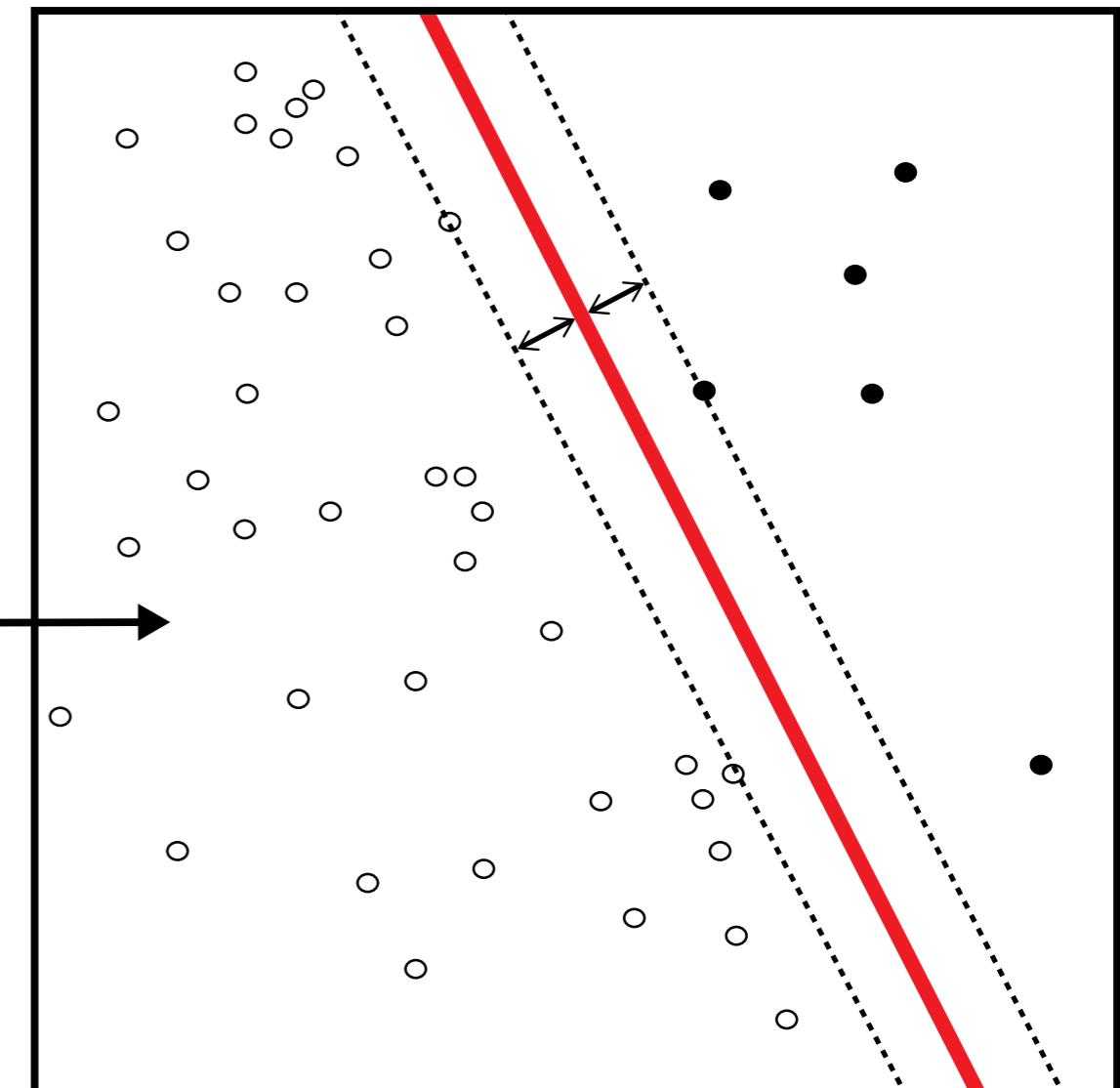
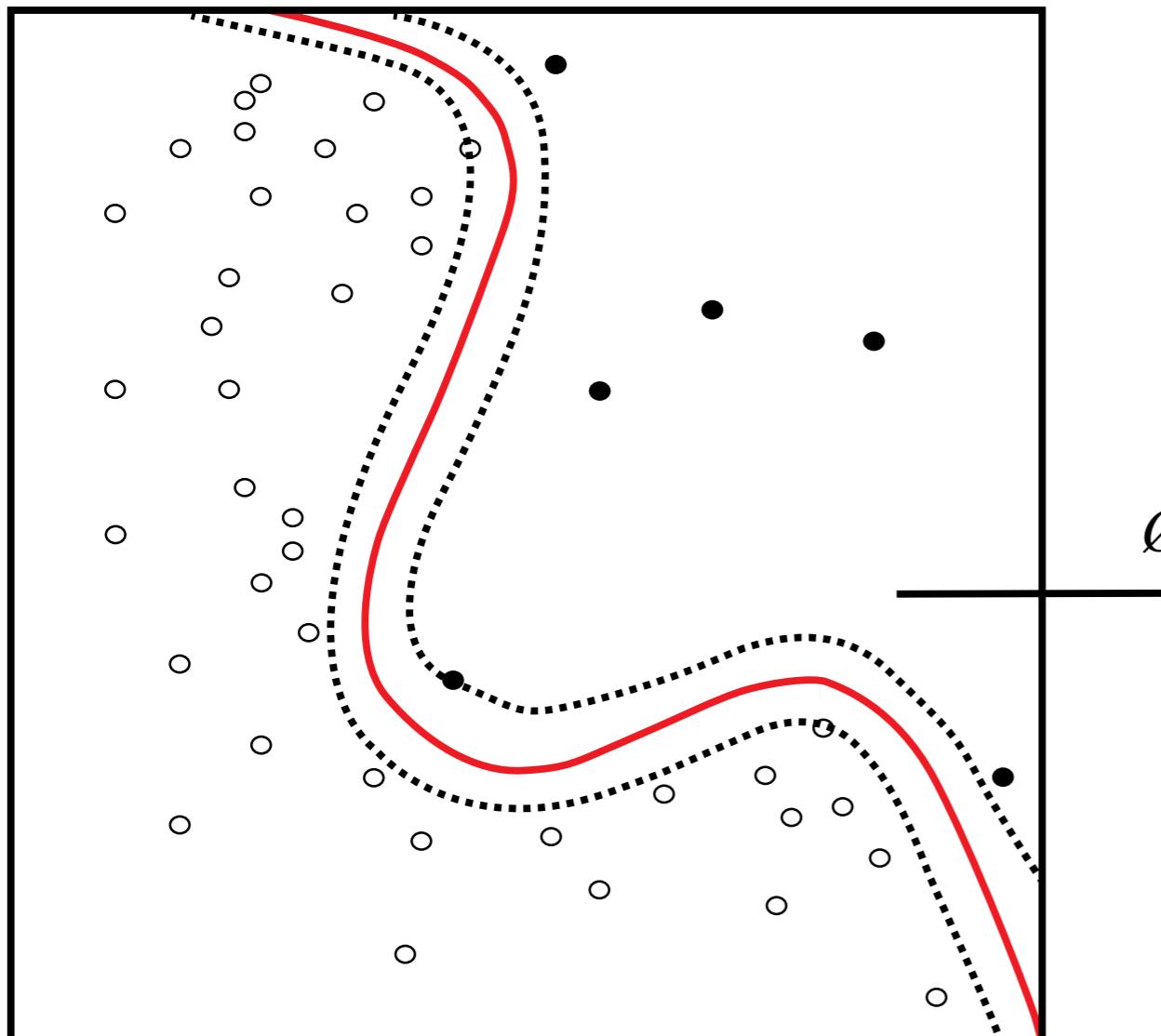
as we've discussed,
for complex
boundaries, we
will use a **non** linear mapping to
a high dimensional
space

$$Z_i = \phi(X_i)$$



Weren't we
done with
kernel functions?

I stopped and leaned against the balustrade,
almost dead with fatigue. Over the blue-black
fjord hung clouds, red as blood—as tongues of
flame. My friends passed on, and alone,
trembling with anguish, I listened to the great,
infinite cry of nature. **E. Munch**



$$Z_i = \phi(X_i)$$

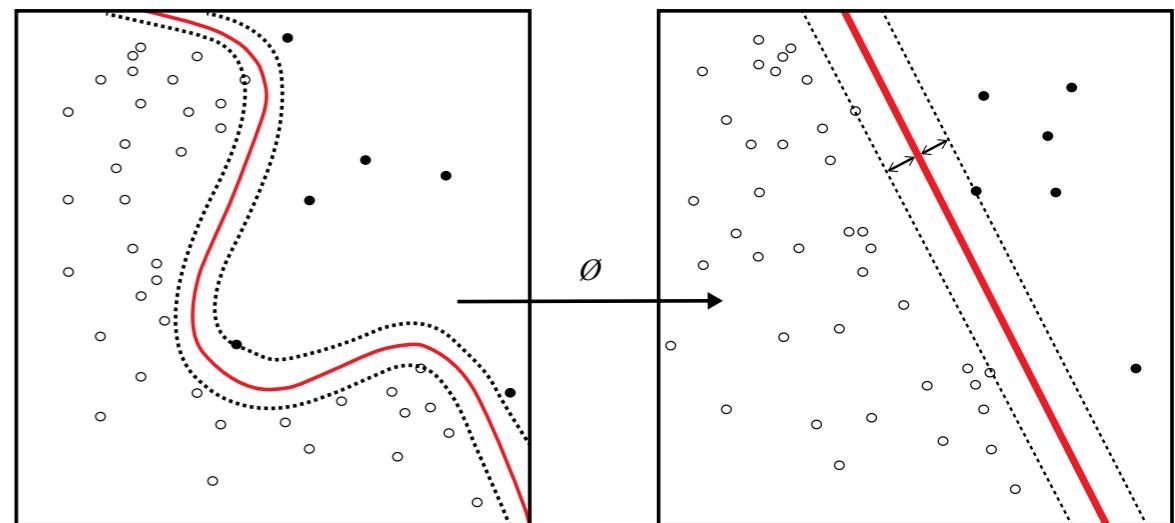
use linear discriminants
in the mapped space!



decision

$$d(\mathbf{X}) = \sum_{i=1}^l y_i \alpha_i \mathbf{X}_i \mathbf{X}^T + b_0$$

this only involves dot products



$$d(\mathbf{X}) = \sum_{i=1}^l y_i \alpha_i \phi(\mathbf{X}_i) \cdot \phi(\mathbf{X}^T) + b_0$$

still dot products, but
this is a little more tricky

enter the kernel trick

what if:

$$K(X_i, X) = \phi(X_i) \cdot \phi(X)$$

K is a function

$$K(X_i, X) = \phi(X_i) \cdot \phi(X)$$

there exists some function K , which is equivalent to the dot product in some transformed space

now not only we
don't have to find
 ϕ , but we don't
have to compute
the dot product!

$$K(X_i, X) = \phi(X_i) \cdot \phi(X)$$

**this sounds too
good to be true!**

is there a catch?

dimensions to which we map



if $d \geq N$, we are
guaranteed 100%
accuracy

how do we find
these kernels?



$$K(\mathbf{X}_i, \mathbf{X}) = \phi(\mathbf{X}_i) \cdot \phi(\mathbf{X})$$

there exists some function K , which is equivalent to the dot product in some transformed space

$$K(\mathbf{X}_i, \mathbf{X}) = (\mathbf{X}_i \cdot \mathbf{X} + 1)^h$$

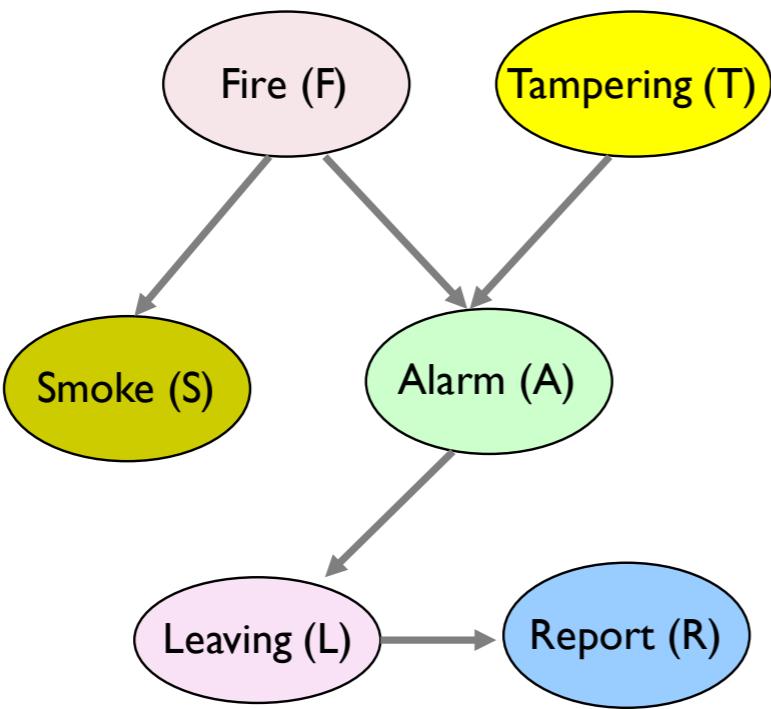
$$K(\mathbf{X}_i, \mathbf{X}) = \exp\left(\frac{-||\mathbf{X}_i - \mathbf{X}||^2}{2\sigma^2}\right)$$

Gaussian

polynomial

**what about
multi-class
classification?**

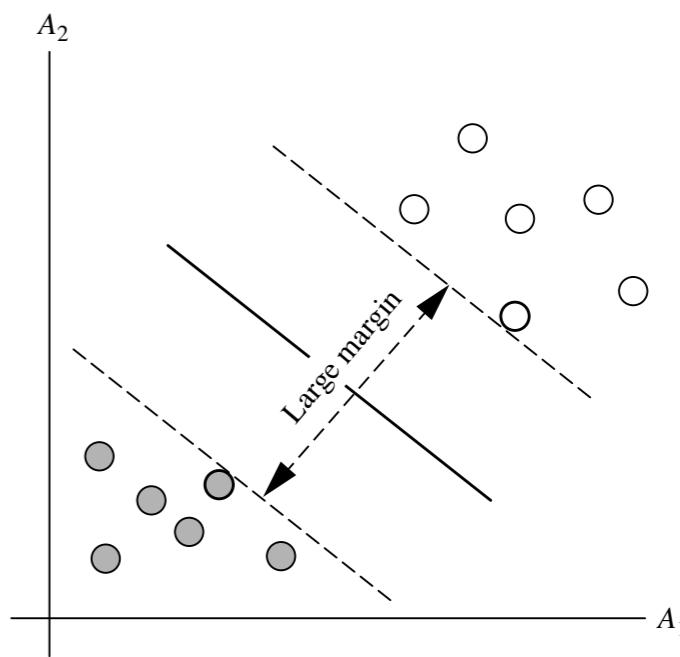




Bayesian belief networks
allow class conditional
independencies between
subsets of variables

DAG's
Markov

SUMMARY

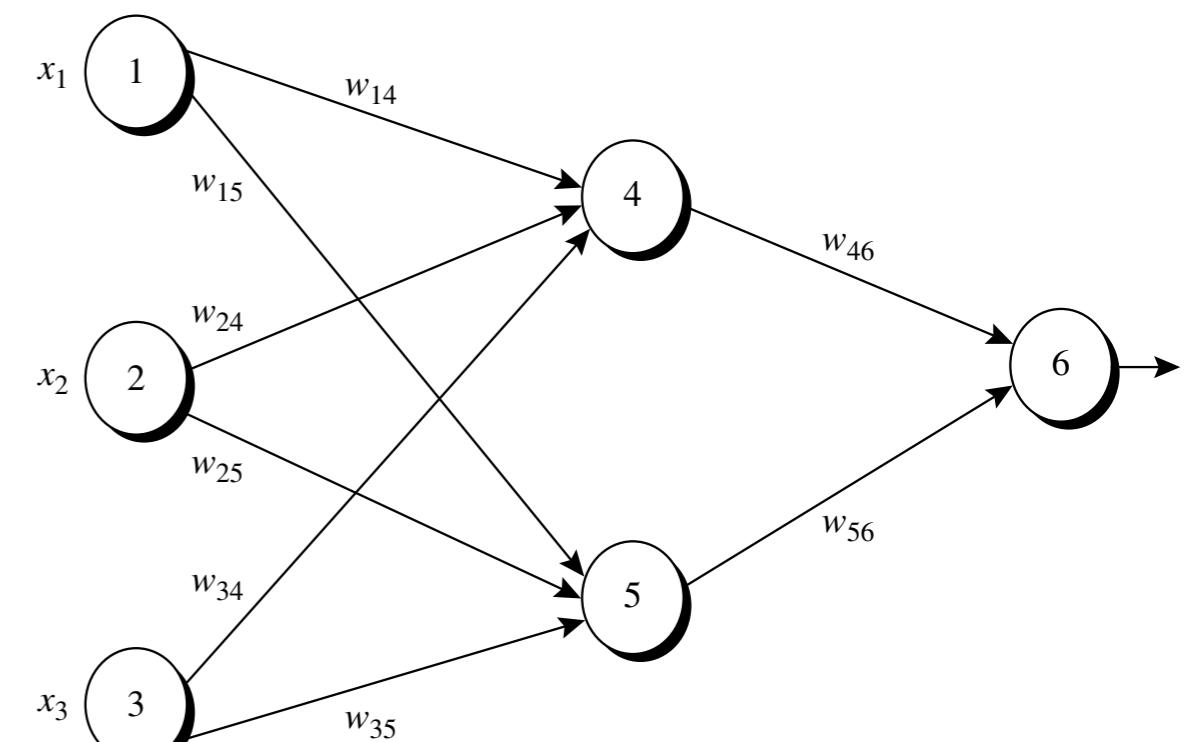


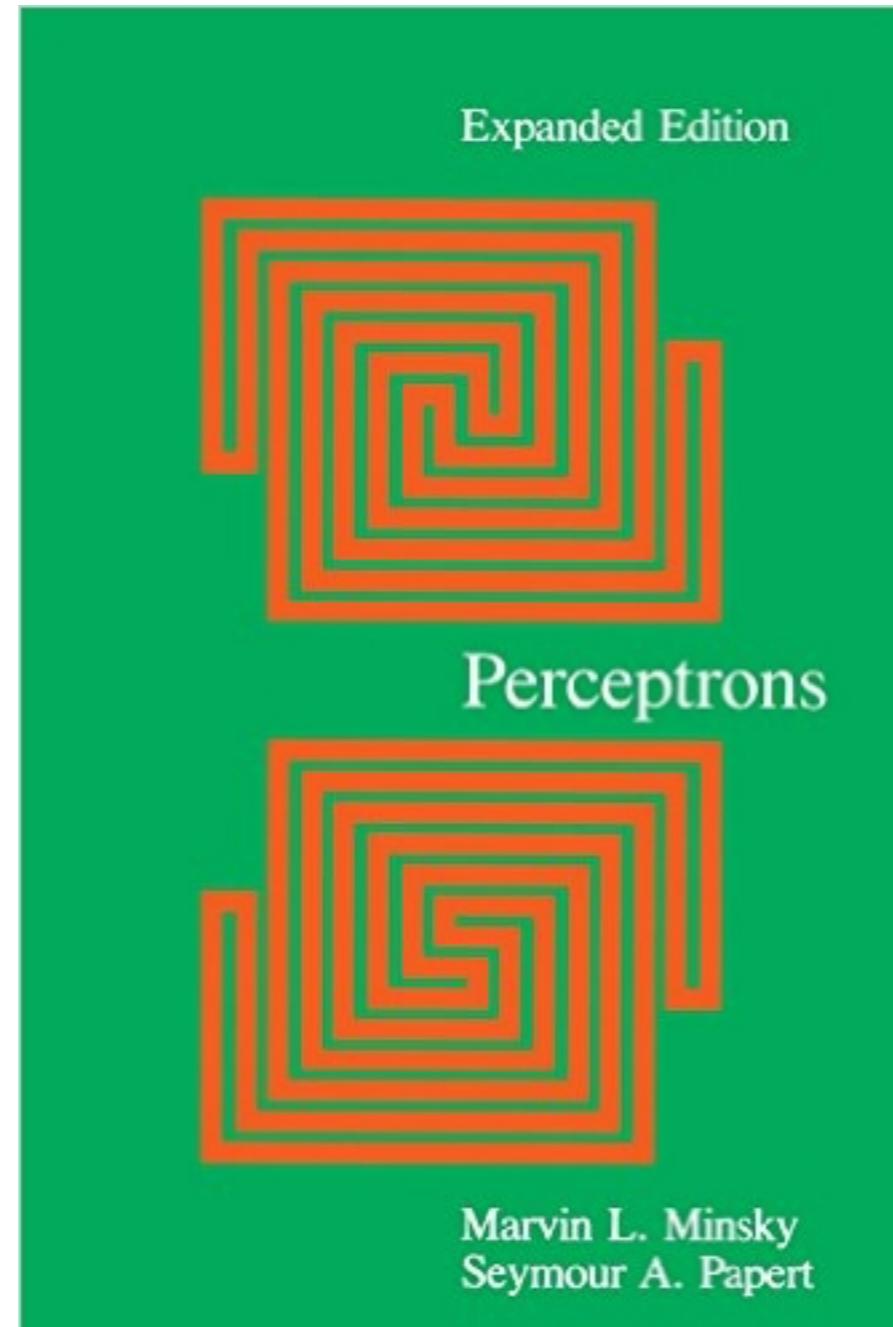
linear discriminants
max margin
kernel trick

NEURAL NETWORKS

Bayesian Networks Support Vector Machines Etc

Summary





cannot represent XOR gates

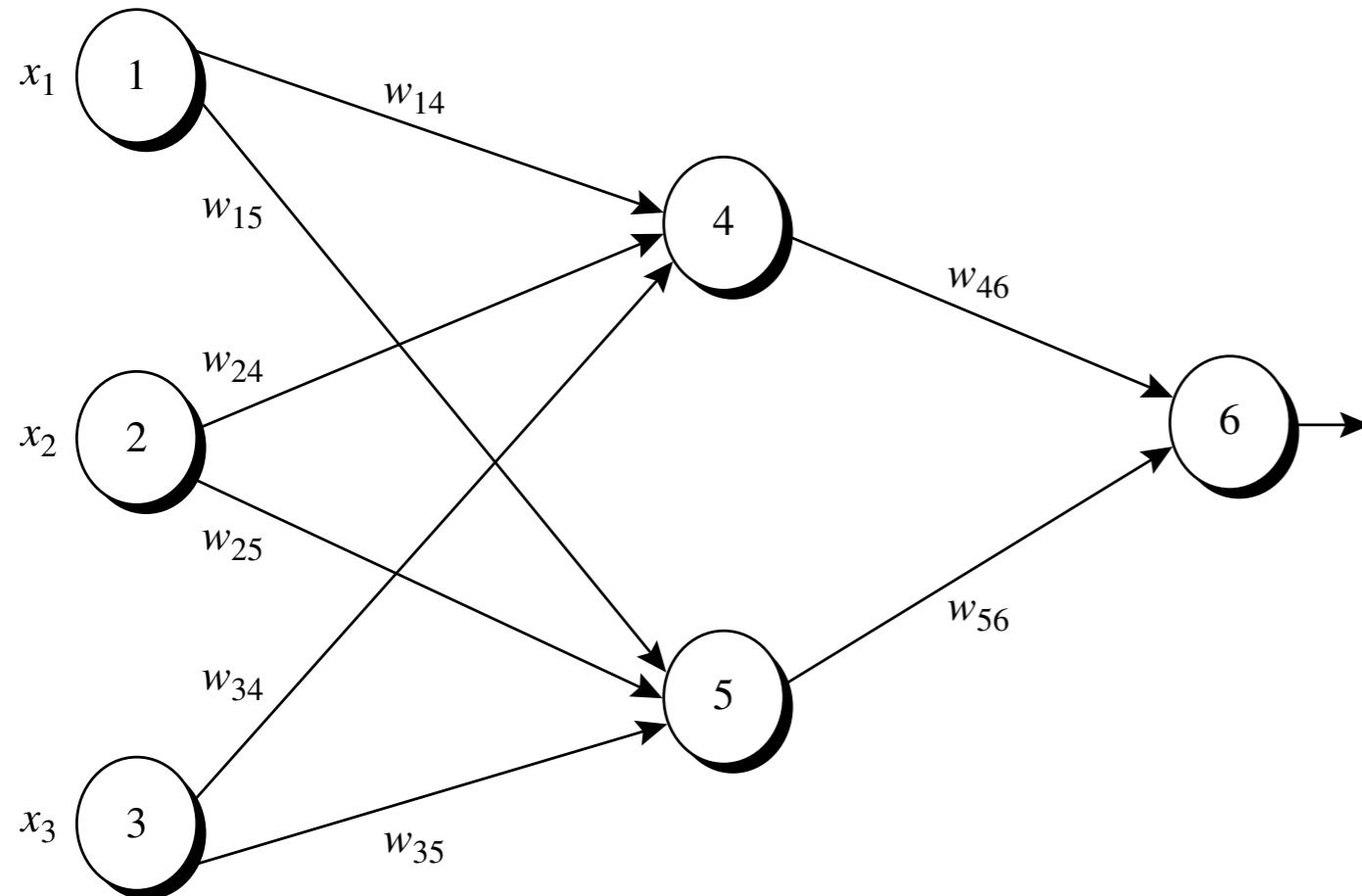
$$f(\mathbf{x}) = b + \sum_i \alpha_i x_i$$



All winter lasted till mid 80's

BACKPROPAGATION

.....



Backpropagation: A neural network learning algorithm

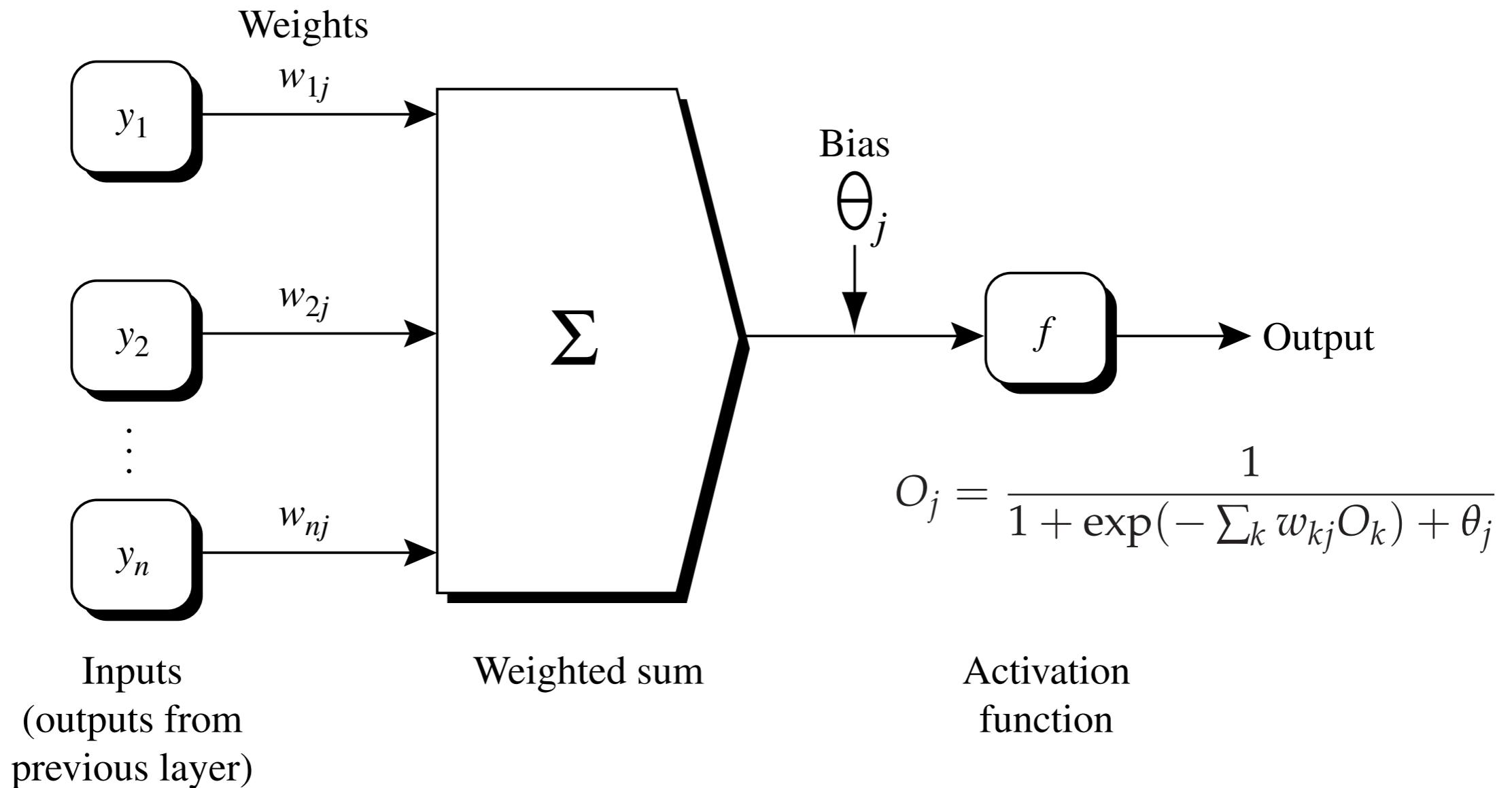
Started by psychologists and neurobiologists to develop and test computational analogues of neurons

A neural network: A set of connected input/output units where each connection has a weight associated with it

During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples

Also referred to as connectionist learning due to the connections between units

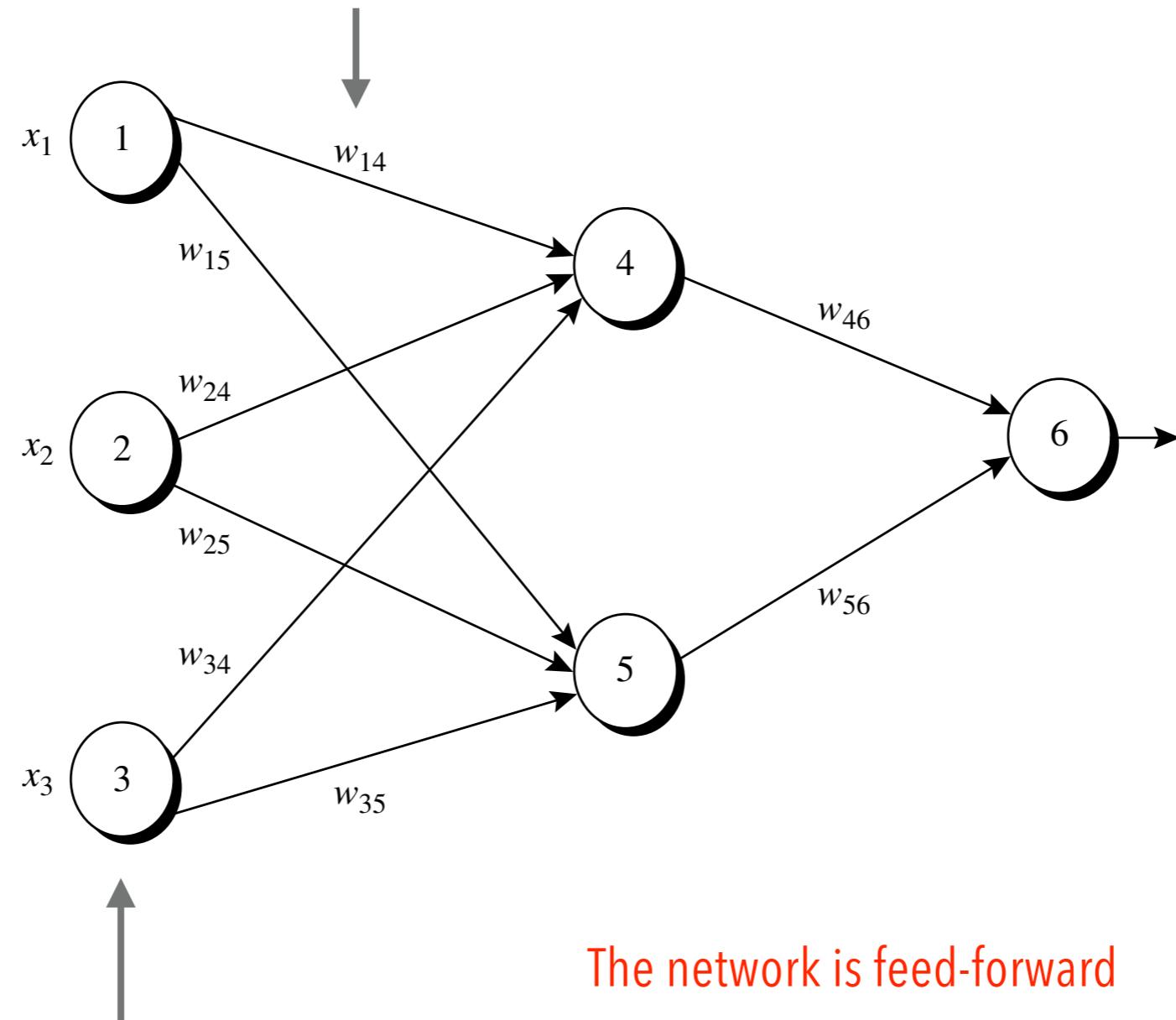
An n-dimensional input vector \mathbf{x} is mapped into variable \mathbf{y} by means of the scalar product and a nonlinear function mapping



The inputs to unit are outputs from the previous layer. They are multiplied by their corresponding weights to form a weighted sum, which is added to the bias associated with unit. Then a nonlinear activation function is applied to it.

They are then weighted
and fed simultaneously
to a hidden layer

The inputs to the network
correspond to the attributes
measured for each training
tuple



Inputs are fed
simultaneously into the units
making up the input layer

The network is feed-forward

The weighted outputs
of the last hidden layer
are input to units
making up the output
layer, which emits the
network's prediction

how do we
decide on the
network
topology?

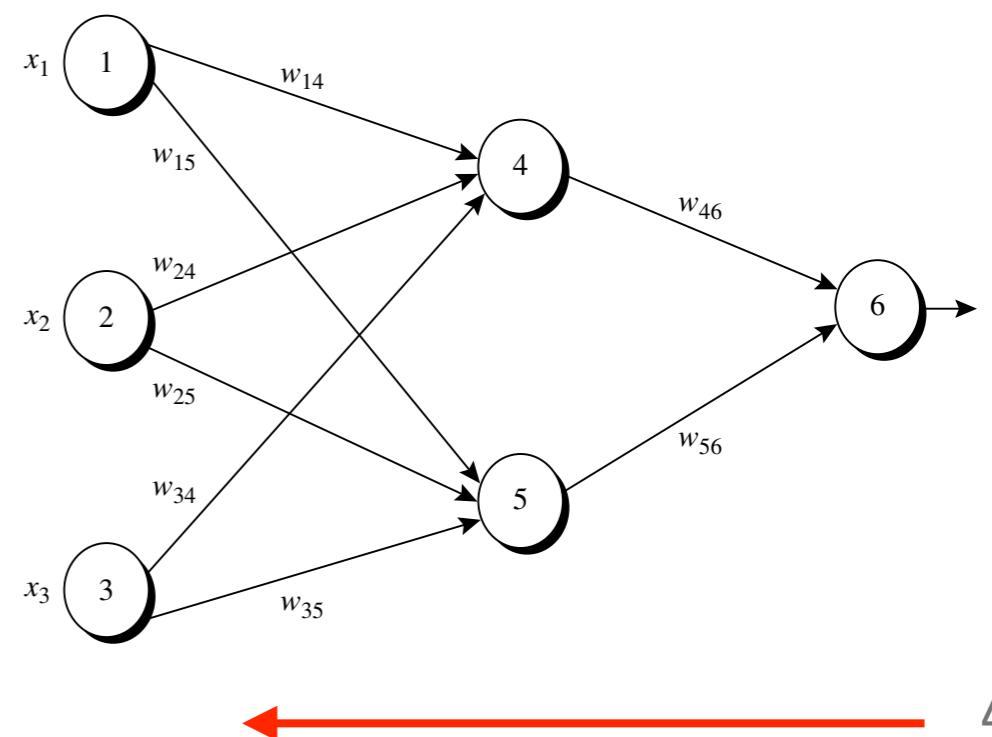


Iteratively process a set of training tuples & compare the network's prediction with the actual known target value

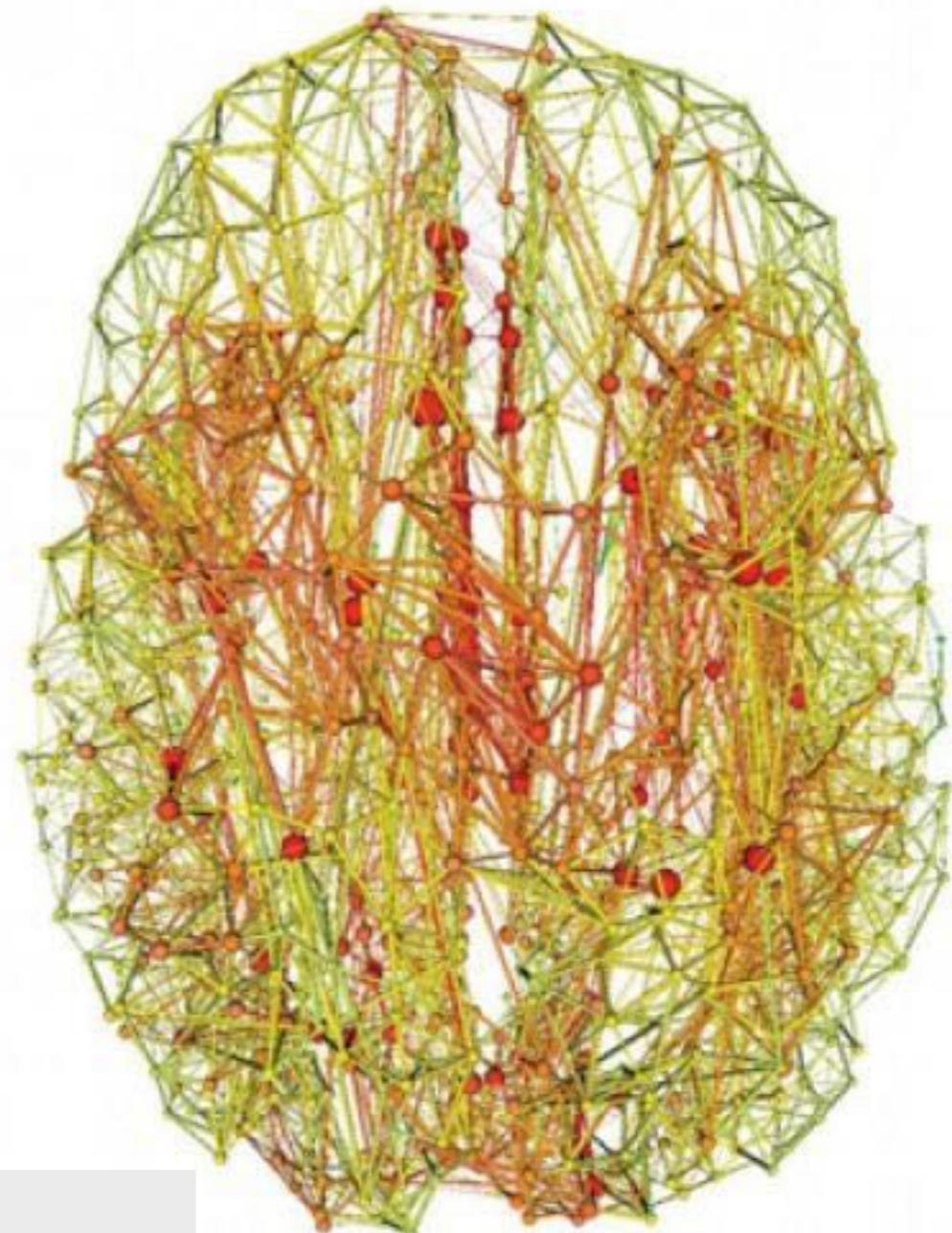
For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value

backpropagation

Modifications are made in the "backwards" direction:
from the output layer,
through each hidden layer
down to the first hidden layer,

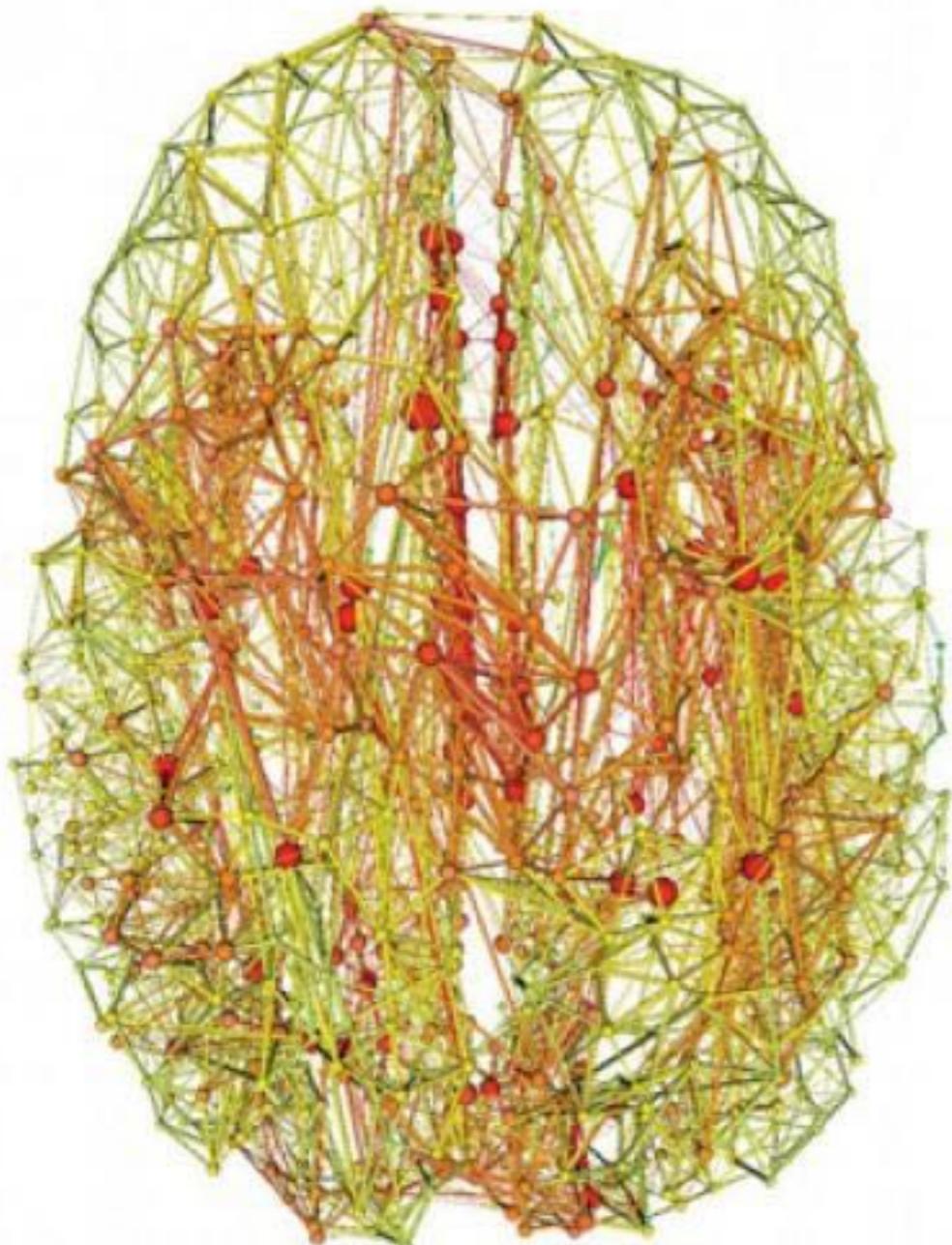


deep networks



Bengio, Y. (2009). Learning deep architectures for AI.
Foundations and trends® in Machine Learning, 2(1), 1-127.

SOME CONSIDERATIONS

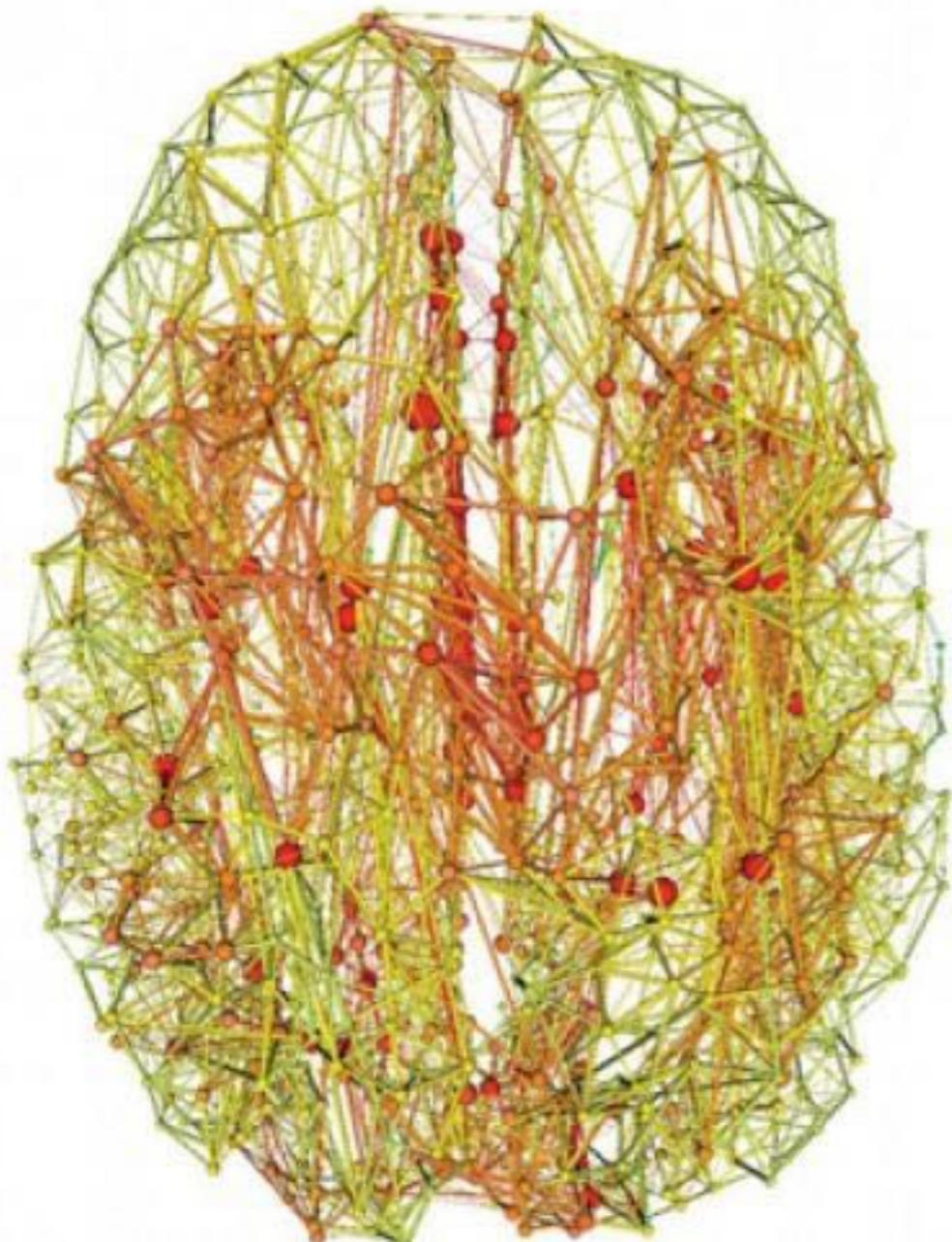


Ability to learn **complex, highly-varying** functions, i.e., with a number of variations much greater than the number of training examples.

Ability to learn **with little human input** the low-level, intermediate, and high-level abstractions that would be useful to represent the kind of complex functions needed for AI tasks.

Ability to learn **from a very large set of examples**: computation time for training should scale well with the number of examples, i.e., close to linearly.

SOME CONSIDERATIONS



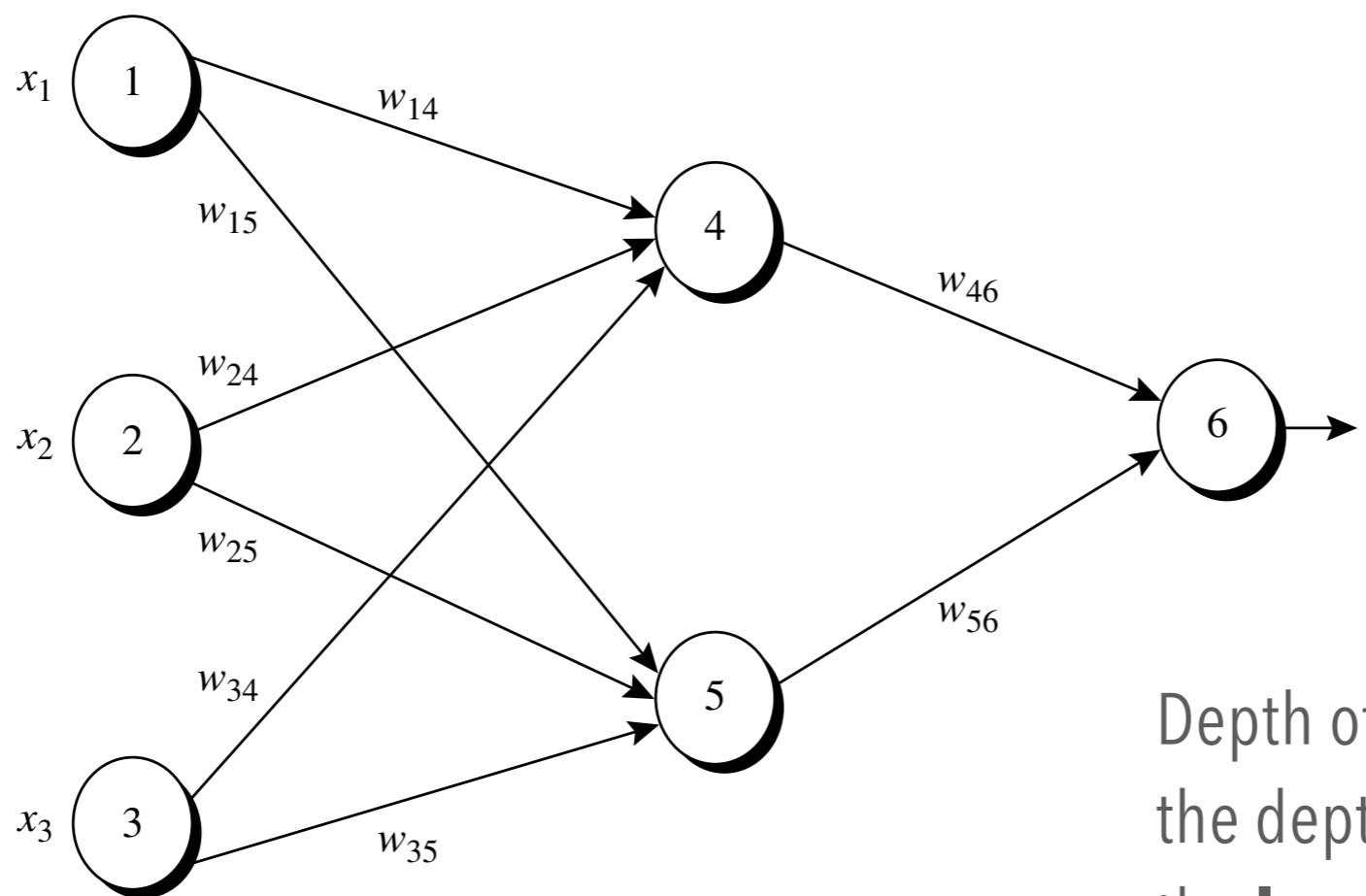
Ability to **learn from mostly unlabeled data**, i.e., to work in the semi-supervised setting, where not all the examples come with complete and correct semantic labels.

Ability to **exploit the synergies present across a large number of tasks**, i.e., multi-task learning. These synergies exist because all the AI tasks provide different views on the same underlying reality.

Strong unsupervised learning (i.e., capturing most of the statistical structure in the observed data), which seems essential in the limit of a large number of tasks and when future tasks are not known ahead of time.

**what is
shallow?**

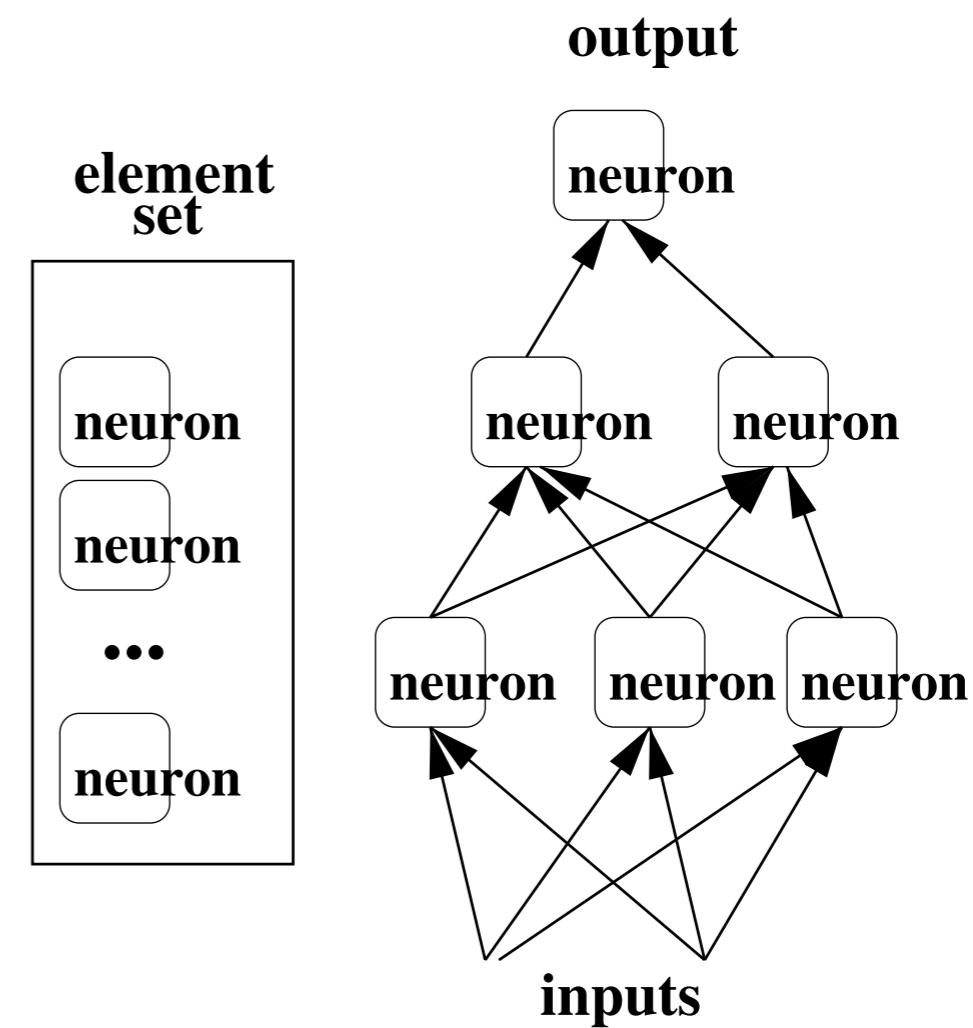
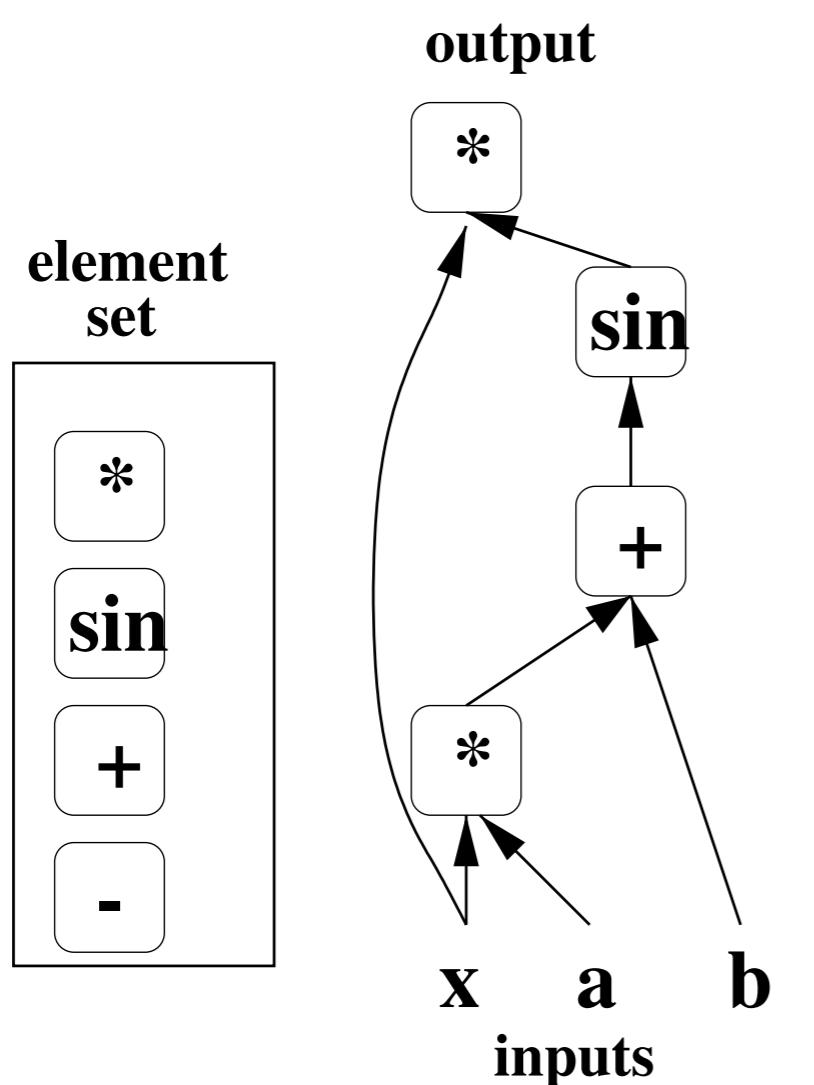


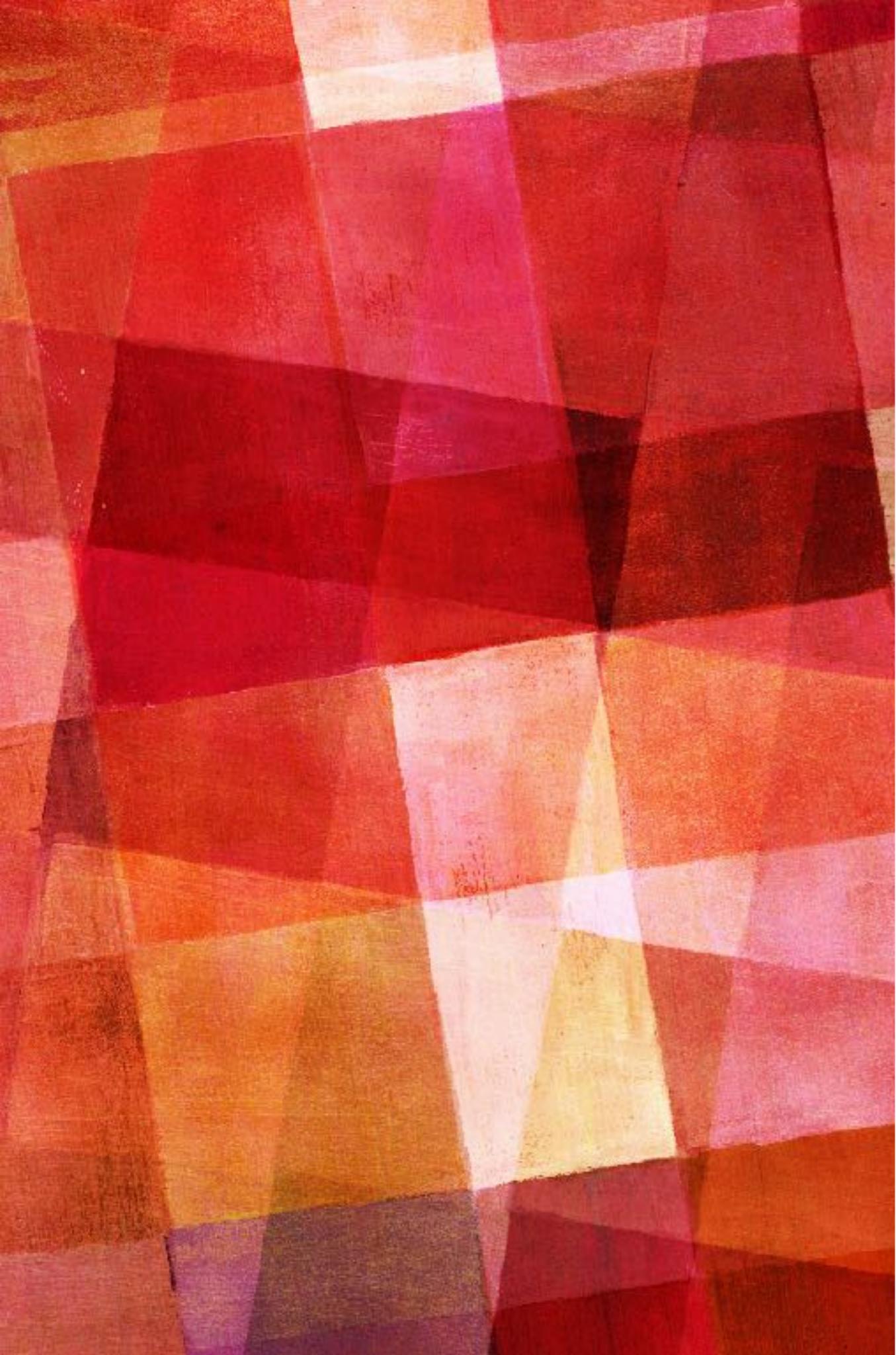


Depth of architecture refers to the depth of that graph, i.e., the **longest path** from an input node to an output node

the expression of a function is **compact**
when it has few computational
elements, i.e., few degrees of freedom
that need to be tuned by learning.

$$f(x) = x \sin(ax + b)$$





DEPTH

If we include affine operations and their possible composition with sigmoids in the set of computational elements, linear regression and logistic regression have depth 1, i.e., have a single level.

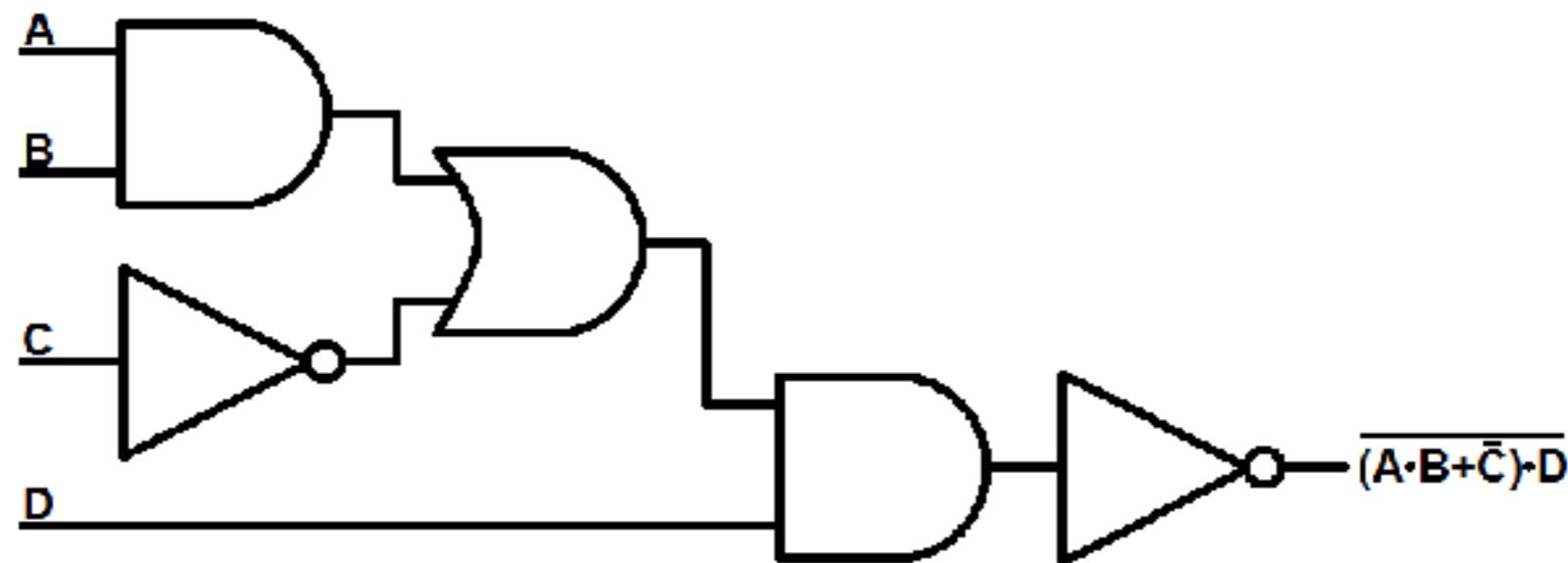
When we put a fixed kernel computation $K(u,v)$ in the set of allowed operations, along with affine operations, kernel machines with a fixed kernel can be considered to have two levels. The first level has one element computing $K(x,x_i)$ for each prototype x_i (a selected representative training example) and matches the input vector x with the prototypes x_i . The second level performs an affine combination.

Boosting usually adds one level to its base learners: that level computes a vote or linear combination of the outputs of the base learners.

Based on current knowledge of brain anatomy, it appears that the cortex can be seen as a deep architecture, with 5–10 levels just for the visual system.

More precisely, functions that can be compactly represented by a depth k architecture might require an exponential number of computational elements to be represented by a depth $k-1$ architecture.

when a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one.

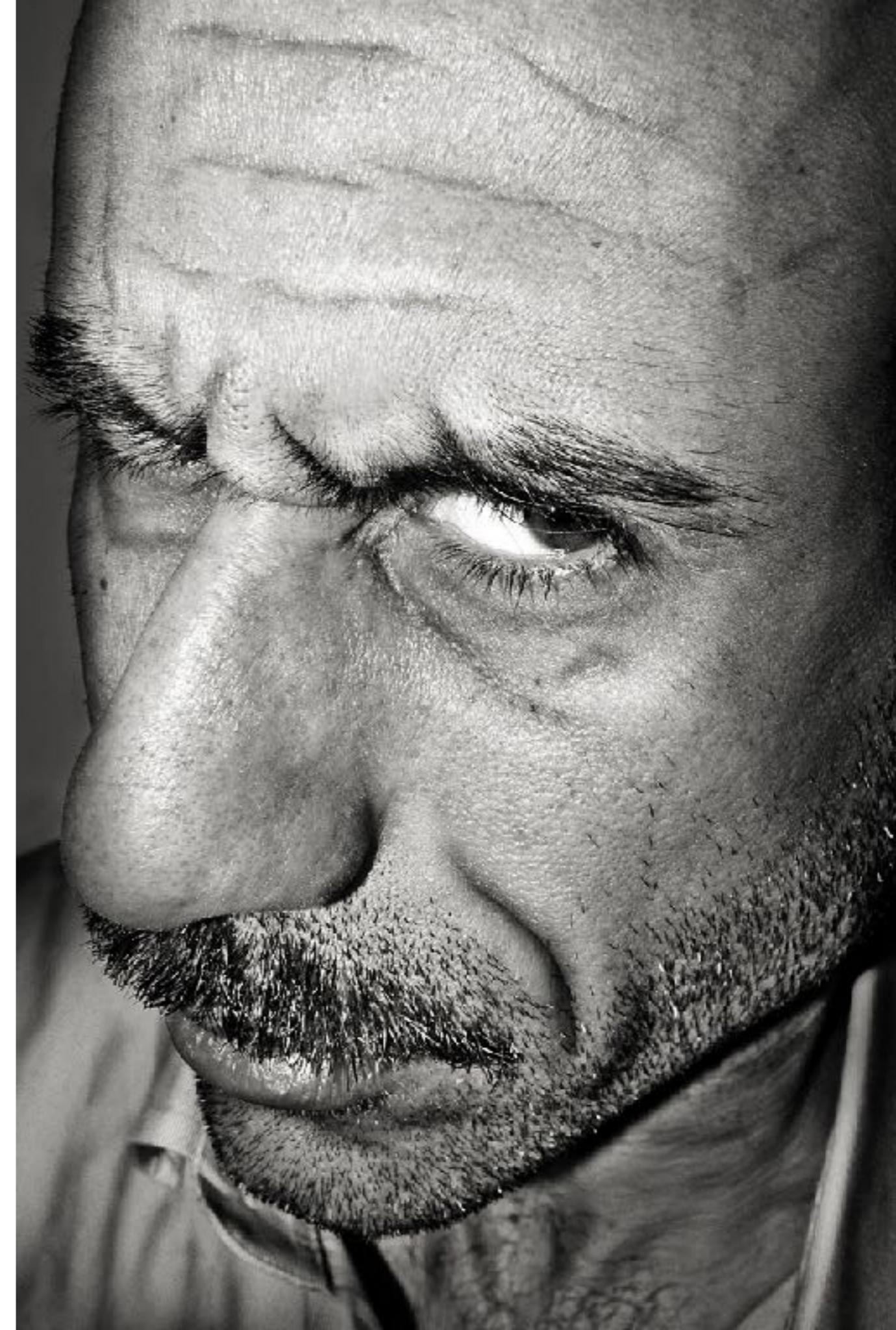


consider a function described by logic gates

there are functions computable with a polynomial-size logic gates circuit of depth **k** that require exponential size when restricted to depth **k-1**

does the result generalize?

there are strong
indications that deep
networks are more
"efficient"



An estimator that is local in input space obtains good generalization for a new input \mathbf{x} by mostly exploiting training examples in the neighborhood of \mathbf{x} .

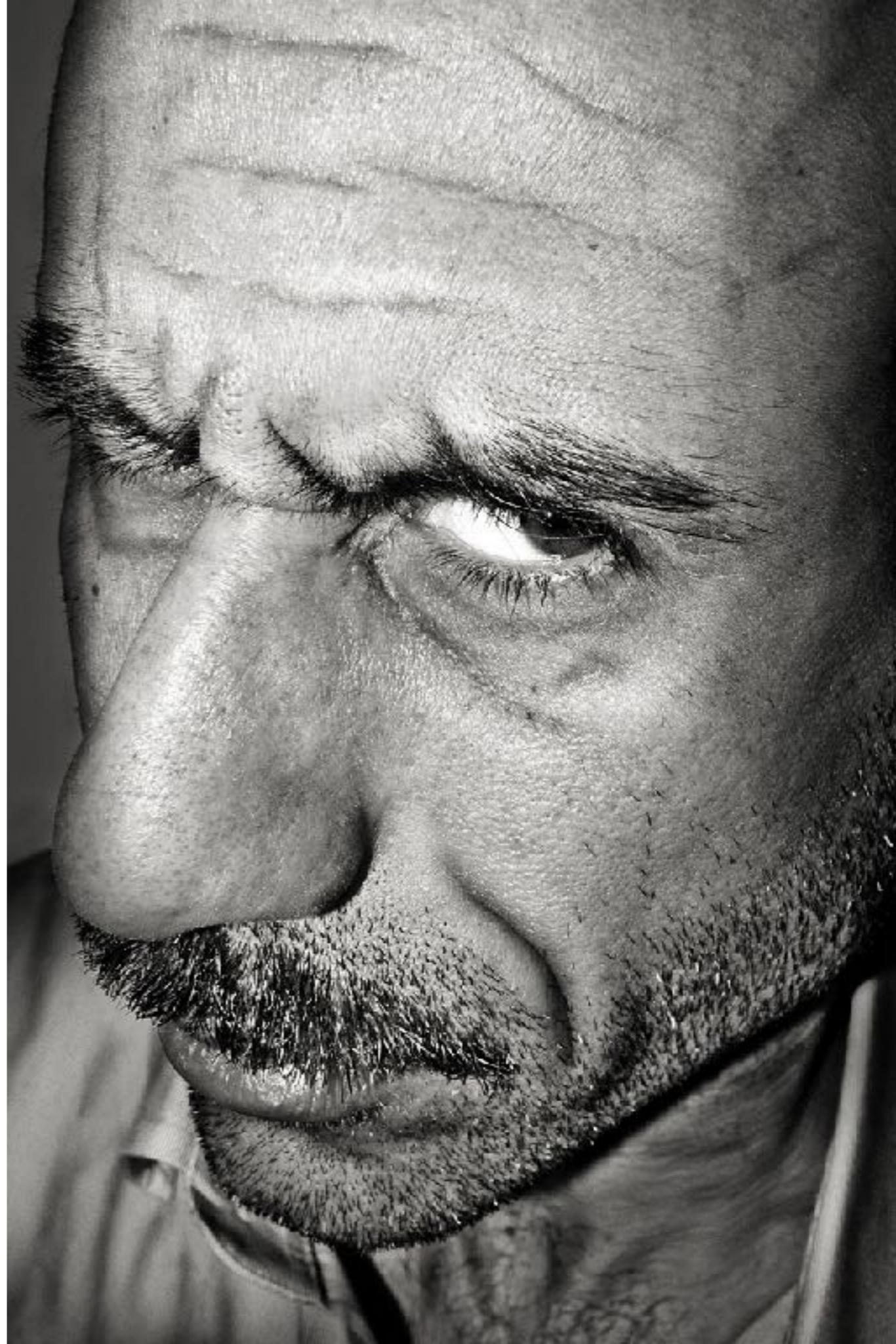
$$f(\mathbf{x}) = b + \sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

smoothness prior

such a prior is often insufficient to generalize when the target function is highly varying in input space.

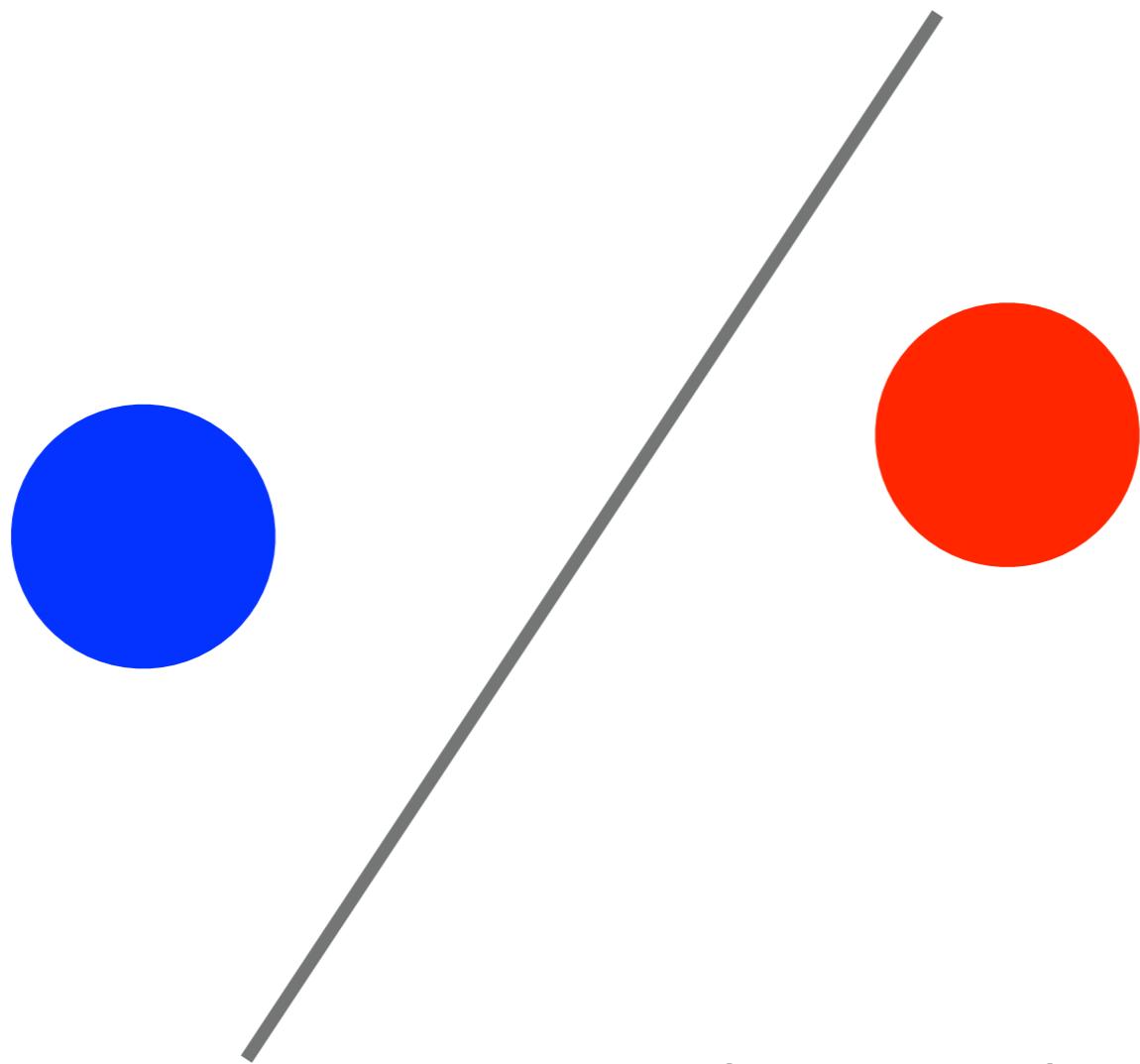
Are the depth 1,
2 and 3
architectures **too**
shallow to
represent
efficiently more
complicated
functions needed
for AI tasks?

there might be no
universally right depth

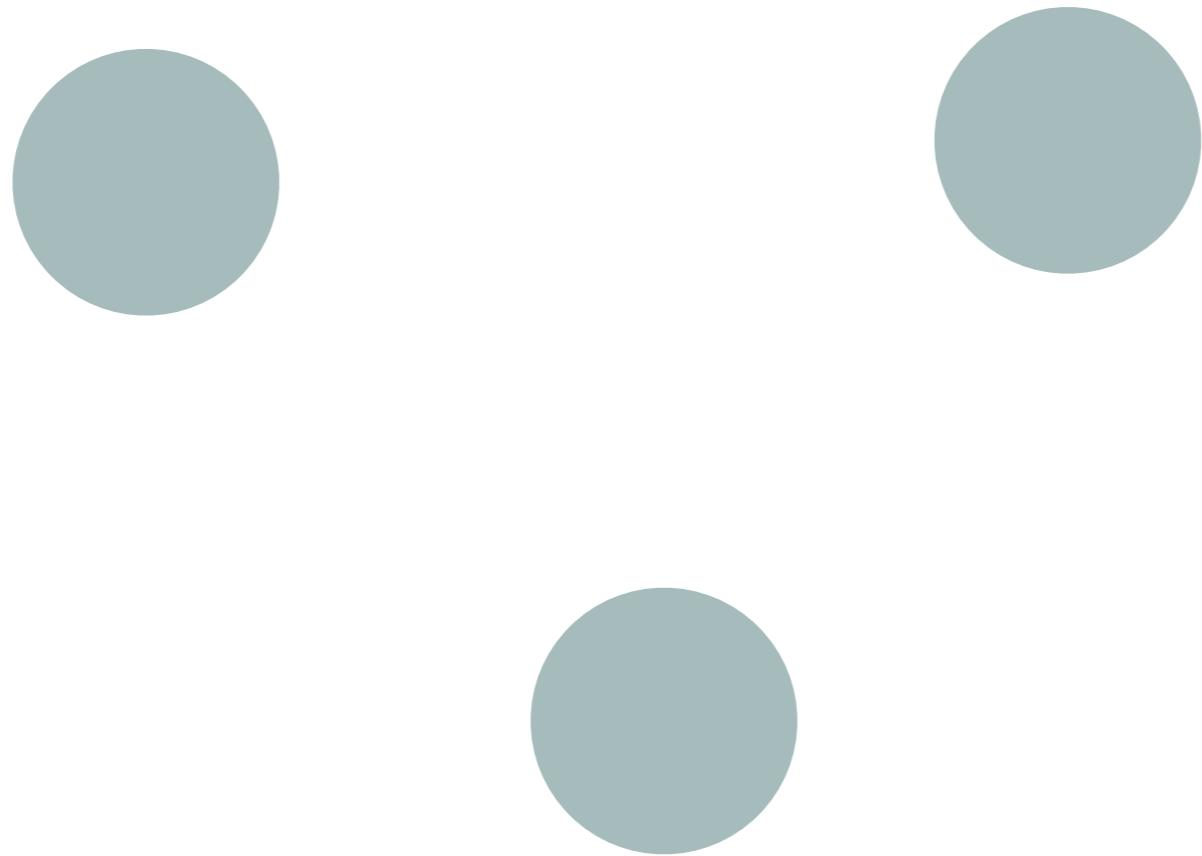


VC dimension

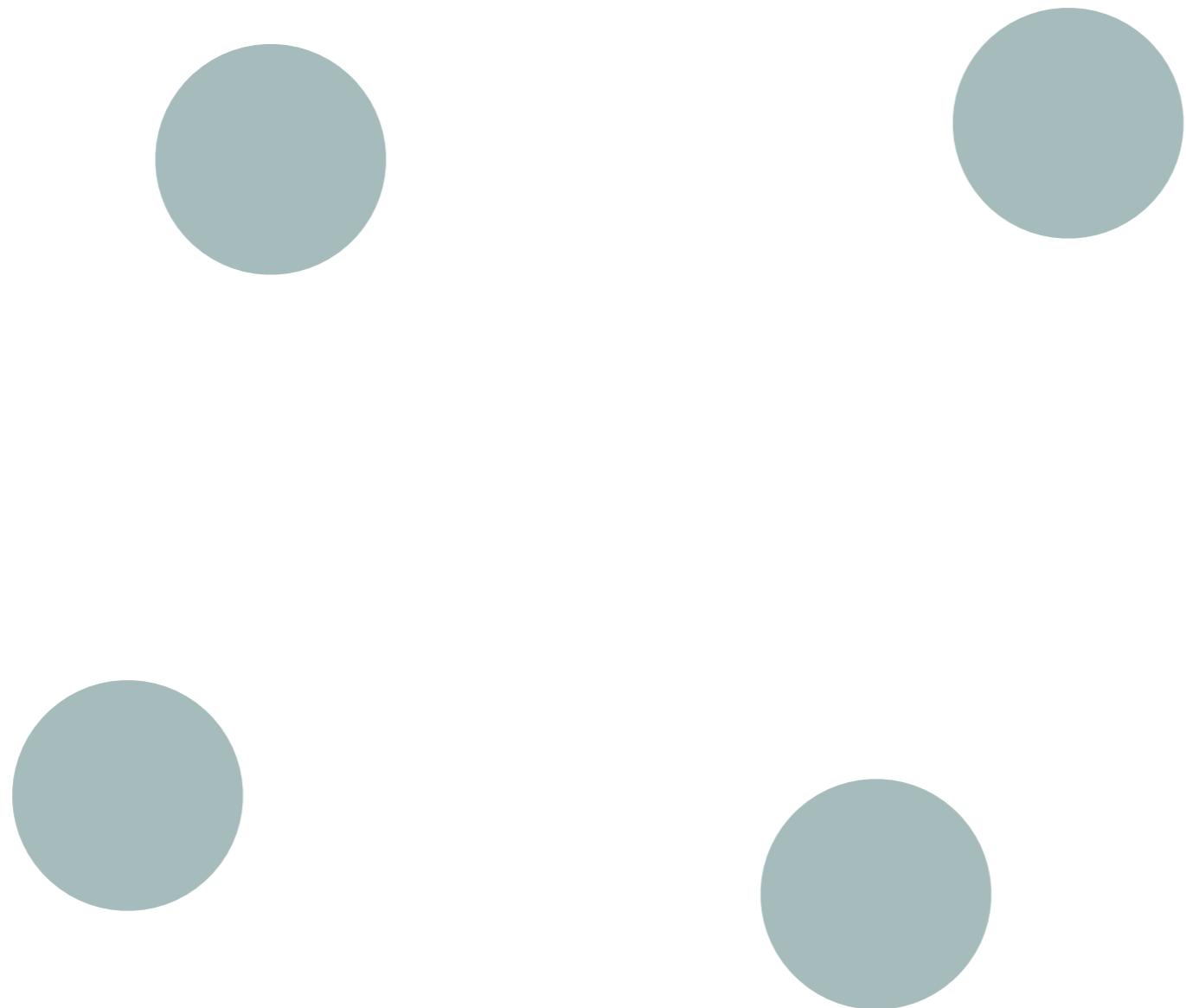
the largest number of points that a model parameterized by Θ makes no error in correctly assigning labels



can these points be shattered by a line?



can these points be shattered by a line?



can these points be shattered by a line?

I7 = 00 | 000...0



17th position, local

$$|7 = |000|$$



distributed, compact

For the same number of possible configurations, a distributed representation can potentially be exponentially more compact than a very local one.

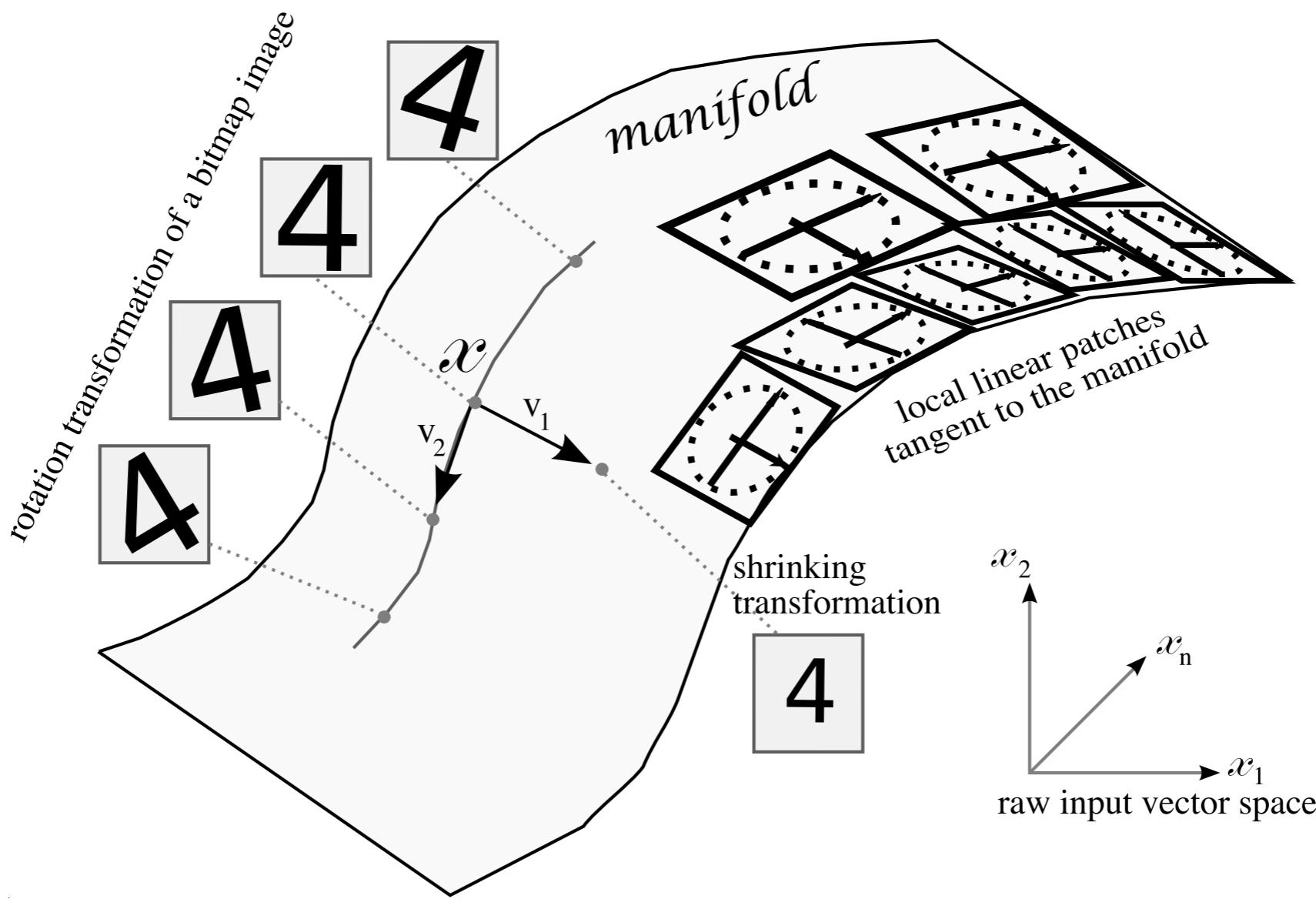


Fig. 3.1 The set of images associated with the same object class forms a manifold or a set of disjoint manifolds, i.e., regions of lower dimension than the original space of images. By rotating or shrinking, e.g., a digit 4, we get other images of the same class, i.e., on the same manifold. Since the manifold is locally smooth, it can in principle be approximated locally by linear patches, each being tangent to the manifold. Unfortunately, if the manifold is highly curved, the patches are required to be small, and exponentially many might be needed with respect to manifold dimension. Graph graciously provided by Pascal Vincent.

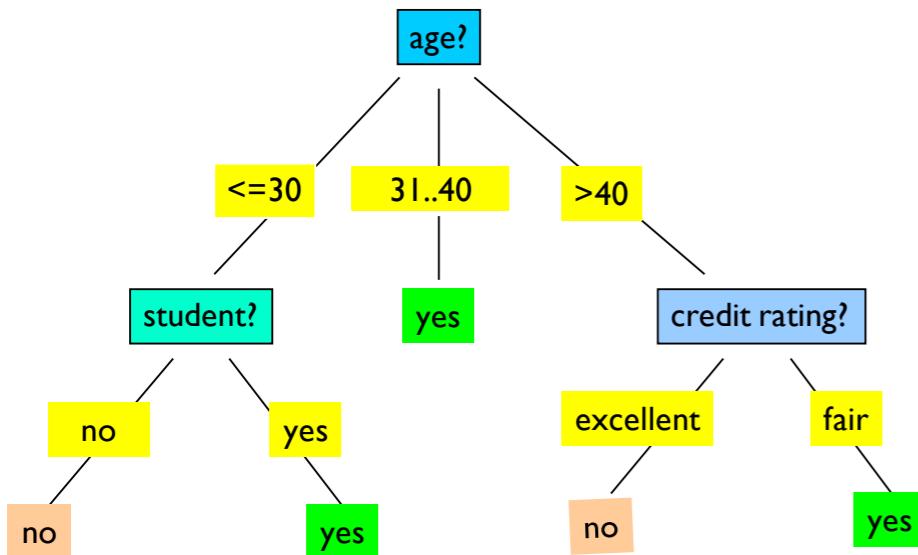
Sparsity: encouraging many units to take the value 0



allows for representations that are in between being fully local (i.e., maximally sparse) and non-sparse (i.e., dense) distributed representations.

Neurons in the cortex are believed to have a distributed and sparse representation, with around 1-4% of the neurons active at any one time.

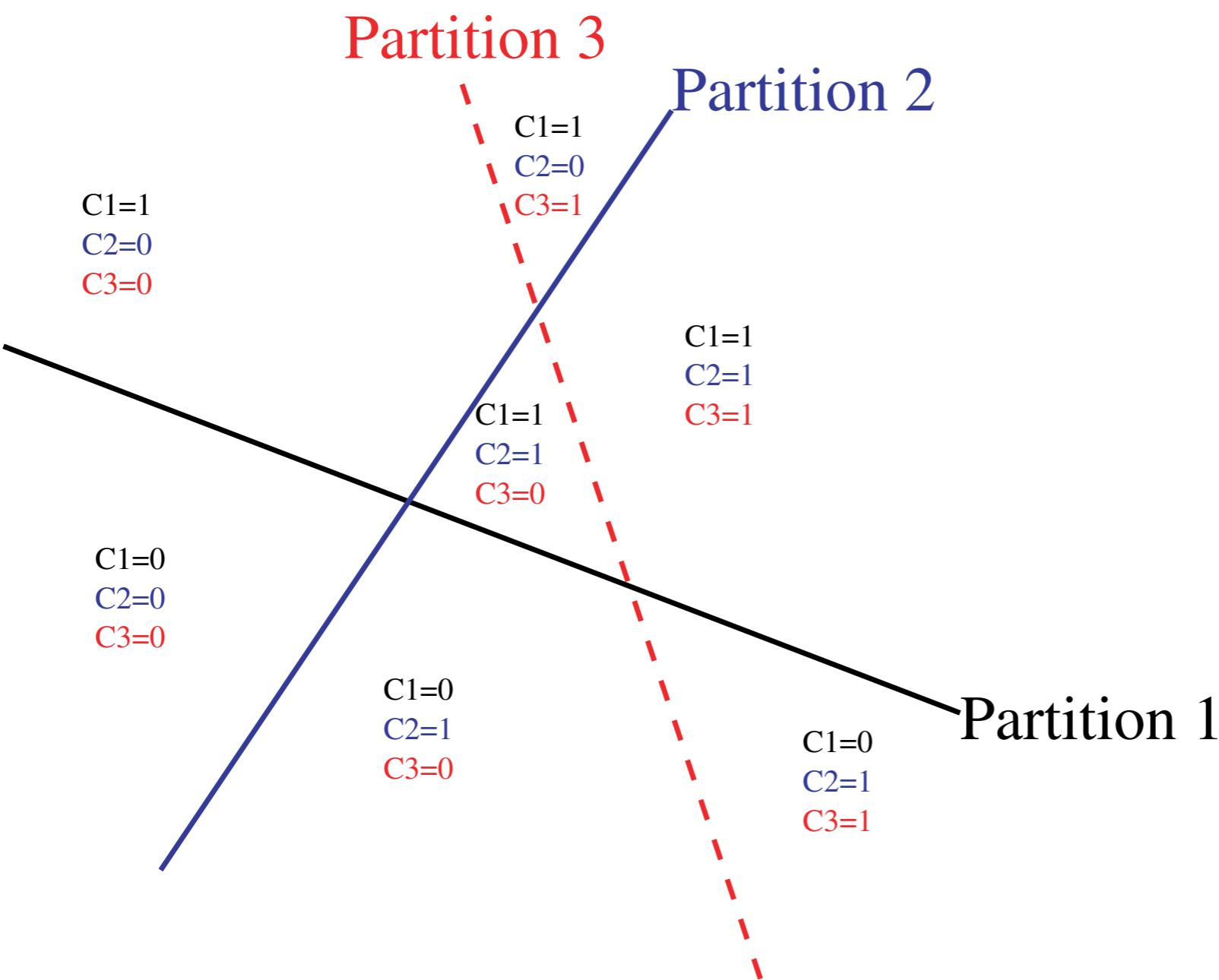
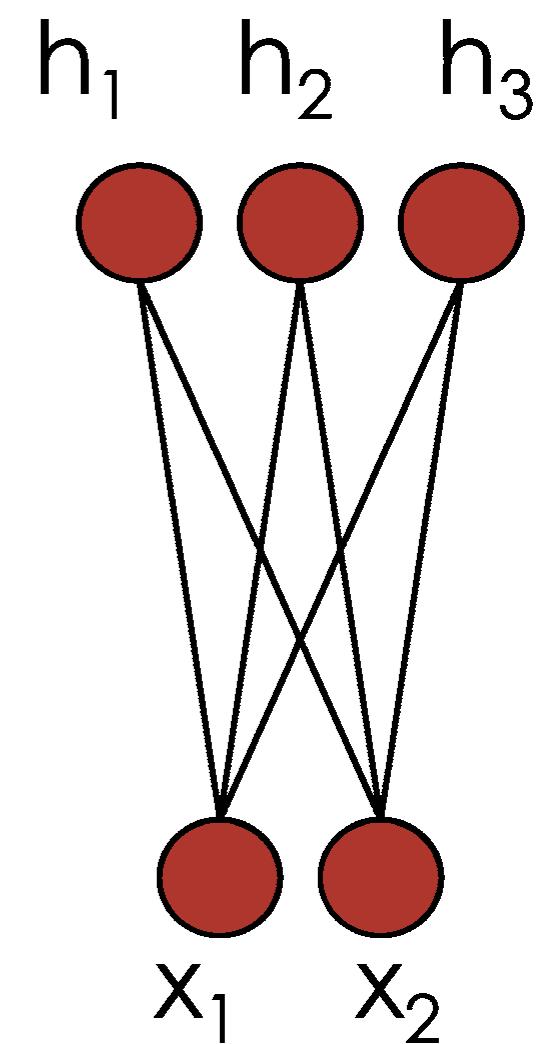
decision trees are local



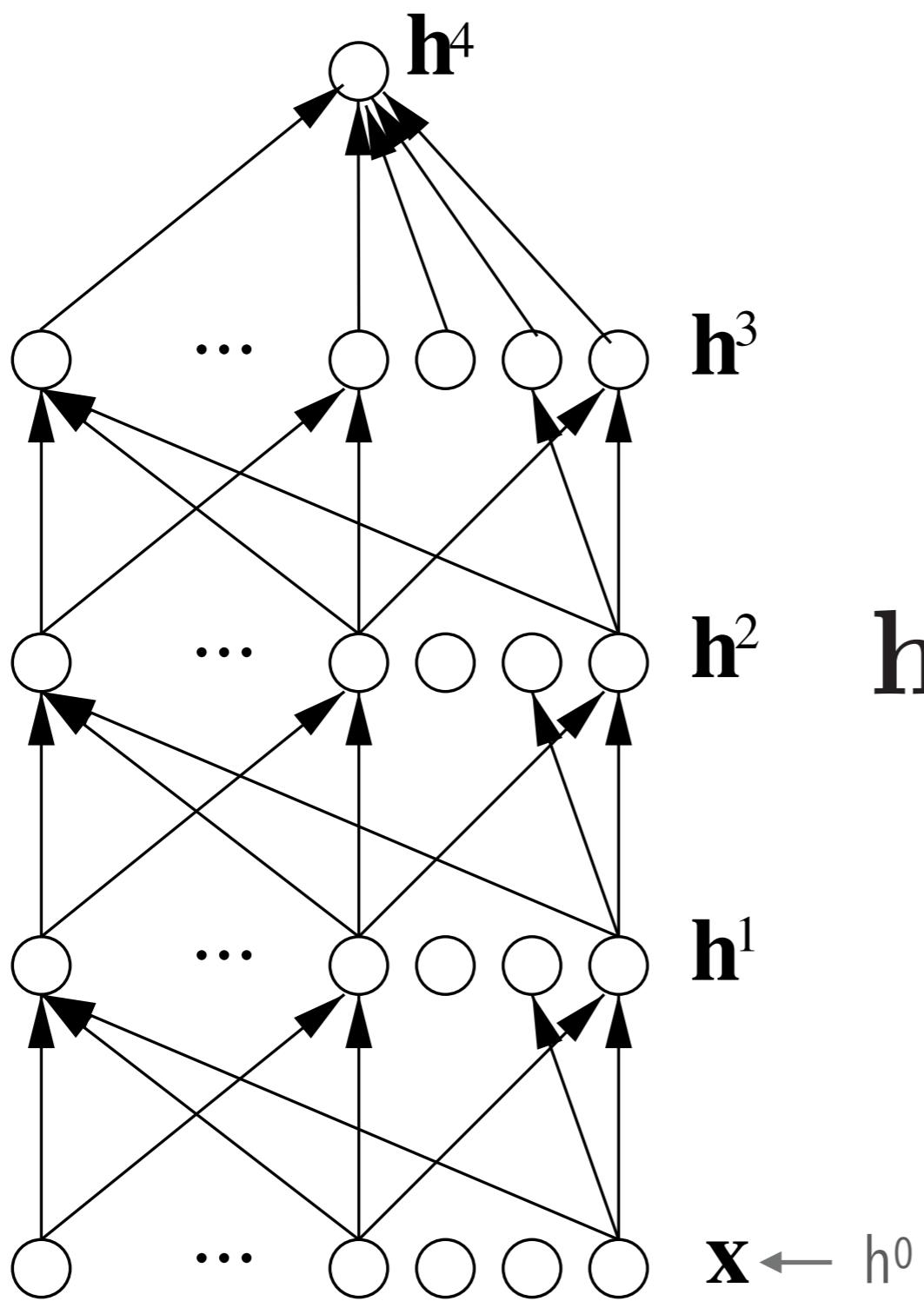
rely on a partition of the input space and using separate parameters for each region with each region associated with a leaf of the decision tree

however tree ensembles are implicitly distributed

RBM with three hidden units

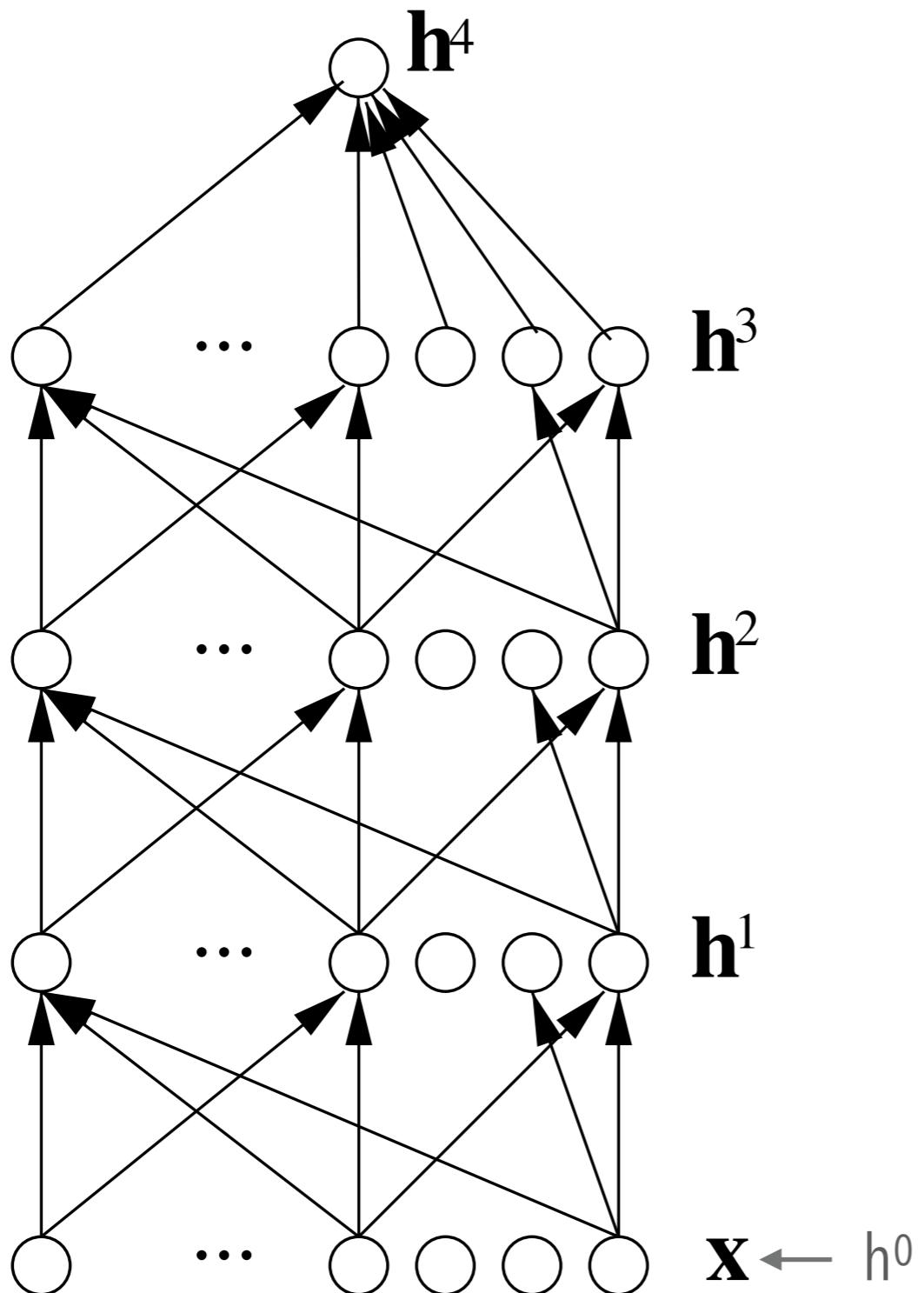


An ensemble of trees can discriminate among a number of regions exponential in the number of trees

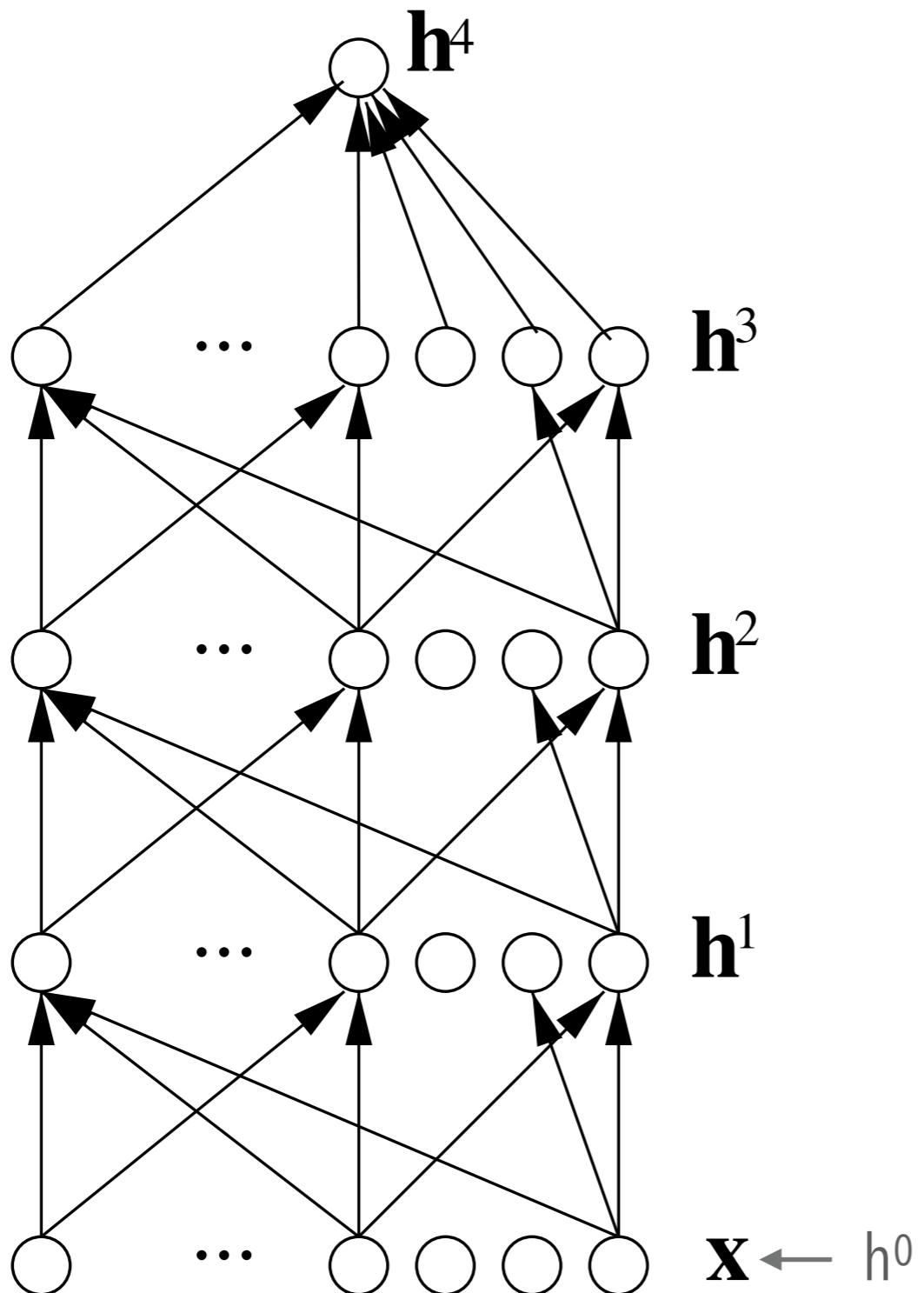


$$\mathbf{h}^k = \tanh(\mathbf{b}^k + W^k \mathbf{h}^{k-1})$$

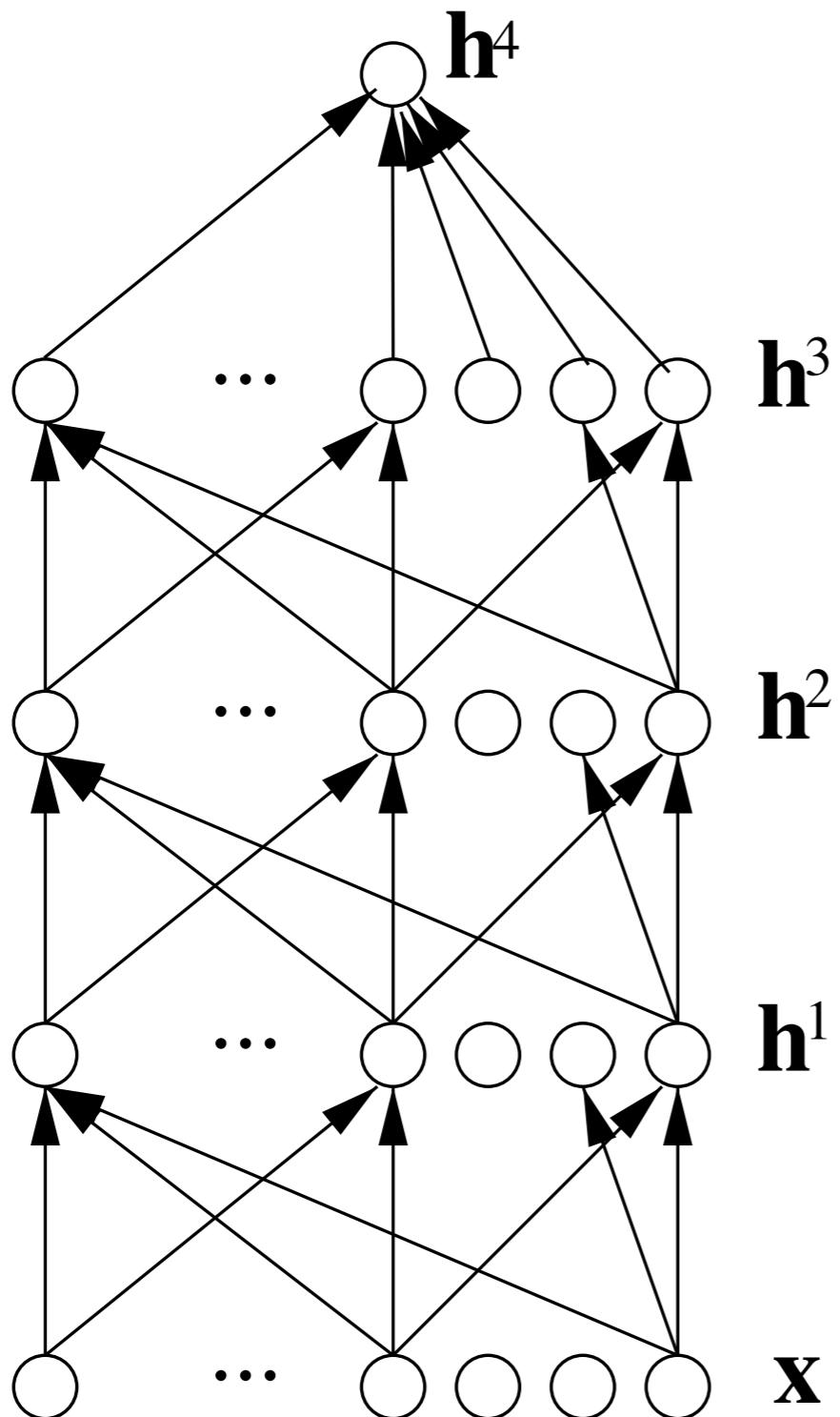
$$h_i^\ell = \frac{e^{\mathbf{b}_i^\ell + W_i^\ell h^{\ell-1}}}{\sum_j e^{\mathbf{b}_j^\ell + W_j^\ell h^{\ell-1}}}$$



what could
be a major
difference
with shallow
networks?



learning
weights of
the various
hidden layers
is hard!



why?

what is it about depth
that makes it hard?



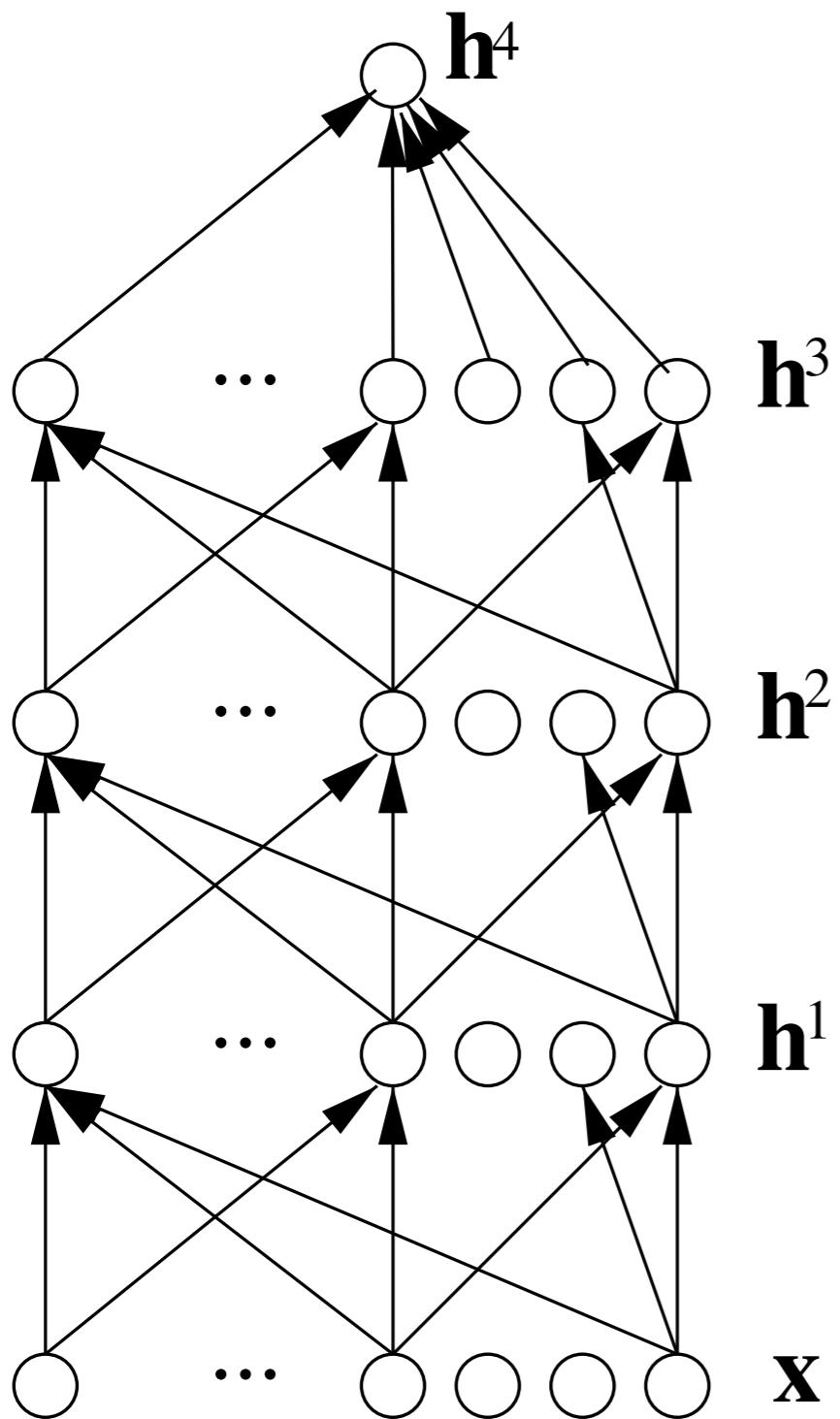
consider a jury

what was his impact in the final decision?



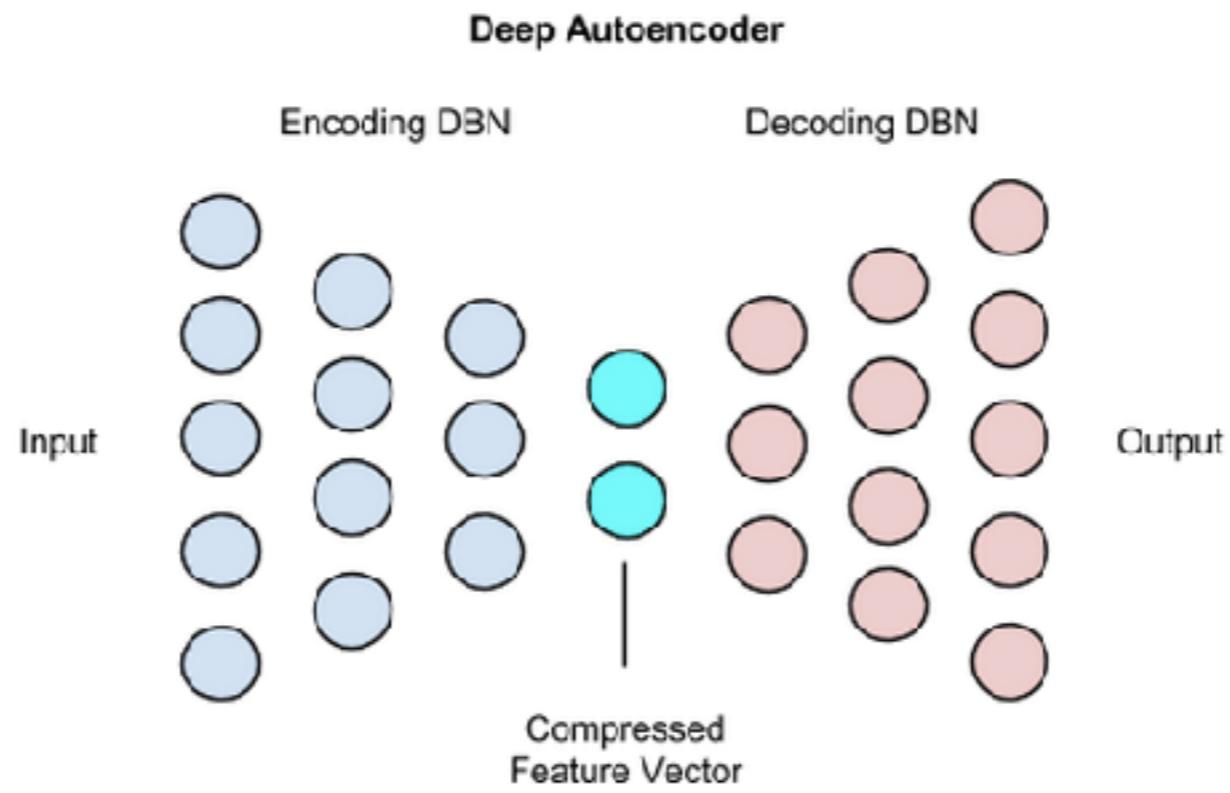
gradients become less meaningful with lots of layers, with random initialization

When starting from random initialization, the solutions obtained with deeper neural networks appear to correspond to poor solutions that perform worse than the solutions obtained for networks with 1 or 2 hidden layers



the major change: unsupervised pre-training

unsupervised pre-training helps
generalization by allowing for a 'better'
tuning of lower layers of a deep architecture.

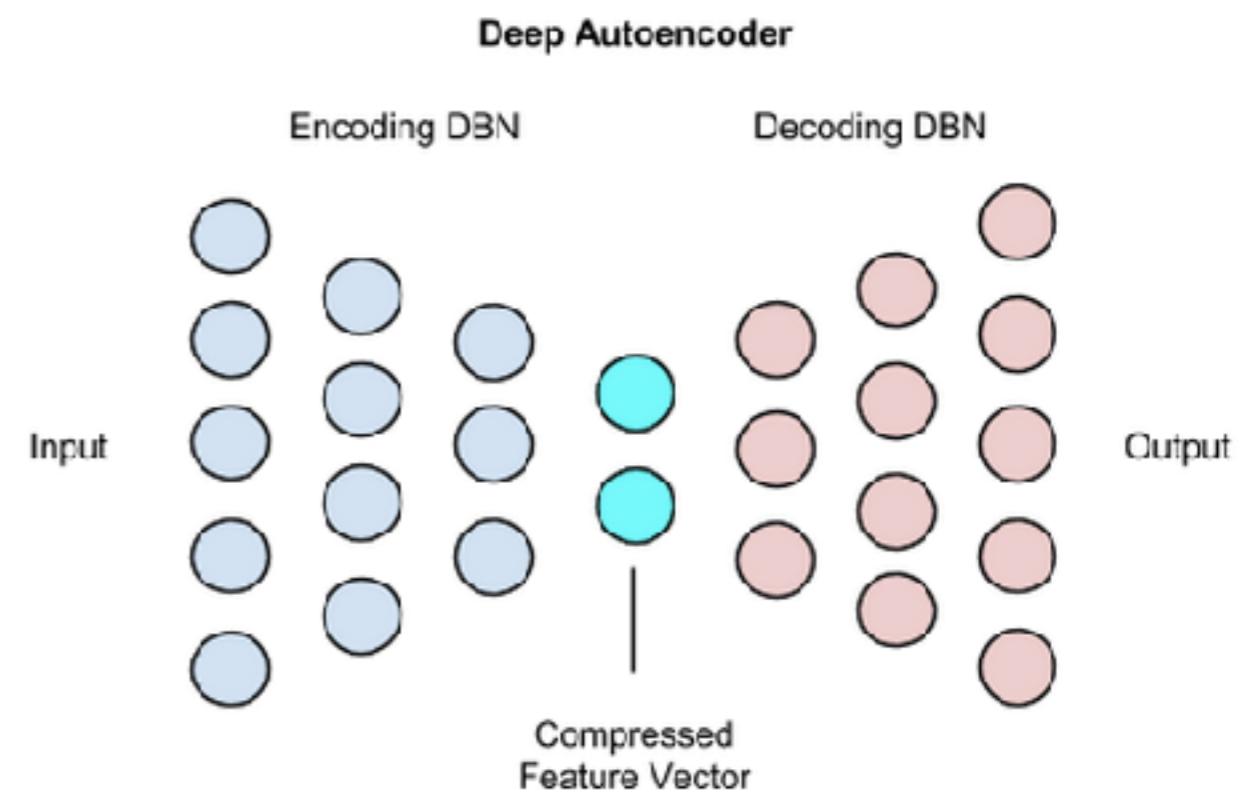


auto-encoders

$$x = f(g(x))$$

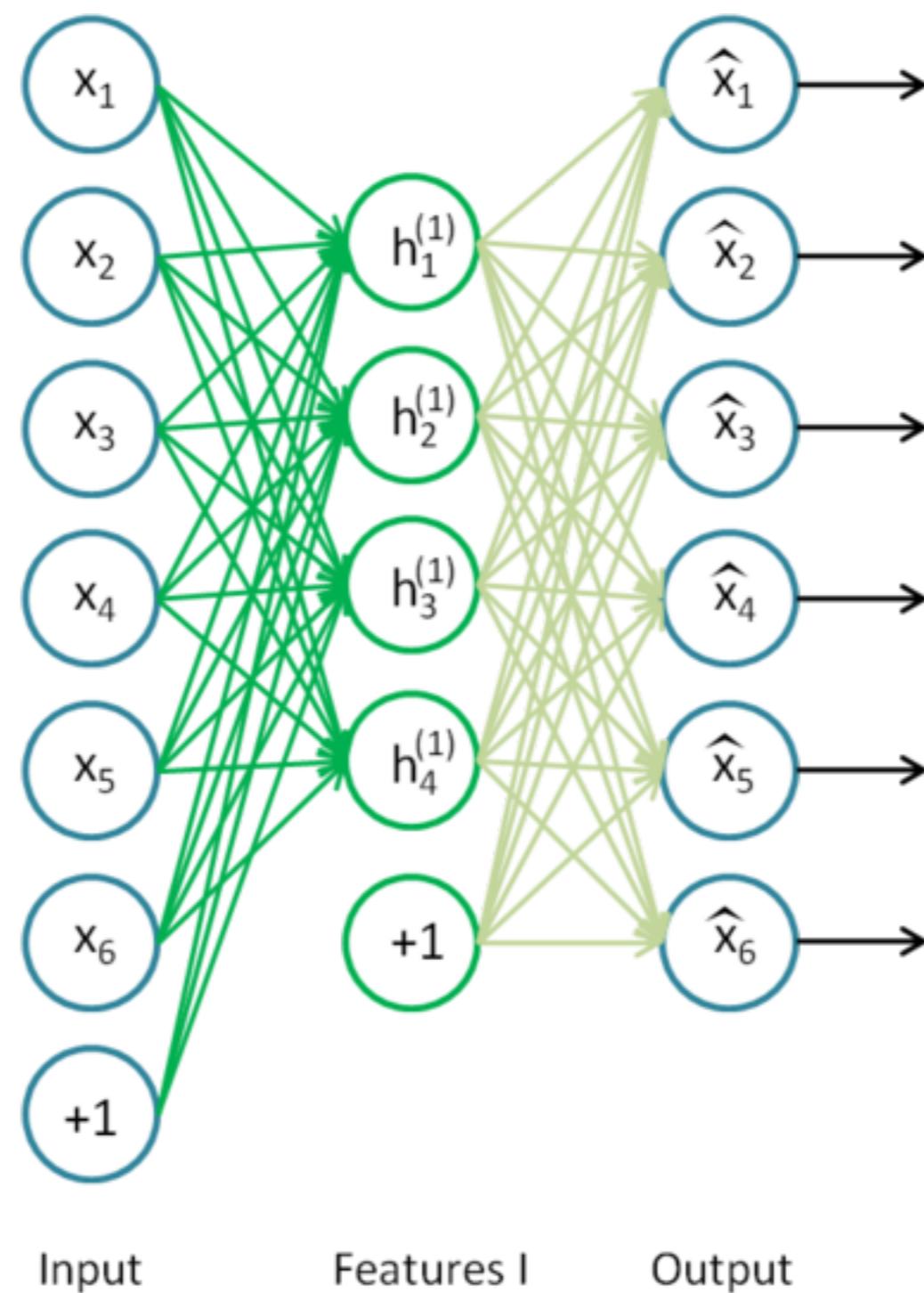
decoder encoder

why don't auto encoders simply learn the identity function?

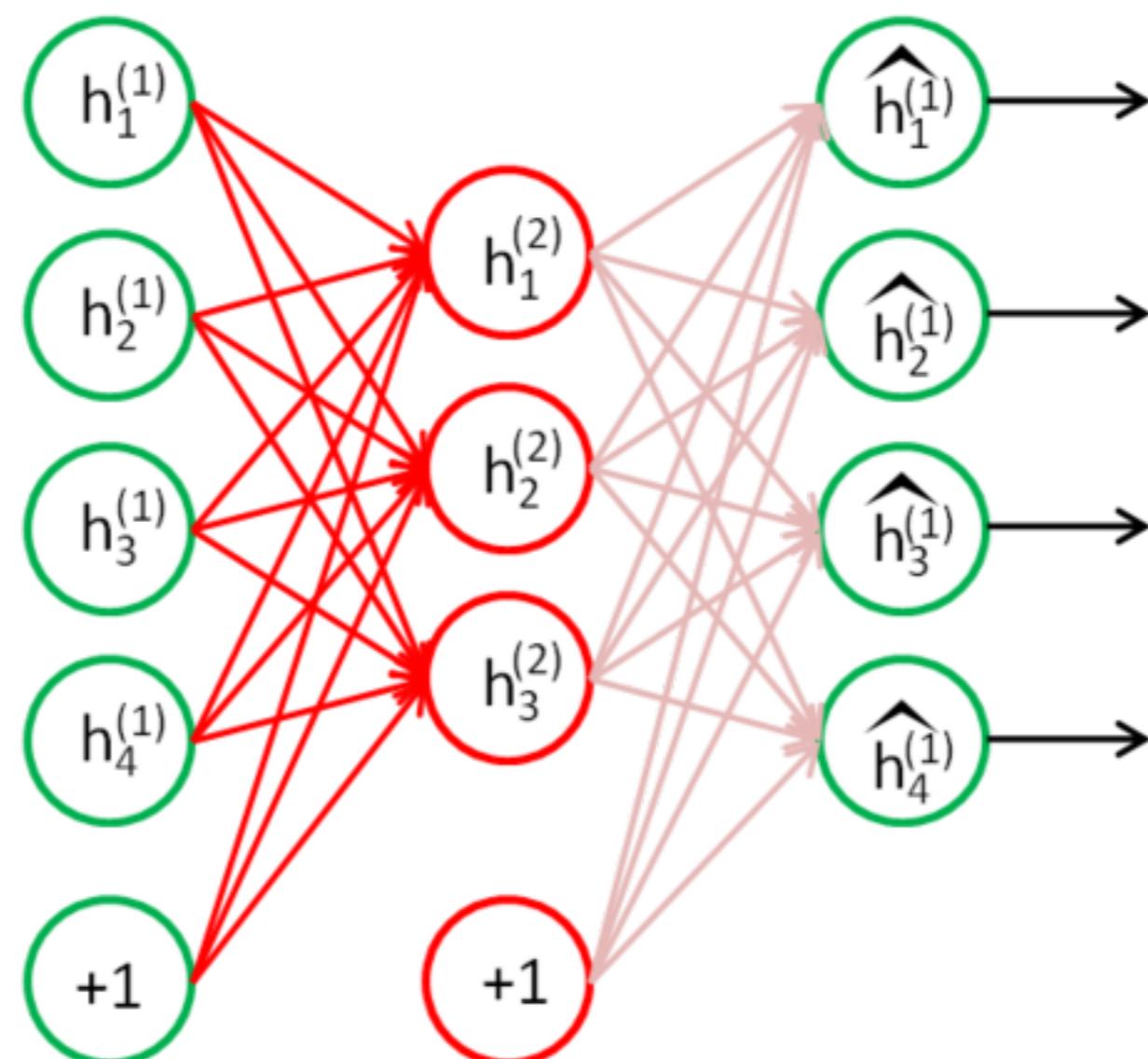


$$x = f(g(x))$$

stacked auto-encoders



stacked auto-encoders

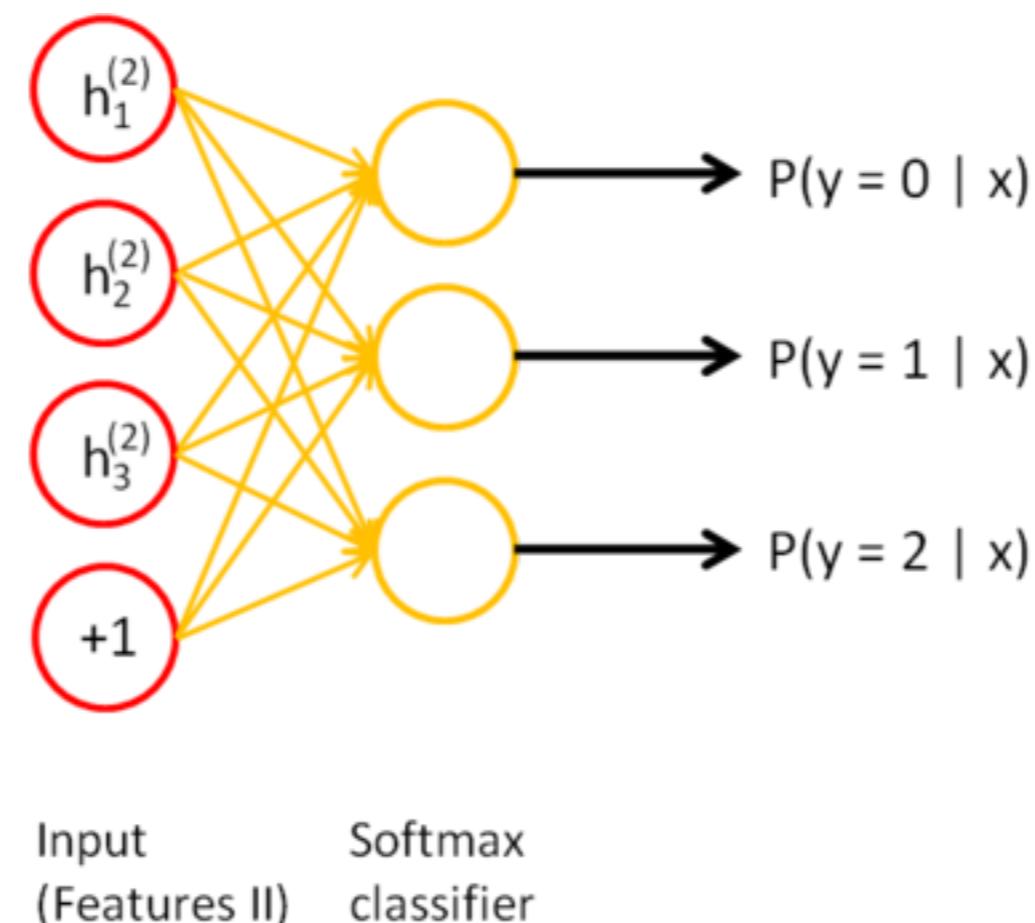


Input
(Features I)

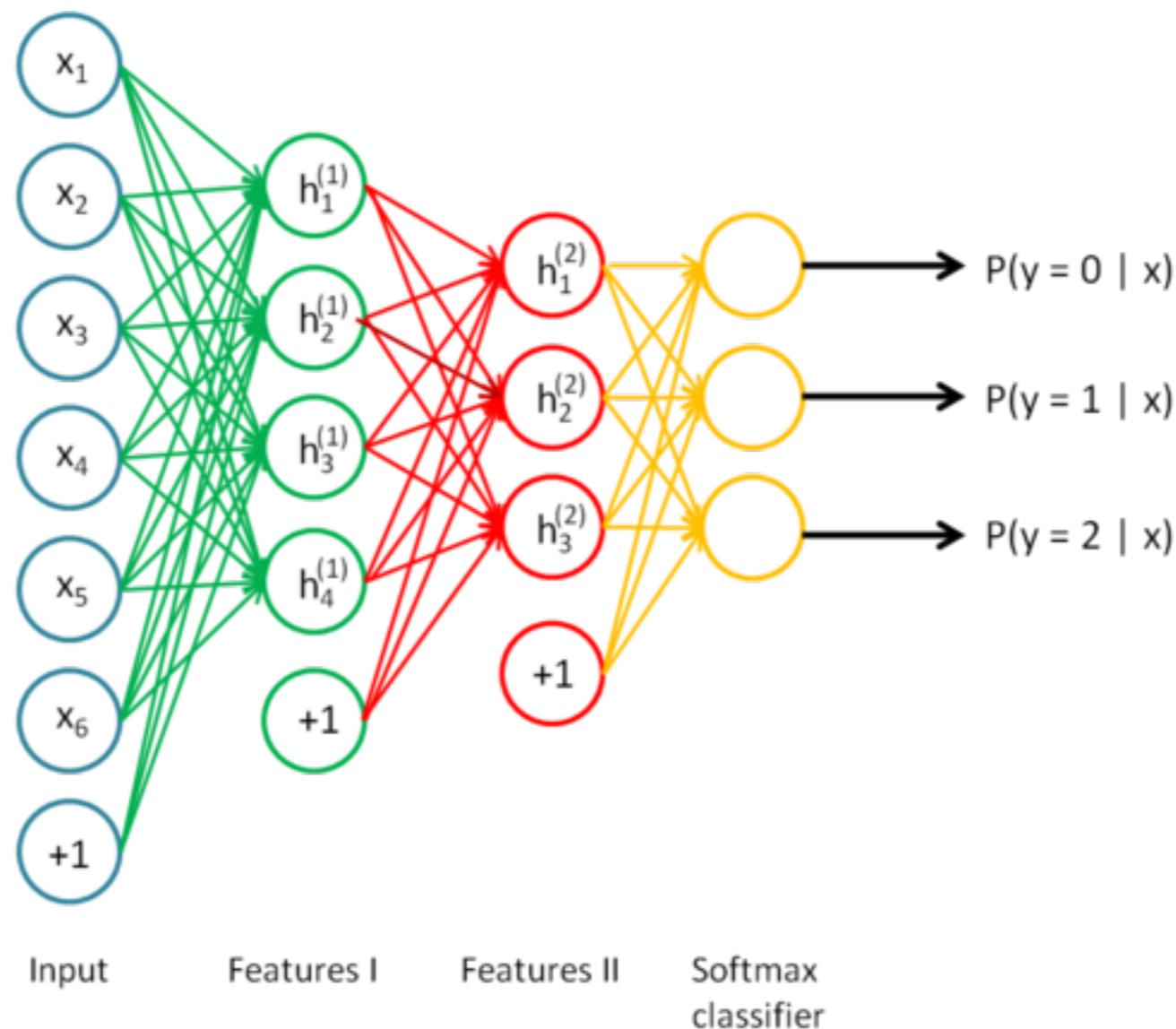
Features II

Output

stacked auto-encoders



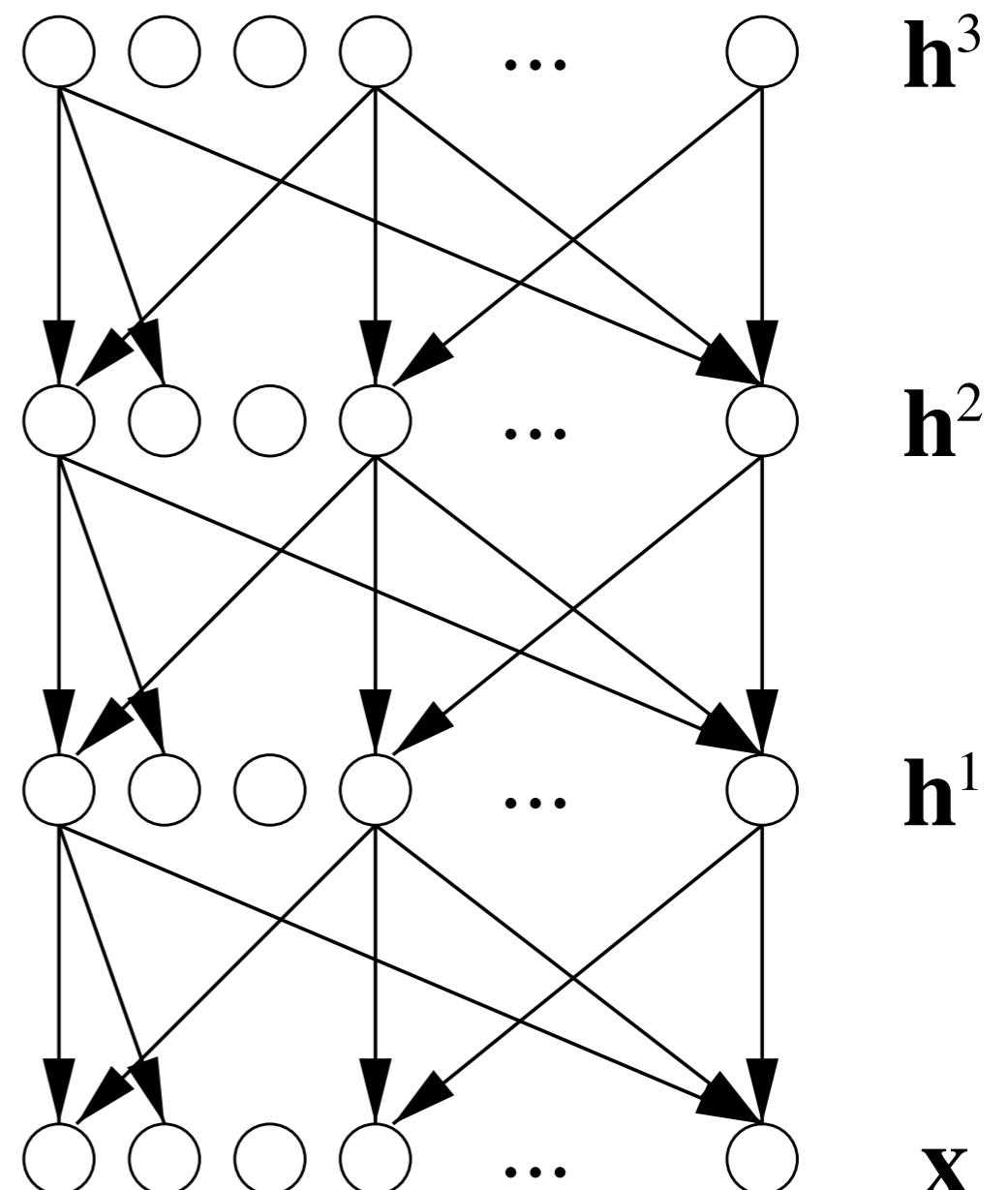
stacked auto-encoders



$$P(\mathbf{x}, \mathbf{h}^1, \dots, \mathbf{h}^\ell) = P(\mathbf{h}^\ell) \left(\prod_{k=1}^{\ell-1} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right) P(\mathbf{x} | \mathbf{h}^1)$$

$$P(\mathbf{h}_i^k = 1 | \mathbf{h}^{k+1}) = \text{sigm}(\mathbf{b}_i^k + \sum_j W_{i,j}^{k+1} \mathbf{h}_j^{k+1})$$

deep generative architectures



note the direction of the arrows

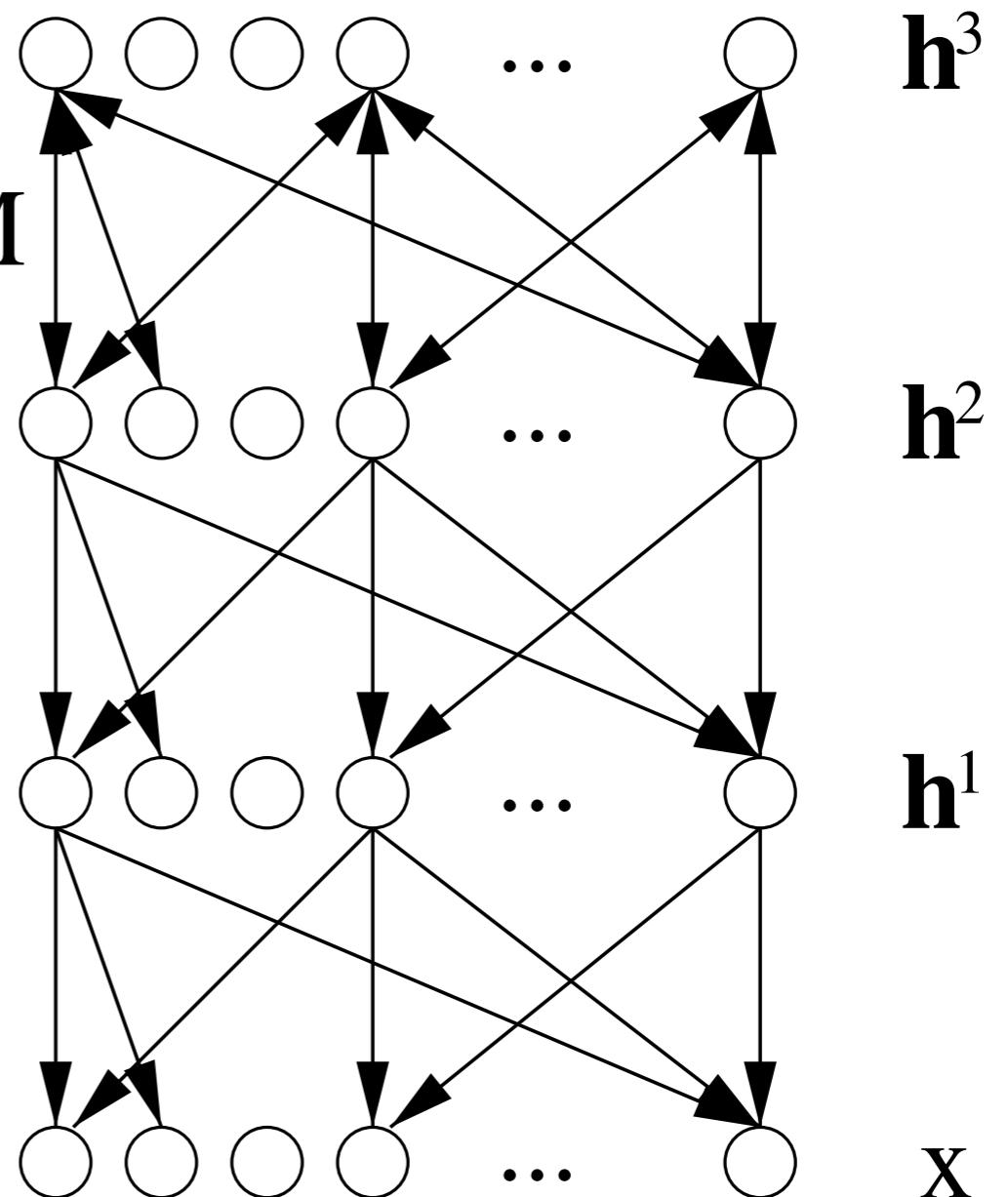
noticed the undirected arrows

$$P(\mathbf{h}^2, \mathbf{h}^3) \sim \text{RBM}$$

$$P(\mathbf{x}, \mathbf{h}^1, \dots, \mathbf{h}^\ell) = P(\mathbf{h}^{\ell-1}, \mathbf{h}^\ell) \left(\prod_{k=1}^{\ell-2} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right) P(\mathbf{x} | \mathbf{h}^1).$$

$$P(\mathbf{h}^{\ell-1}, \mathbf{h}^\ell) \propto e^{\mathbf{b}' \mathbf{h}^{\ell-1} + \mathbf{c}' \mathbf{h}^\ell + \mathbf{h}^\ell' W \mathbf{h}^{\ell-1}}$$

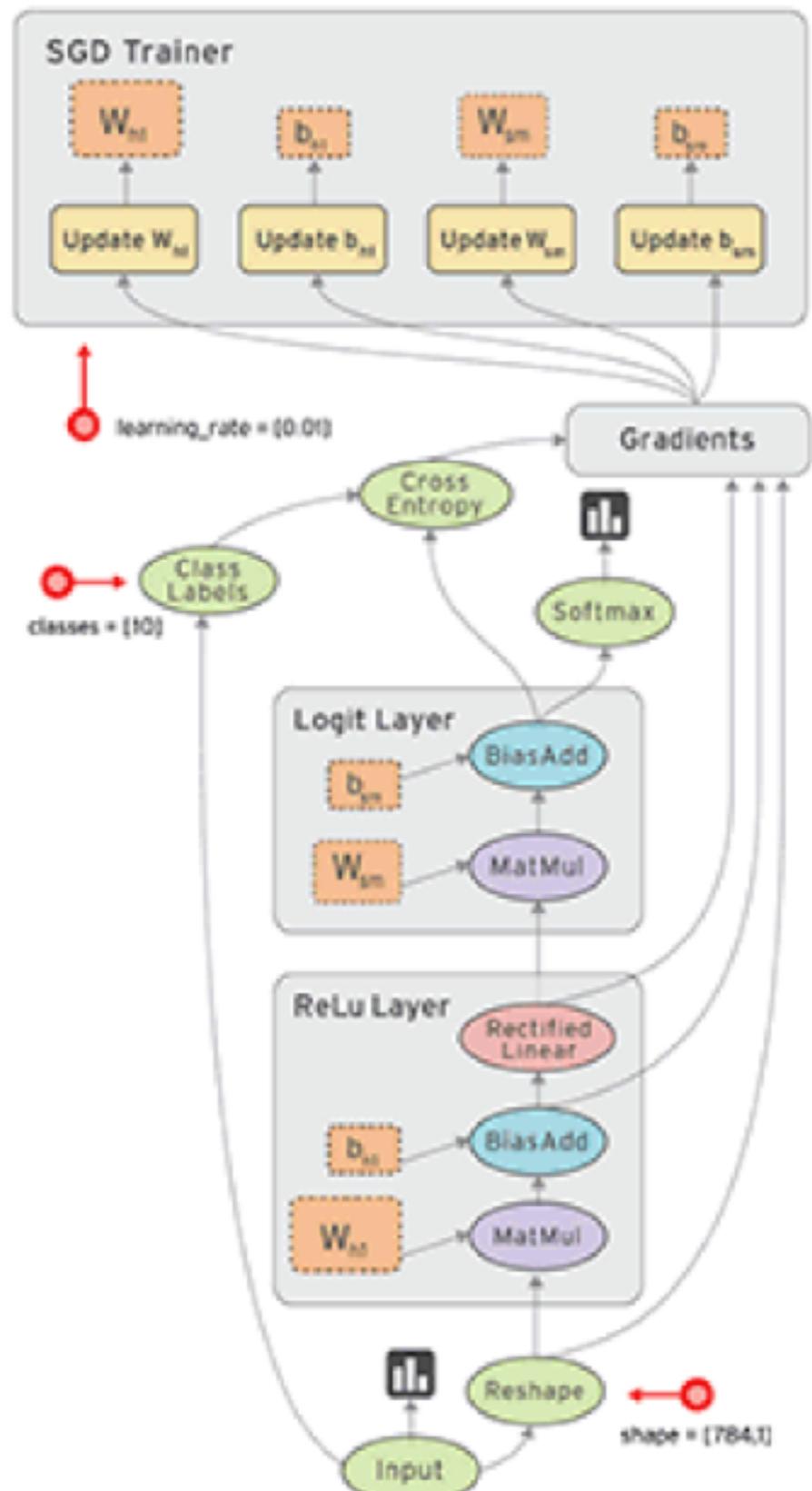
restricted boltzmann machine



$$X_i \rightarrow X_{i+1}$$

Gibbs Sampling

The point of Gibbs sampling is that given a multivariate distribution it is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution.



<https://www.tensorflow.org>

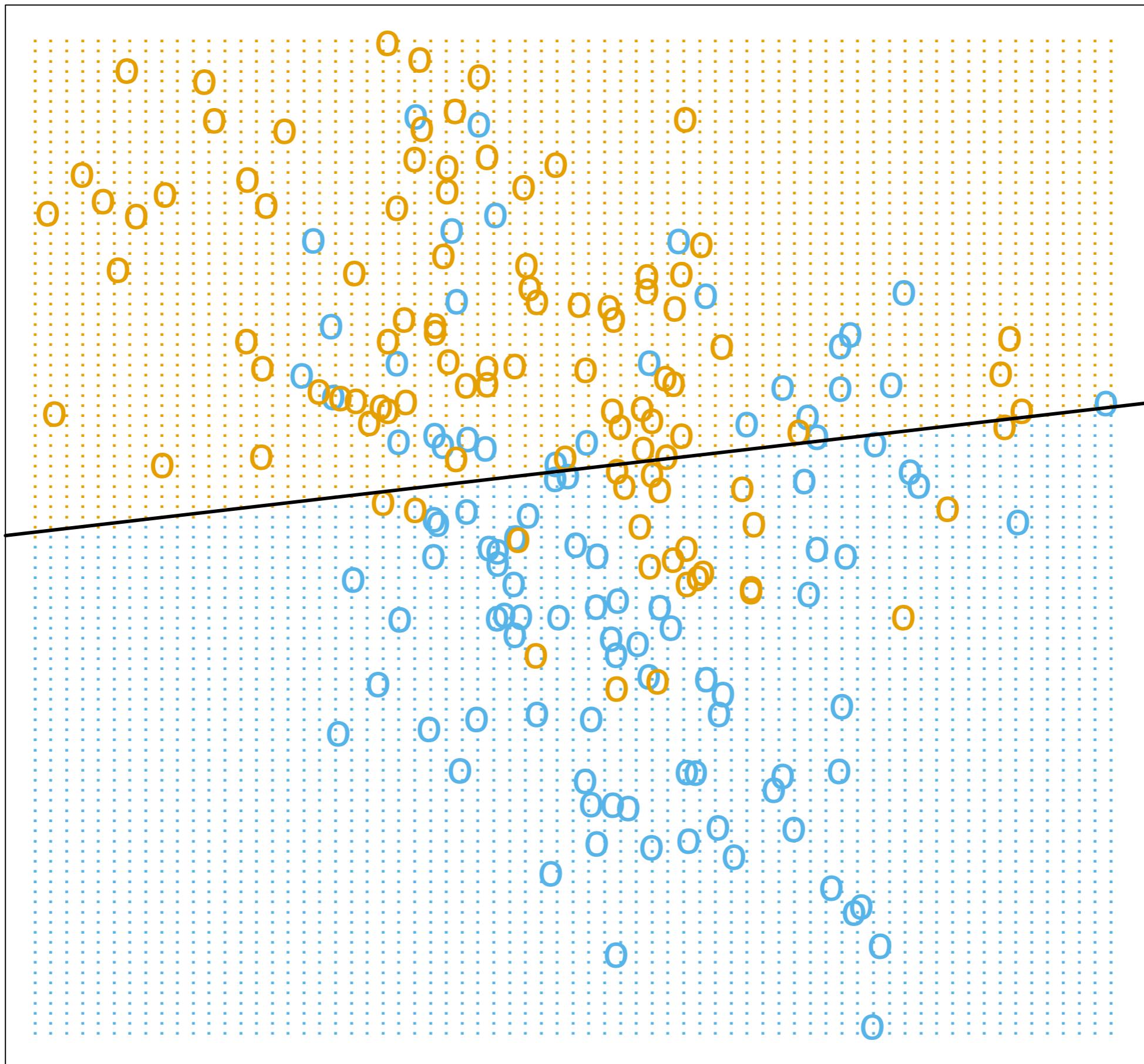
<http://deeplearning.net/software/theano/index.html#>

ETC.

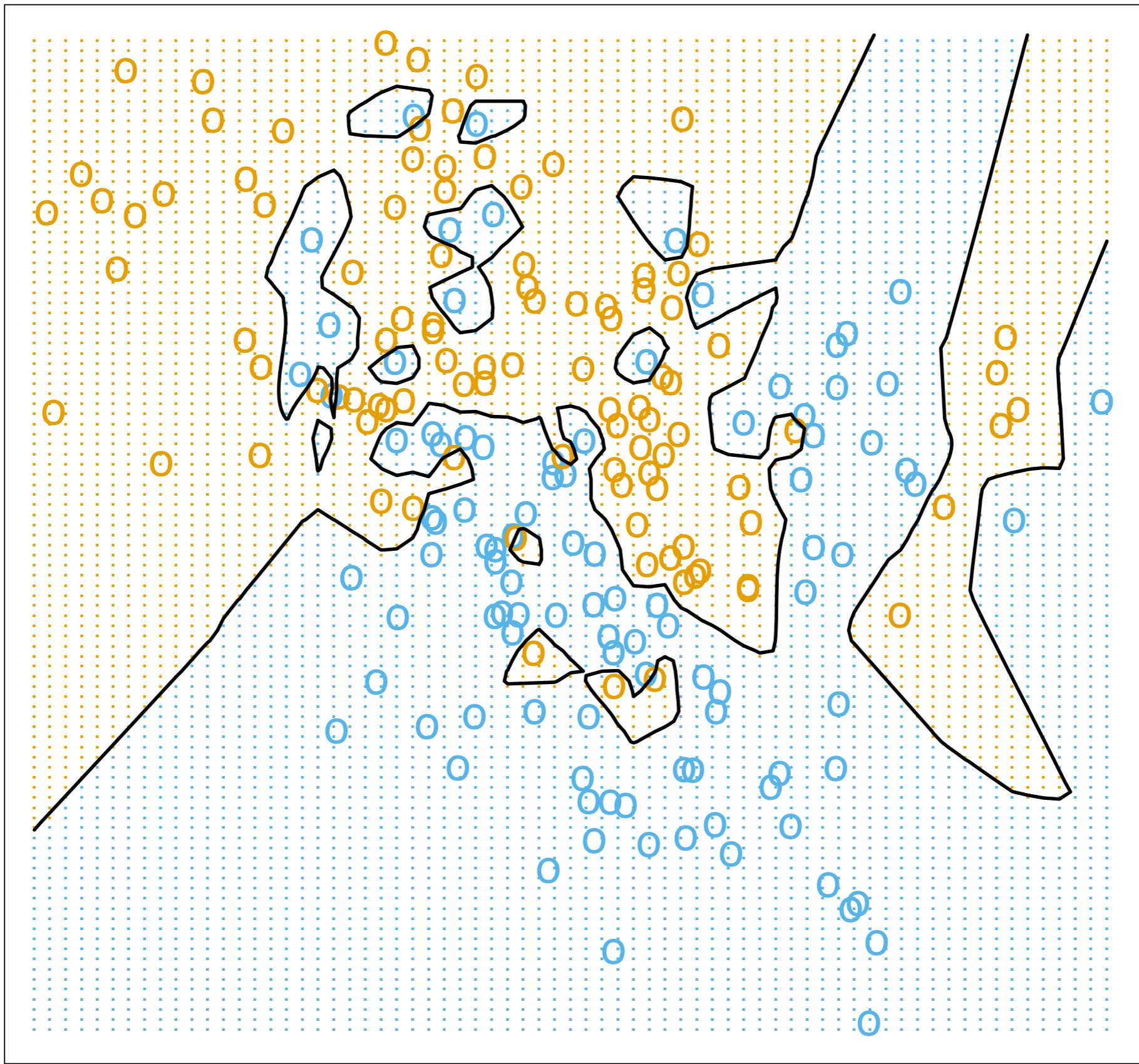
.....

Bayesian Networks Support Vector Machines Neural Networks

Summary

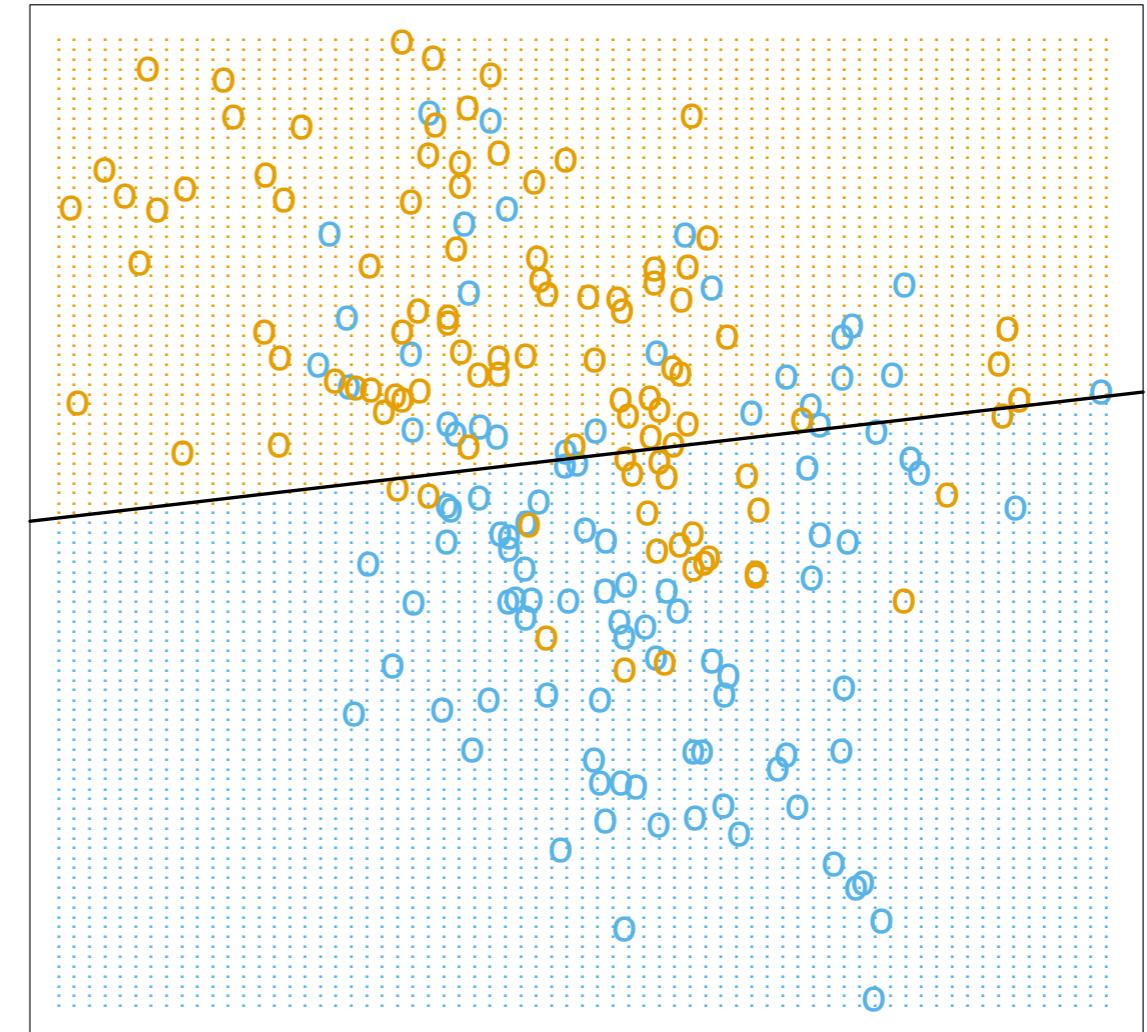
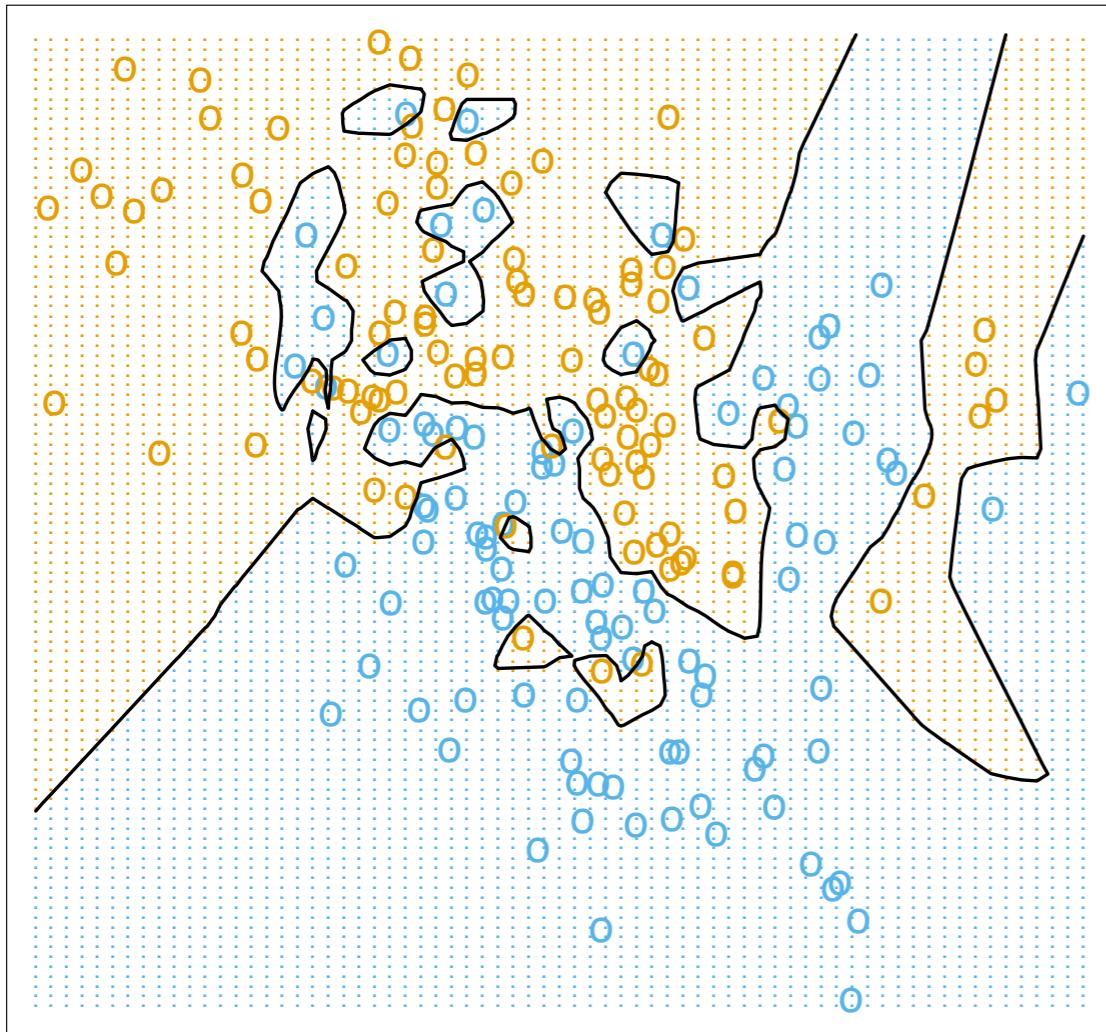


1-Nearest Neighbor Classifier



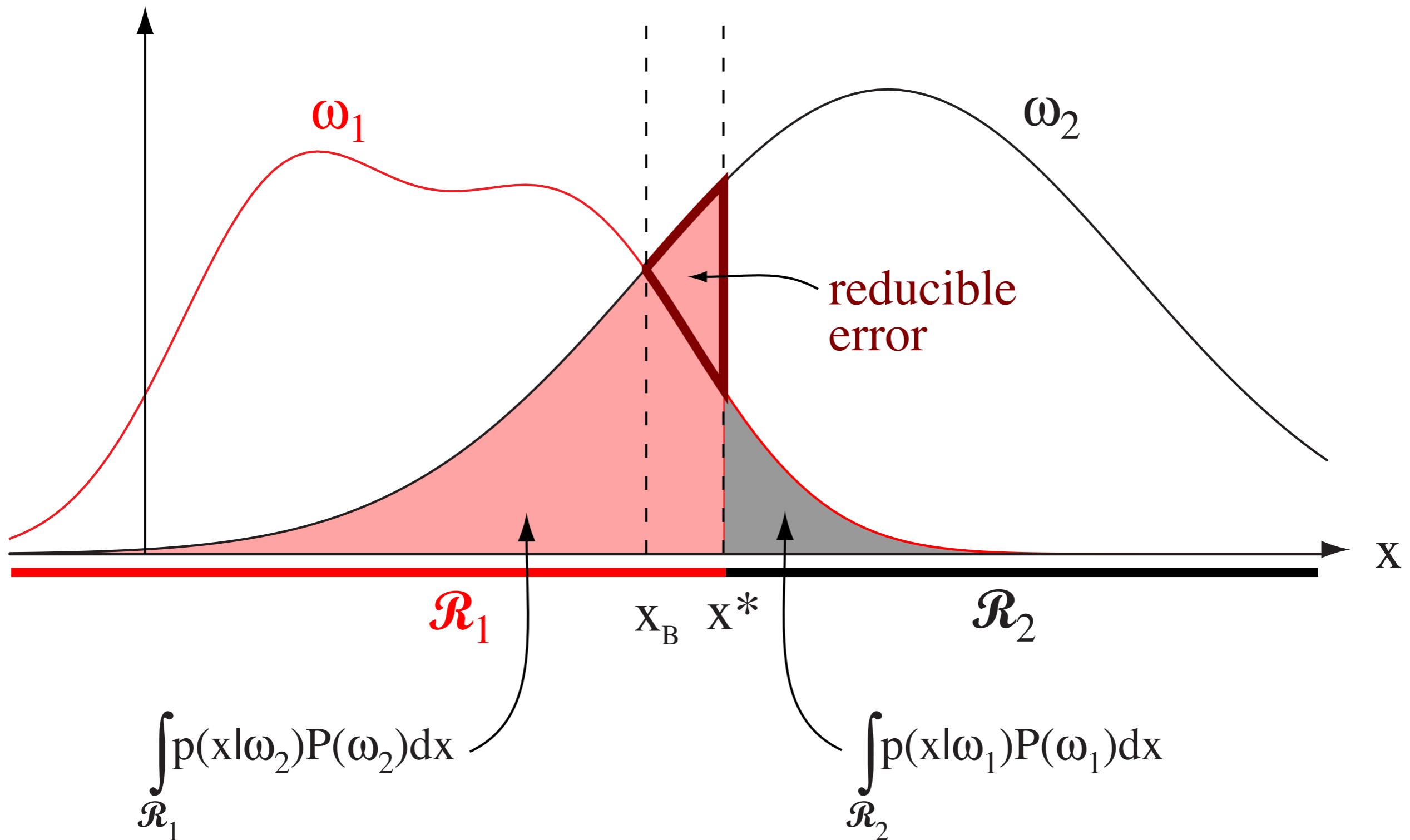
what is the difference?

1-Nearest Neighbor Classifier



1-NN never worse than twice Bayes error

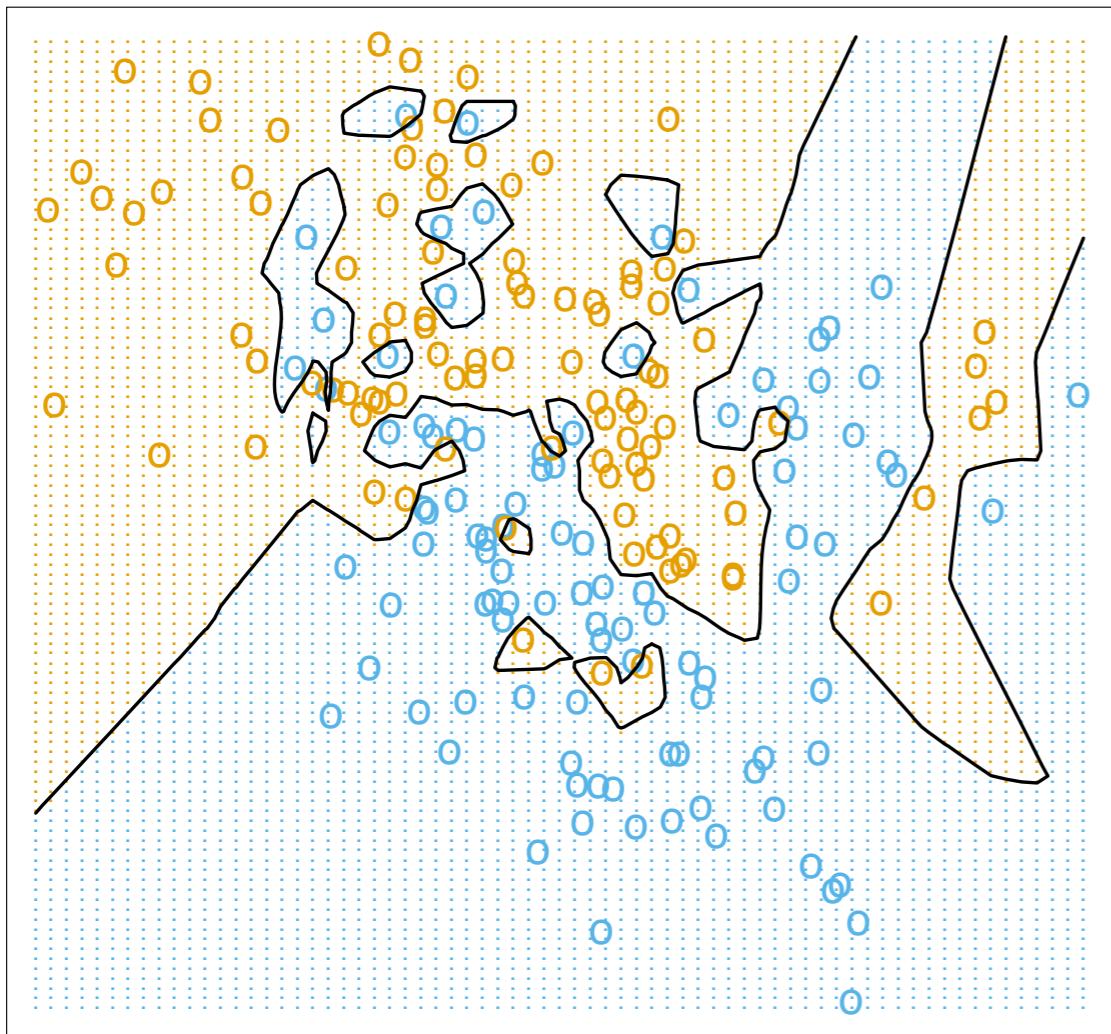
$$p(x|\omega_i)P(\omega_i)$$



K-NN

.....

1-Nearest Neighbor Classifier

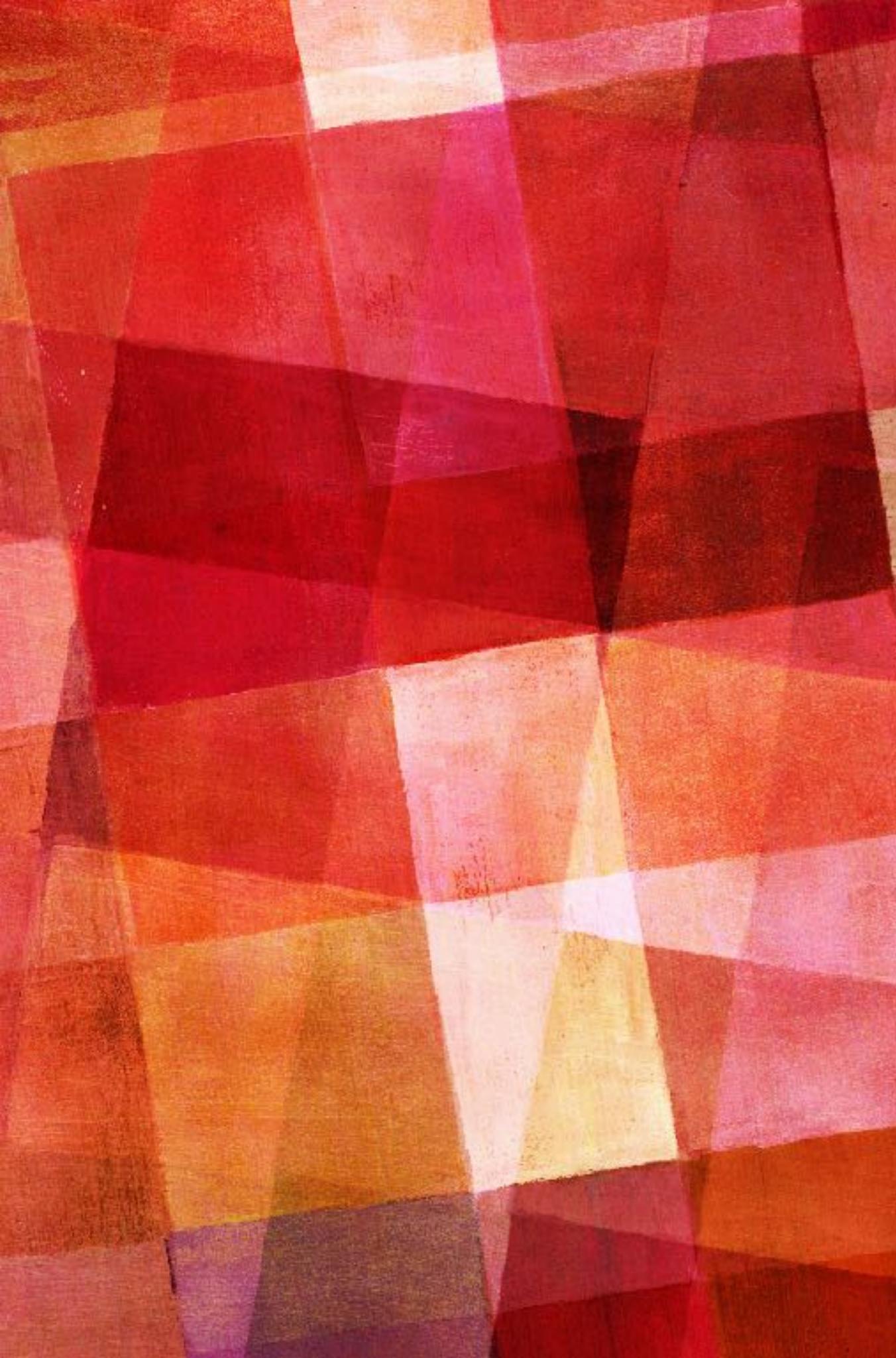


All instances correspond to points in the n-D space

The nearest neighbor are defined in terms of Euclidean distance, $d(\mathbf{X}_1, \mathbf{X}_2)$

Target function could be discrete or real valued

For discrete-valued, **k**-NN returns the most common value among the k training examples nearest to \mathbf{x}_q



MULTI CLASS

Method 1. One-vs.-all: Learn a classifier one at a time

Given m classes, train m classifiers: one for each class

Classifier j : treat tuples in class j as positive & all others as negative

To classify a tuple X , the set of classifiers vote as an ensemble

Method 2. All-vs.-all: Learn a classifier for each pair of classes

Given m classes, construct $m(m-1)/2$ binary classifiers

A classifier is trained using tuples of the two classes

To classify a tuple X , each classifier votes. X is assigned to the class with maximal vote

Build a classifier using the labeled data

Use it to label the unlabeled data,
and those with the most confident
label prediction are added to the
set of labeled data

Repeat

semi-supervised

Uses labeled and unlabeled data to build a classifier

Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f_1

Then f_1 and f_2 are used to predict the class label for unlabeled data X

The tuple having the most confident prediction from f_1 is added to the set of labeled data for f_2 , & vice versa

co-training



ACTIVE LEARNING

.....



Class labels are expensive to obtain

Active learner: query human (oracle) for labels

Pool-based approach: Uses a pool of unlabeled data

L: a small subset of D is labeled, U: a pool of unlabeled data in D

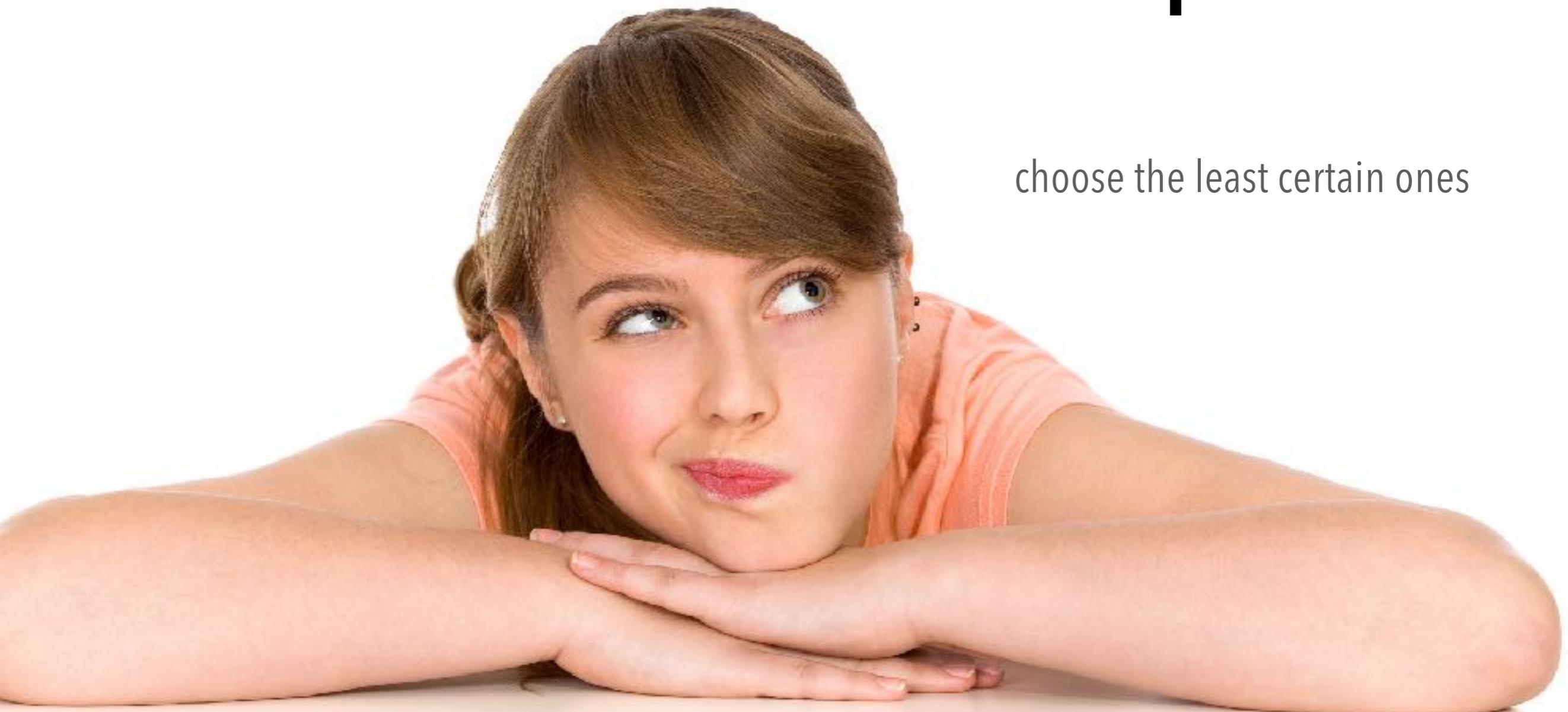
Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)

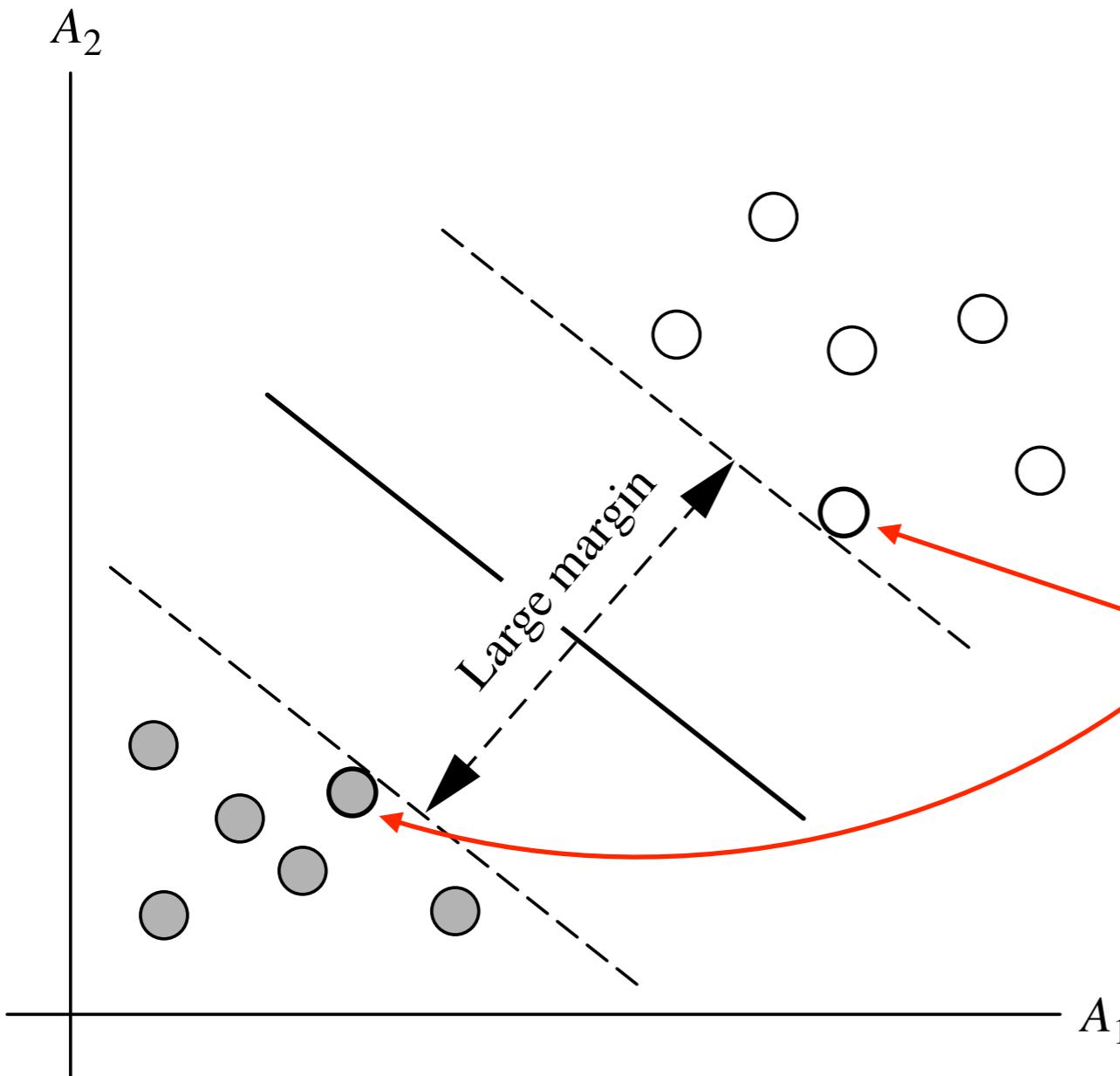
The newly labeled samples are added to L, and learn a model

Goal: Achieve high accuracy using as few labeled data as possible

How to choose the data tuples to be queried?

choose the least certain ones







transfer learning

works best when the data shares regularities