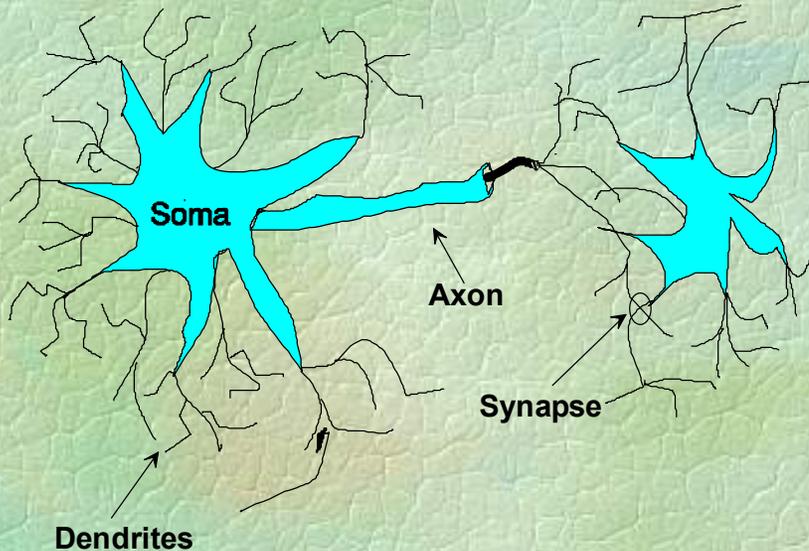


# Нейронные сети

Хомичёва Ирина Вадимовна  
аспирант лаборатории Теоретическая генетика

# Биологическая мотивация

**Нервная система:  $10^{11}$   
нейронов участвуют в  $10^{15}$   
передающих связях**

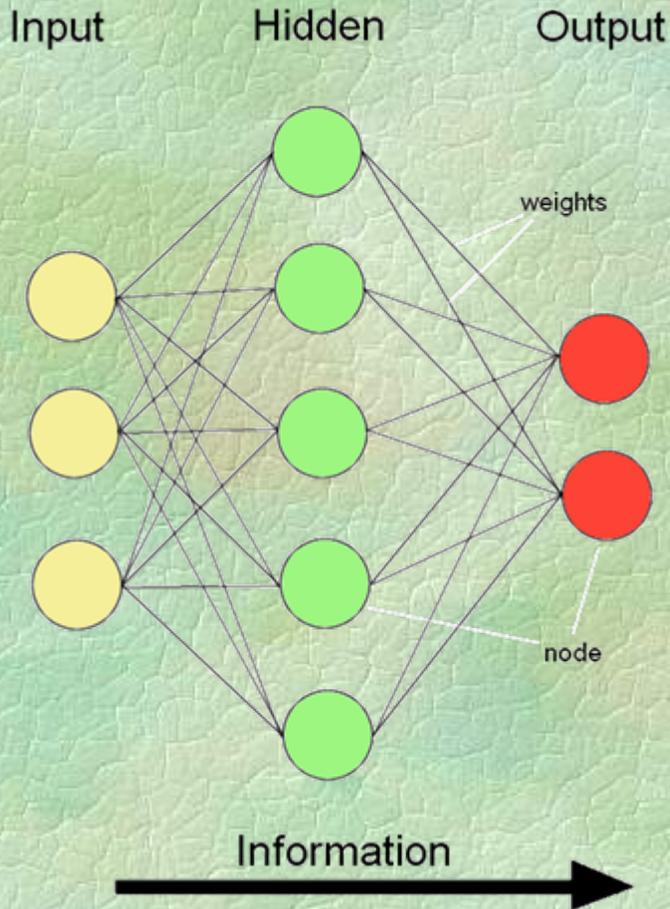


Уникальной способностью нейрона является приём, обработка и передача электрохимических сигналов по нервным путям.

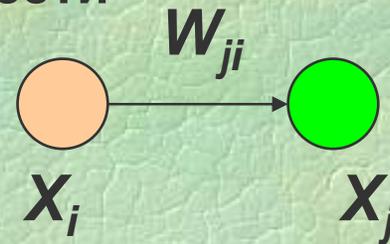
Каждый нейрон принимает взвешенную сумму входных сигналов и при определённых условиях имеет возможность передавать сигнал дальше.

**Искусственные нейронные сети есть попытка воспроизвести способность нервных биологических систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга**

# Нейронная сеть-есть ориентированный мультиграф со взвешенными взаимосвязями.



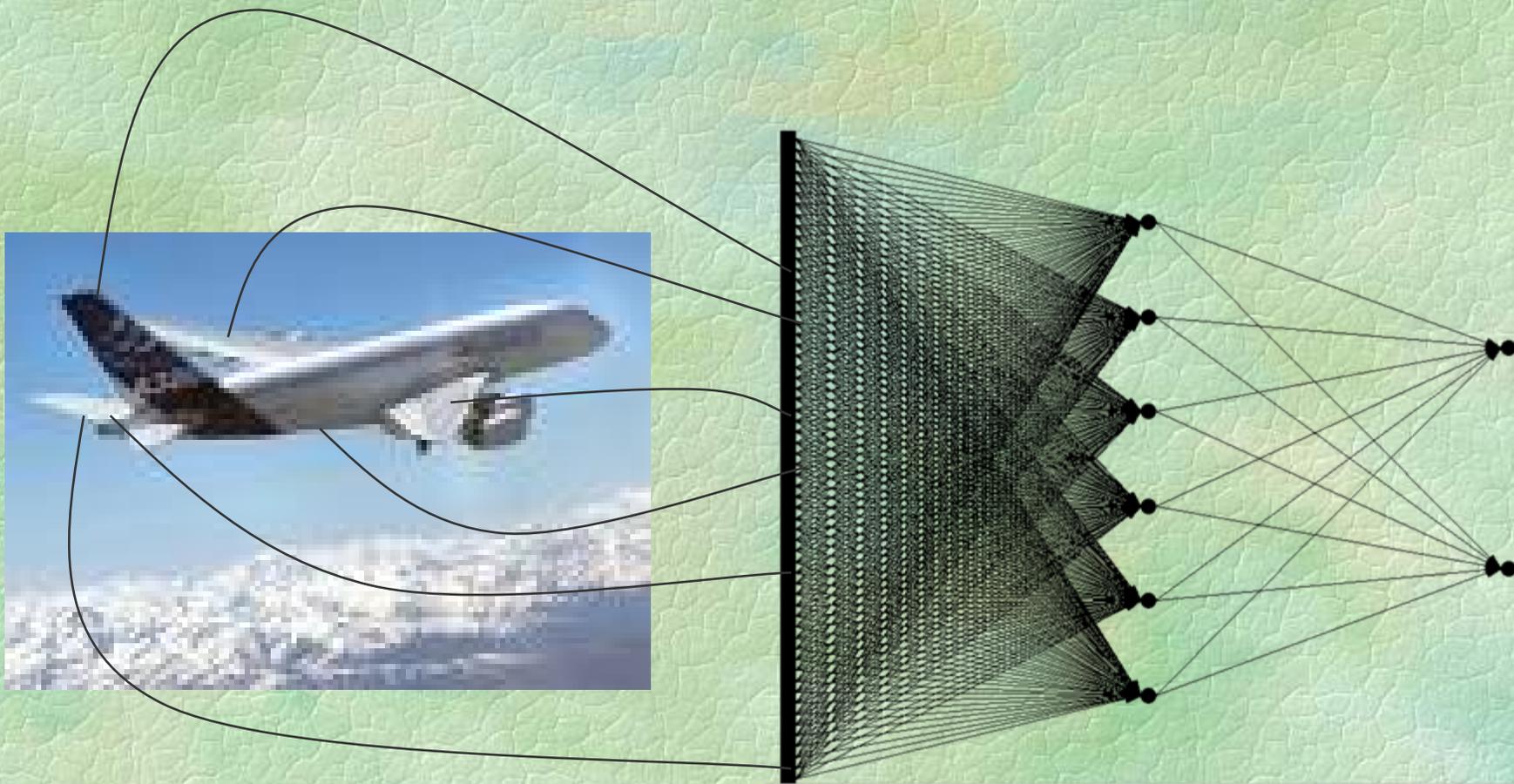
Множество вершин - множество нейронов сети



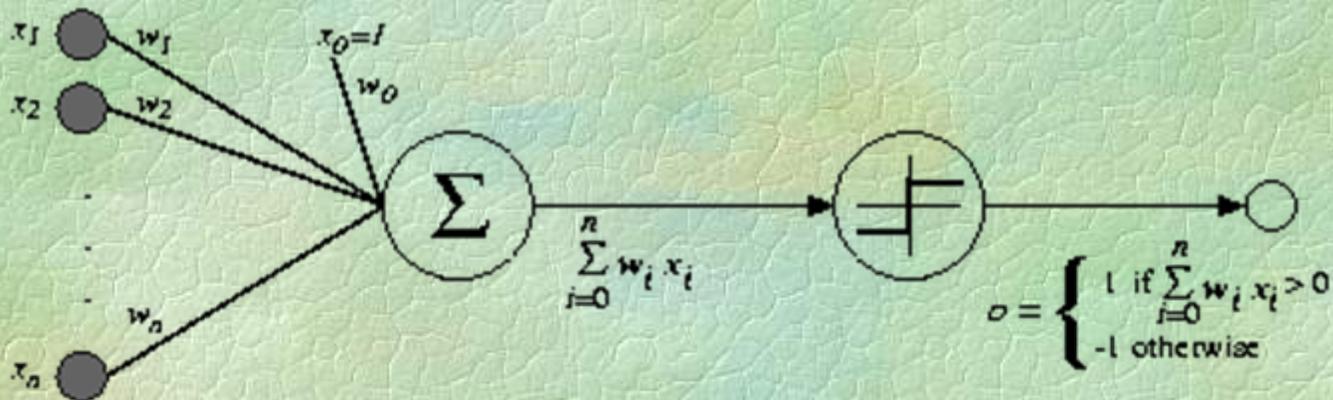
$W_{ji}$  - вес пути от вершины  $X_i$  к  $X_j$   
параметр сети

Обучение нейронной сети – изменение параметров сети

# Применение нейронной сети для контроля технического состояния авиалайнера



# Модель нейрона



$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \dots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

# Основные парадигмы обучения нейронной сети



обучение с оракулом  
(supervised learning)

обучение без оракула  
(unsupervised  
learning)

reinforcement  
learning

*feed-forward nn*  
(*Backpropagation*)

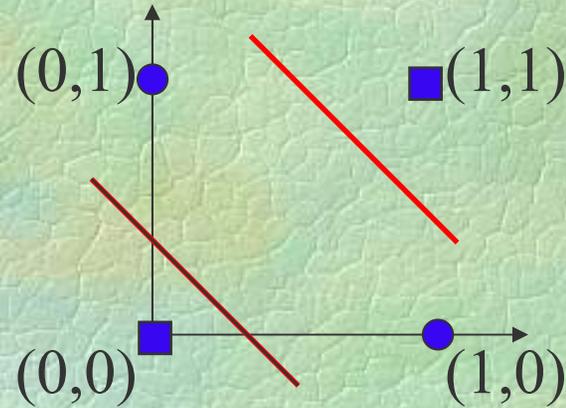
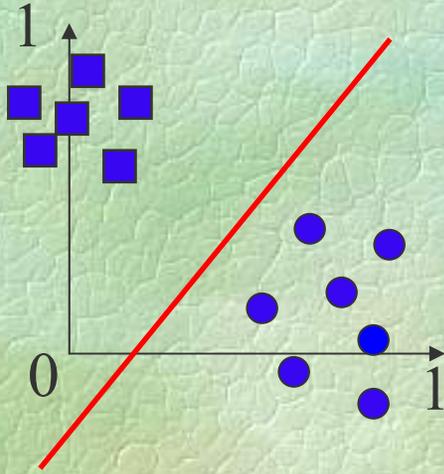
*time-lagged nn*

*recurrent nn*

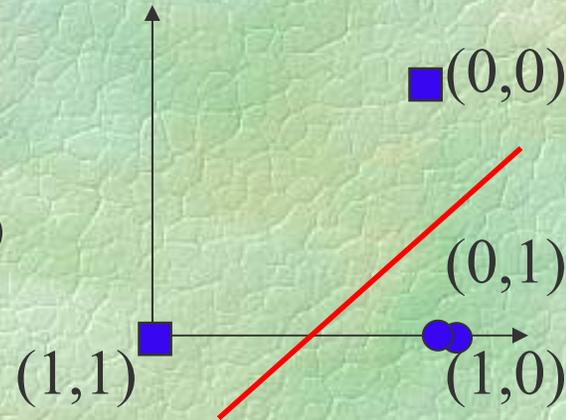
*Kohonen network*

*Principal Component  
Analysis*

# Линейная отделимость. Проблема XOR



Данные называются линейно-отделимыми, если существуют гиперплоскость, однозначно разделяющая их на два класса



# Обучение простейшей сети в случае линейно-отделимого пространства обучающих данных

Обучение простейшей сети, состоящей из одного нейрона, может происходить в соответствии с правилом Видроу-Хоффа (или дельта-правилом).

Значение целевой функции для некоторого примера:  $t$ .

Выходное значение нейрона:  $o$ .

Инициализировать веса случайным образом.

Итеративно исправлять веса после каждого акта применения сети к данным:

$\Delta w_i = \eta (t - o) x_i$  *правило Видроу-Хоффа или*

$w_j \leftarrow w_j + \Delta w_j$  *дельта-правило*

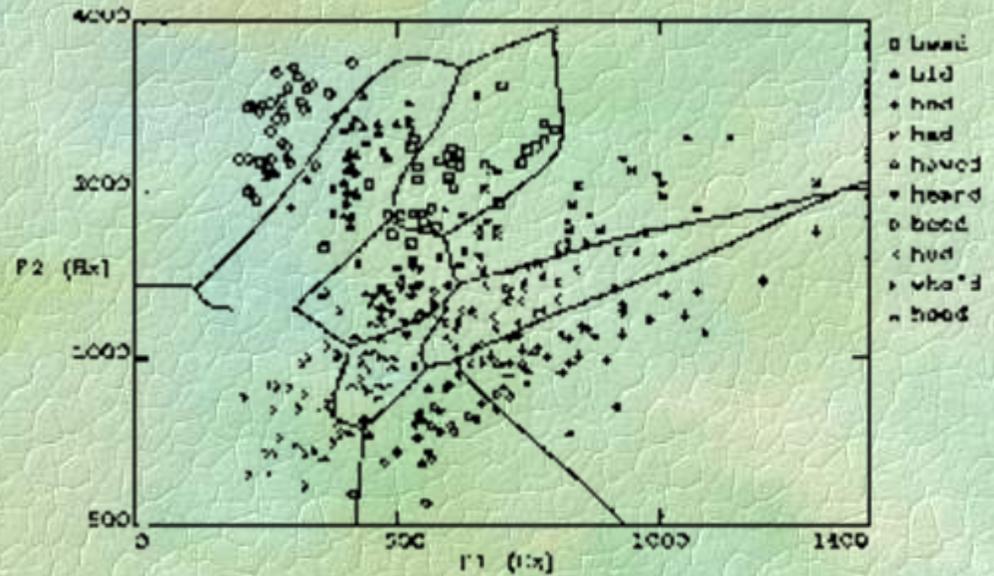
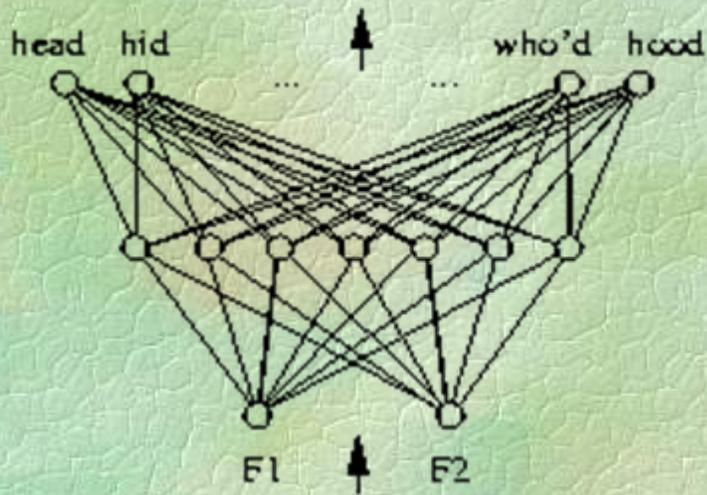
$\eta$  - норма обучения, параметр, который влияет на скорость сходимости правила.

Для того, чтобы убедиться, что правило работает, выберем некоторые крайние точки значений целевой функции и нейрона. Если  $t=o$  для некоторого входного вектора, то веса сети не меняются, результат достигнут. Если же  $t = +1$ , а  $o = -1$ , то  $\Delta w_i = \eta 2 x_i$ .

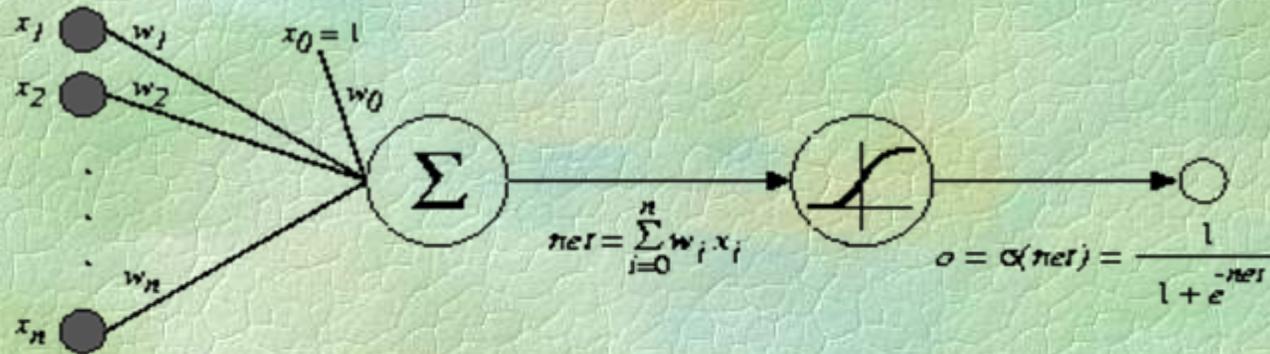
Вспомним, что т.к.  $o = -1$ , то  $\vec{w} \bullet \vec{x} \leq 0$ . Предположим, что  $x_i > 0$ , тогда увеличение весов сети приведёт к тому, что значение сети будет стремиться к значению функции.

В предположении линейной отделимости правило Видроу-Хоффа сходится к искомому решению за конечное число итераций.

# Многослойные нейронные сети



# Сигмоидная функция активации



$\sigma(x)$  is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Nice property:  $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

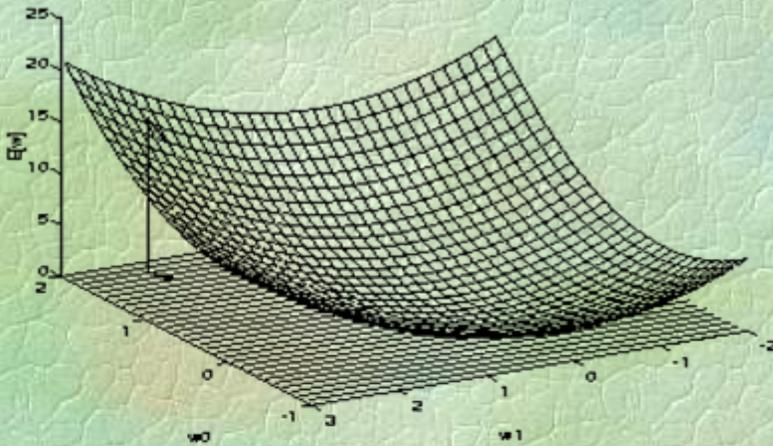
We can derive gradient decent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoid units →  
Backpropagation

# Метод градиентного спуска

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Функционал ошибки



Gradient

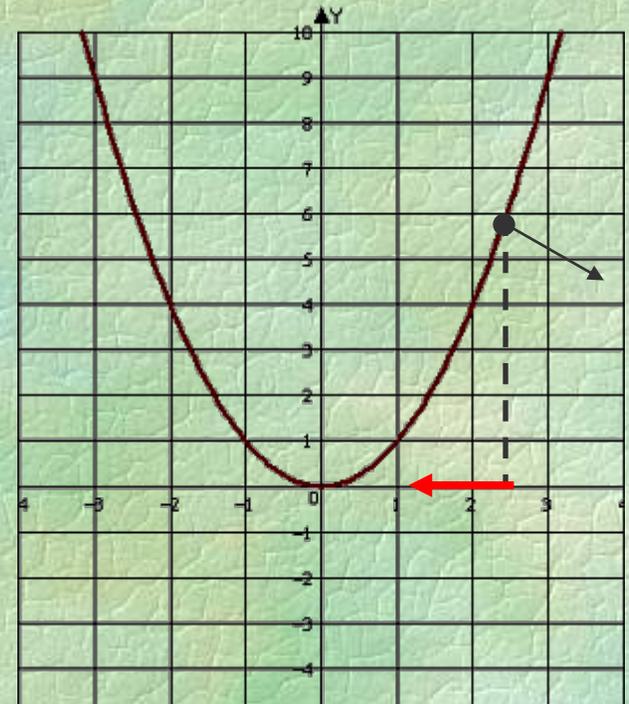
$$\nabla E[\vec{w}] \equiv \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$



# Метод градиентного спуска

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d)\end{aligned}$$

$$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d) (-x_{i,d})$$

# Дельта правило

Дельта правило позволяет избежать локально-оптимального решения.

При этом изменение вектора весов происходит после каждого акта применения сети к обучающему примеру.

**Функционал ошибки**

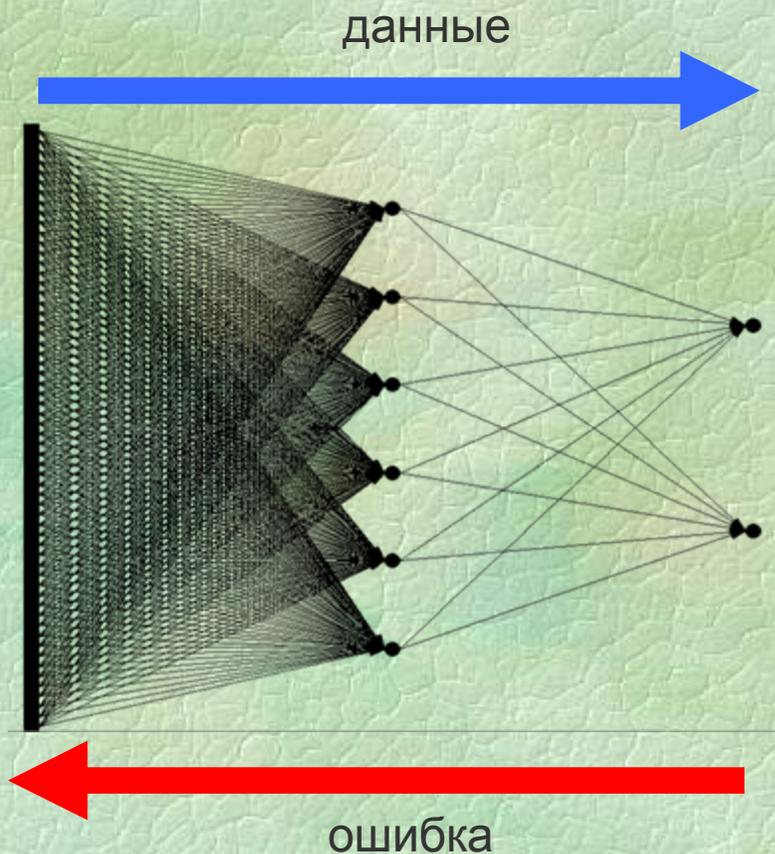
$$E_d[\vec{w}] \equiv \frac{1}{2}(t_d - o_d)^2$$

**Правило  
корректировки весов**

$$\Delta w_i = \eta(t - o)x_i$$

# Backpropagation - алгоритм обратного распространения ошибки

-исправление весов сети в соответствии с методом градиентного спуска

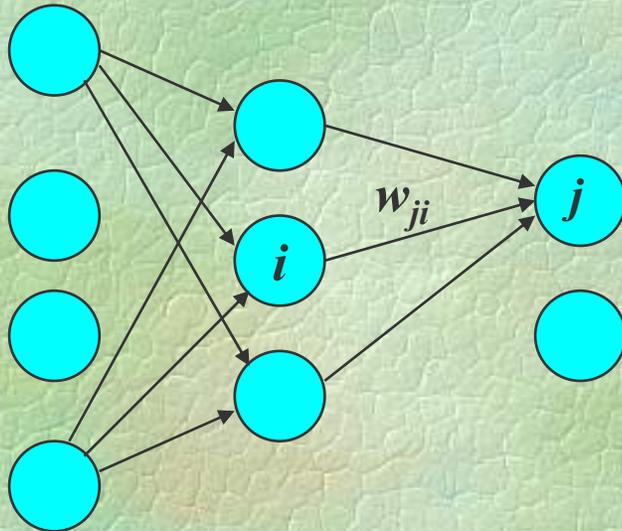


Определяет два потока в сети: прямой поток от входного слоя к выходному и обратный поток – от выходного слоя к входному.

Прямой поток продвигает входные векторы через сеть, в результате чего в выходном слое получаются выходные значения сети.

Обратный поток подобен прямому, но он продвигает назад по сети значения ошибок, в результате чего определяются величины, в соответствии с которыми следует корректировать весовые коэффициенты в процессе обучения. В обратном потоке значения проходят по взвешенным связям в направлении, обратном направлению прямого потока, т.е. в обратном потоке элементы некоторого слоя получают сигналы ошибок от каждого элемента следующего слоя.

# Backpropagation - алгоритм обратного распространения ошибки



Функционал ошибки

$$E_d(\vec{w}) = 1/2 \sum_{k \in \text{outputs}} (t_k - o_k)^2$$

$x_{ji}$  –  $i$ -тый вход на нейрон  $j$ ;

$w_{ji}$  – вес, приписанный связи между  $i$ -тым и  $j$ -тым нейронами;

$net_j = \sum_i w_{ji} x_{ji}$  – взвешенная сумма значений, поступивших на вход нейрону  $j$ ;

$o_j$  – значение выходного нейрона с номером  $j$ ;

$t_j$  – значение целевой функции;

$\text{outputs}$  – множество нейронов выходного слоя

# Backpropagation – правило корректировки весов выходного слоя

Напомним, что для минимизации функционала ошибки весовые коэффициенты корректируются в направлении, обратном вектору градиенту, а именно:

$$\Delta w_{ij} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

В терминах введённых обозначений, по правилу вычисления производной сложной функции:

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}.$$

В случае, когда  $j$  – нейрон выходного слоя:

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}, \quad (*)$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in \text{outputs}} (t_k - o_k)^2$$

# Backpropagation – правило корректировки весов выходного слоя

Заметим, что

$$\frac{\partial}{\partial o_j} (t_k - o_k)^2 = 0, \quad k \neq j.$$

Тогда

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = \frac{1}{2} 2(t_j - o_j) \frac{\partial (t_j - o_j)}{\partial o_j} = -(t_j - o_j).$$

Осталось вычислить ещё один множитель в уравнении (\*). Для этого вспомним, что

$$o_j = \sigma(\text{net}_j),$$

и в соответствии с формулой для производной сигмоидальной функции (см. слайд 10)

$$\frac{\partial o_j}{\partial \text{net}_j} = \frac{\partial \sigma(\text{net}_j)}{\partial \text{net}_j} = o_j(1 - o_j).$$

Окончательно, уравнение (\*) примет вид:

$$\frac{\partial E_d}{\partial \text{net}_j} = -(t_j - o_j) o_j (1 - o_j).$$

И правило исправления весов выходного слоя:

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} (t_j - o_j) o_j (1 - o_j) x_{ji}.$$

# Backpropagation – правило корректировки весов внутреннего слоя

Для вывода правила корректировки весов внутреннего слоя, введём величину:  $Downstream(j)$  – множество нейронов, на вход которым поступает выходное значение нейрона  $j$ .

$$\begin{aligned}\frac{\partial E_d}{\partial net_j} &= \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} (-\delta_k) \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} (-\delta_k) \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} = \\ &= \sum_{k \in Downstream(j)} (-\delta_k) w_{kj} \frac{\partial o_j}{\partial net_j} = \sum_{k \in Downstream(j)} (-\delta_k) w_{kj} o_j (1 - o_j).\end{aligned}$$

Обозначив

$$\delta_j = -\frac{\partial E_d}{\partial net_j},$$

получаем

$$\delta_j = o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}.$$

Правило исправления весов внутреннего слоя:

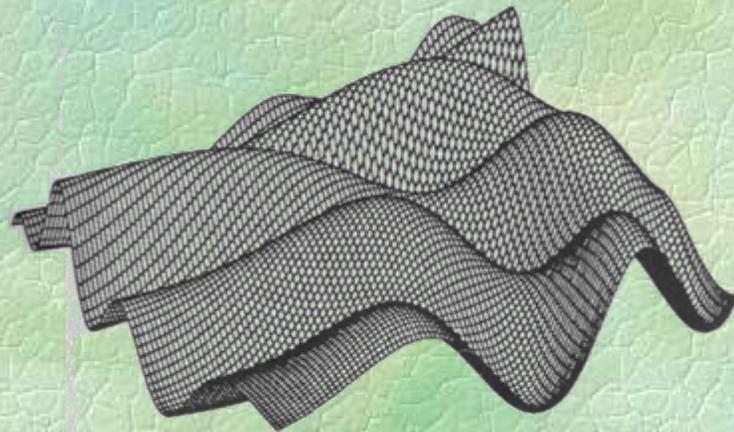
$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

Заметим, что правило исправления весов внутреннего слоя является частным случаем правила исправления весов внутреннего слоя, когда  $Downstream(j) = outputs$ .

# Backpropagation - алгоритм обратного распространения ошибки

- применяется для многослойных однопоточковых сетей
- функция активности - сигмоид
- осуществляет исправление весов сети в соответствии с методом градиентного спуска

В случае, если пространство ошибки имеет несколько локальных минимумов, есть опасность не найти глобального минимума



# Практические советы

## Overfitting in ANNs

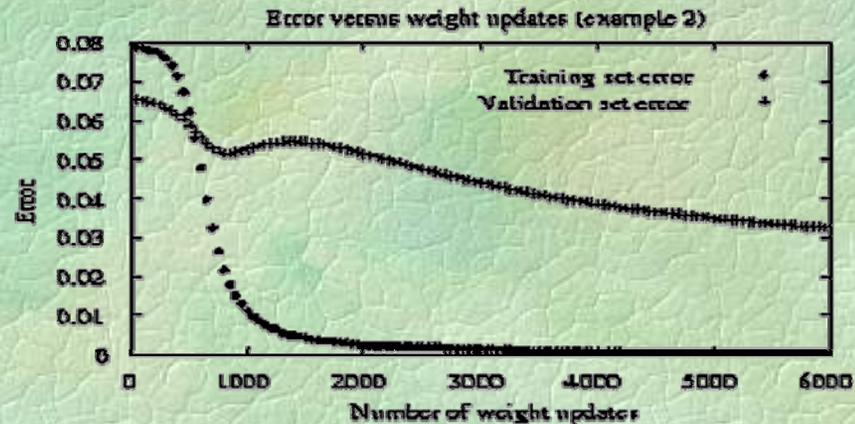
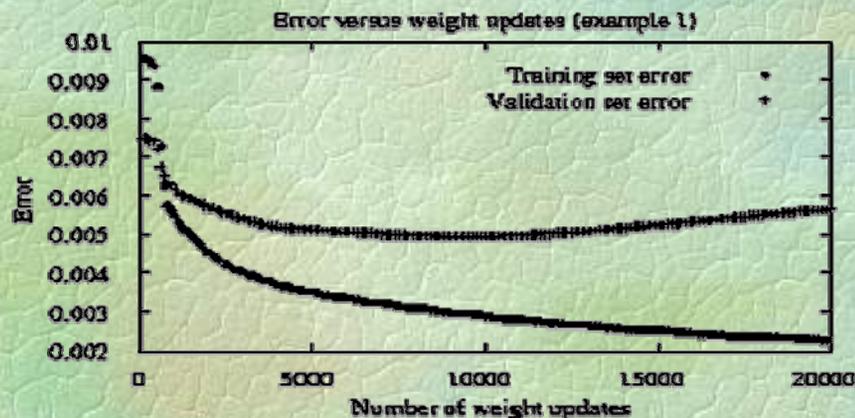
Баум и Хасслер, 1989

$$N > \frac{W}{\varepsilon}$$

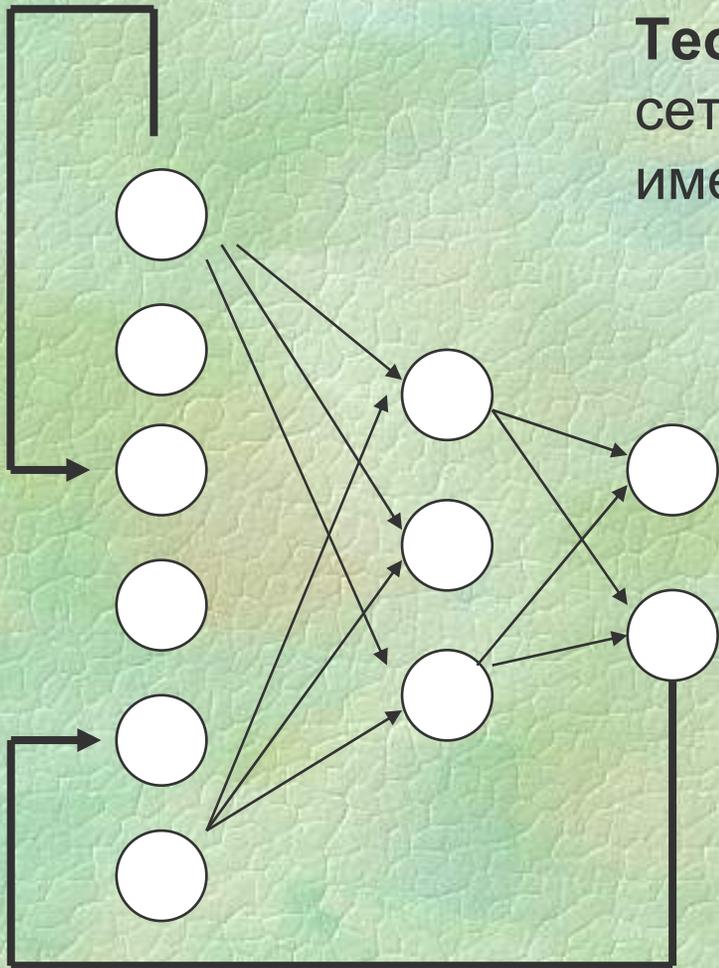
$N$  - размер обучающих данных

$W$  - число весовых коэффициентов

$\varepsilon$  - доля ошибок



# Рекуррентные нейронные сети



**Теорема.** Для любой рекуррентной сети существует сеть с прямой связью, имеющая идентичное поведение.

Обнаружение временных закономерностей

Выполнение задач, зависящих от последовательности состояний

Свойства краткосрочной памяти: может предсказать следующий элемент последовательности по текущему и предшествующему вводу.

# Применение в биоинформатике

предсказание структуры и функции белков, и их классификация (Nakata *et al.*, 1995, Wood *et al.*, 2004)

распознавание промоторов, кодирующих участков ДНК (Cai *et al.*, 1995, Reese, 2001);

обработка и анализ данных микроэрея:  
кластеризация профилей экспрессии с целью выявления биологически осмысленных групп генов (Sawa *et al.*, 2003), моделирование динамики экспрессии генов (Vohradsky, 2001), обнаружение временных закономерностей, предсказание состояния генной сети в следующий временной такт (Liang *et al.*, 1998), обнаружение классов в образцах различных тканей (Herrero *et al.*, 2001);

# Neural network model of gene expression

JLÍ VOHRADSKY *The FASEB Journal*. 2001;15:846-854

В статье искусственные нейронные сети используются как модель динамики экспрессии генов

В модели рассматриваются прямые и обратные связи между генами

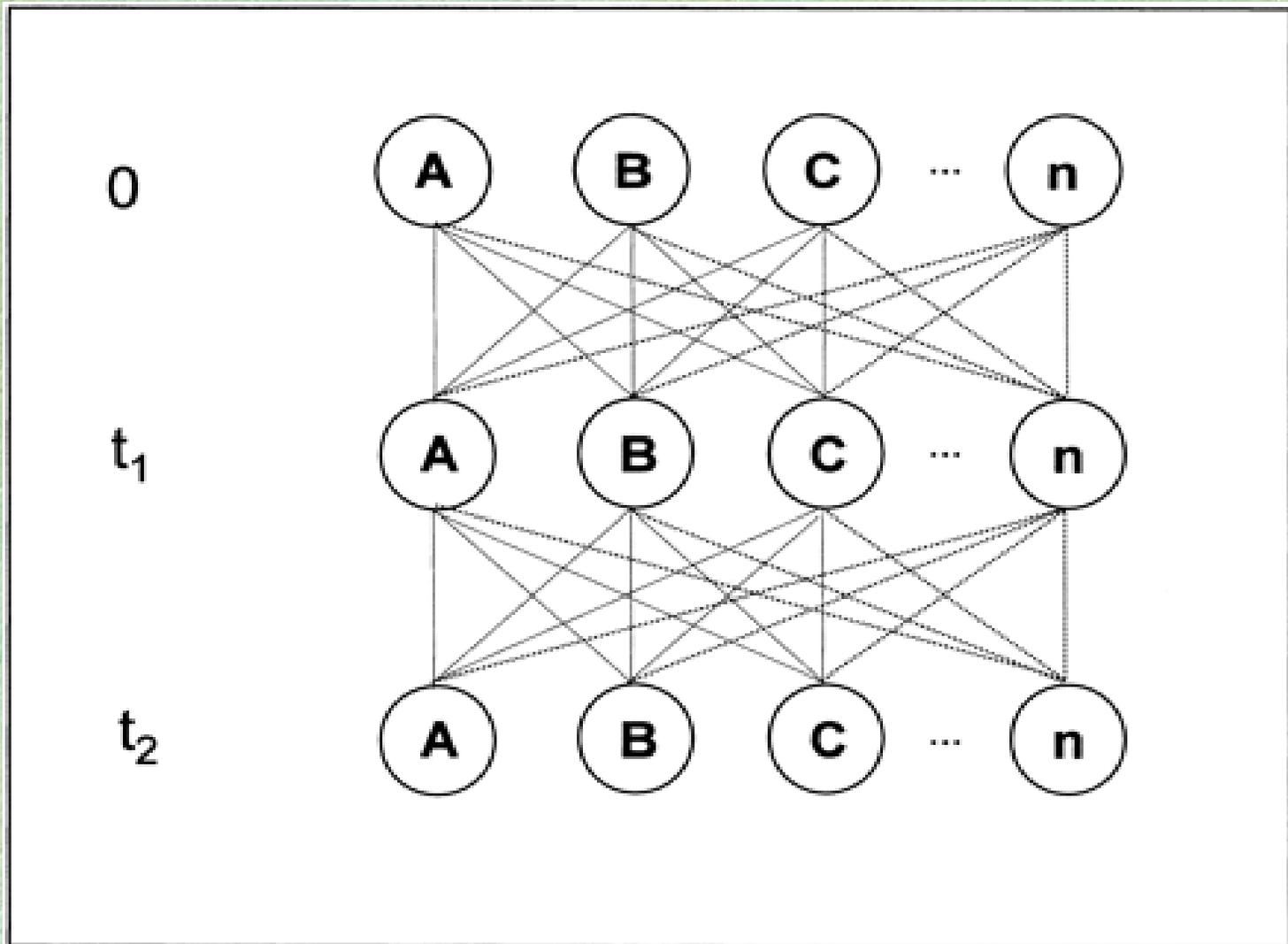
Весы сети отражают регуляторный эффект одного гена на экспрессию других Генов

Экспрессия генов моделируется одной сетью, а так же композицией сетей, обучаемых независимо для моделирования транскрипции и трансляции.

В статье обсуждаются методы вычисления параметров модели на основании экспериментальных данных.

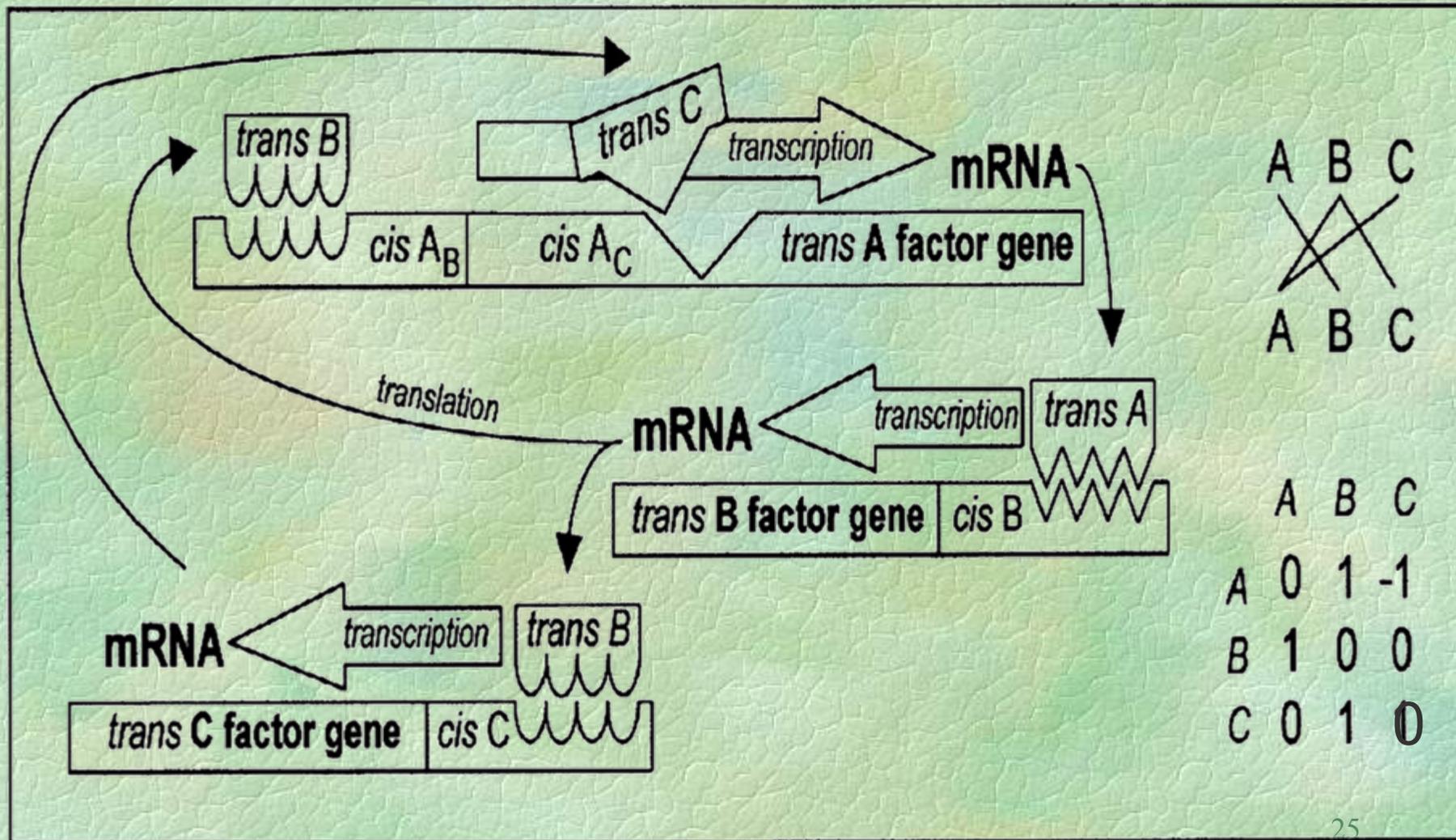
Результаты, полученные при помощи модели, соотносятся с экспериментальными наблюдениями.

# Формальное представление регуляции транскрипции нейронной сетью



# Пример булевых правил

$A=B$ ,  $A=\text{not } C$ ;  $B=A$ ;  $C=B$



# Моделирование генной сети рекуррентной нейронной сетью

Выход нейрона в момент времени  $t + \Delta t$  определяется через уровни экспрессии в момент времени  $t(y_j)$  и значениями весов  $w_{ij}$ , которые приписаны соответствующим связям. Регуляторный эффект генов, который испытывает на себе ген  $i$ :

$$g_i \approx \sum_j w_{ij} y_j.$$

**Регуляторный эффект:**

$$g_i = \left\{ 1 + \exp \left[ - \left( \sum_j w_{ij} y_j + b_i \right) \right] \right\}^{-1}$$

$y_i$  - уровень экспрессии гена в предыдущий временной такт

$w_{ij}$  - веса нейронной сети

$b_i$  - параметр задержки реакции

# Моделирование генной сети рекуррентной нейронной сетью

Изменение уровня экспрессии гена  $i$  выражается через регуляторный вклад других генов  $\rho$  и эффект деградации:

$$dz_i/dt = \rho_i - x_i$$

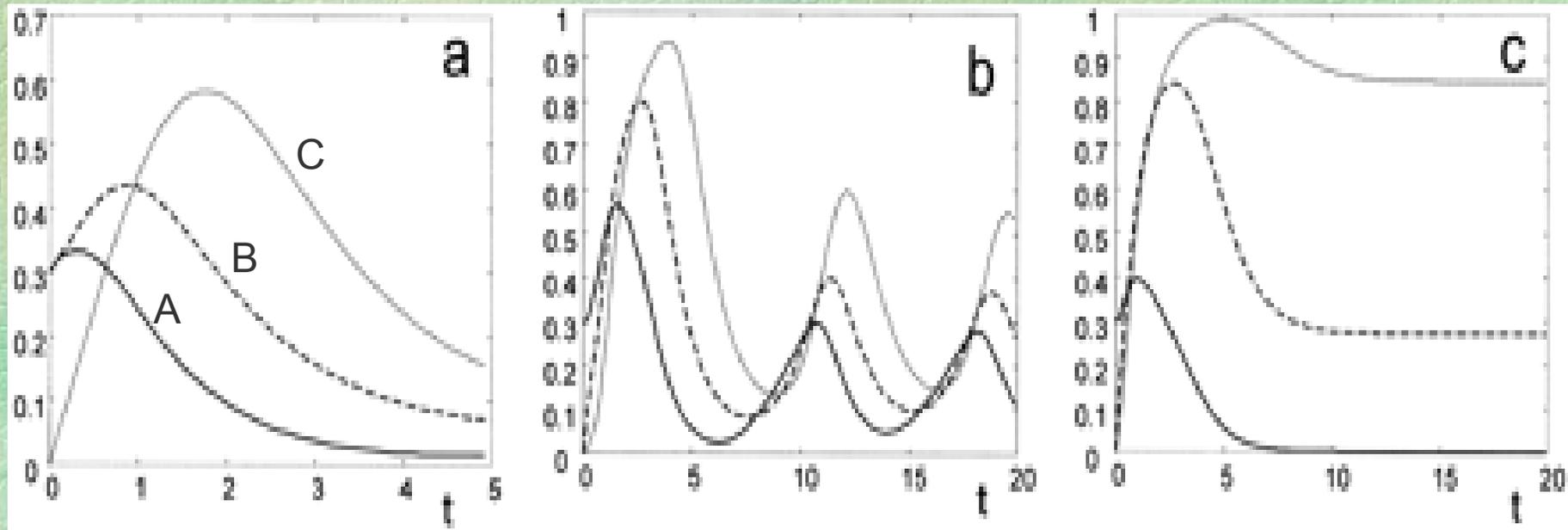
Уровень экспрессии гена  $i$ :  $\rho_i = k_{1i} g_i$

Эффект деградации:  $x_i = k_{2i} z_i$

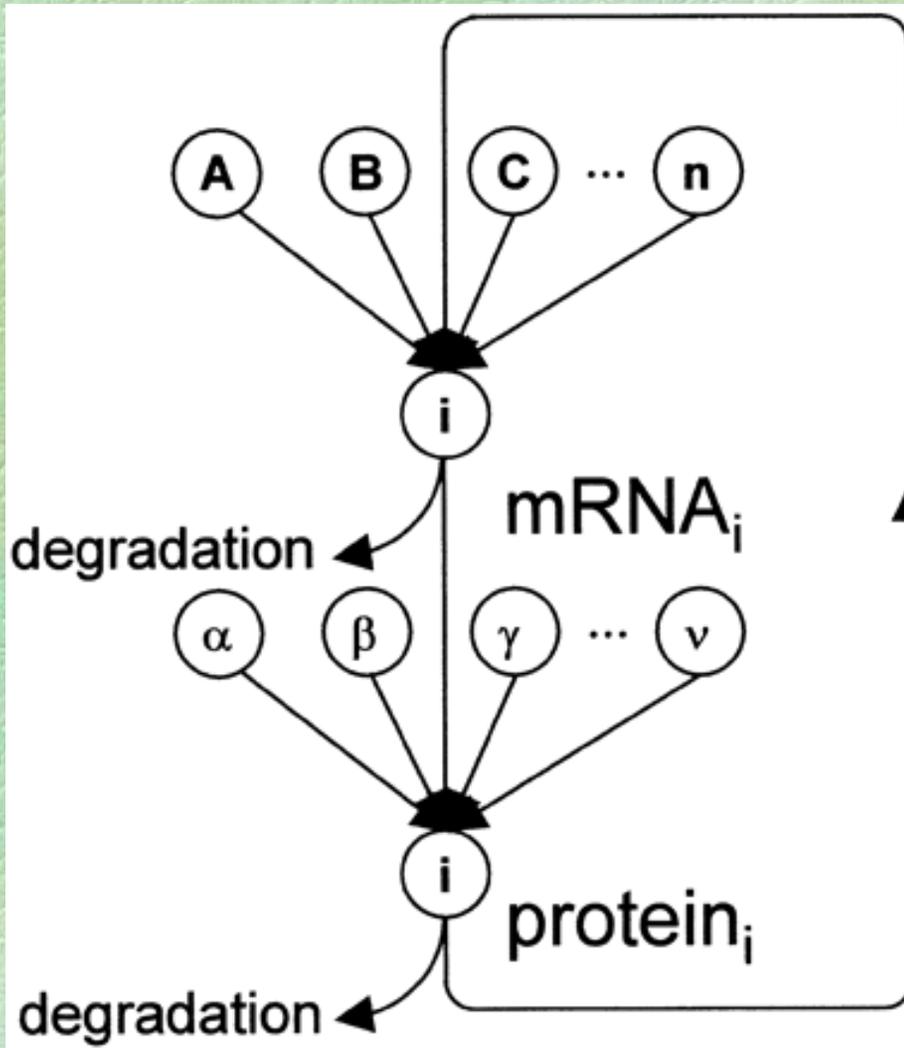
Таким образом, модель имеет вид:

$$\frac{dz_i}{dt} = k_{1i} \frac{1}{1 + \exp[-(\sum_j w_{ij} y_j + b_i)]} - k_{2i} z_i$$

# Паттерн экспрессии генов А, В и С для разных значений параметра задержки реакции



# Модель экспрессии генов, состоящая из двух компартов



Регуляторные белки  
A, B, C...n контролируют  
экспрессию гена  $i$

Трансляция мРНК  
контролируется набором  
факторов

# Извлечение архитектуры нейронных сетей из экспериментальных данных

Максимально задействовать априорную информацию о модели: предполагаемые зависимости, гипотезы относительно параметров задержки

Достаточное количество экспериментальных данных

Использование процедур оптимизации, чтобы избежать сходимости к локальному минимуму

Необходима предобработка экспериментальных данных, например, предварительная кластеризация профилей экспрессии

# Bayesian Learning



# Сведения из теории вероятностей

Рассмотрим случайный (недетерминированный) эксперимент, результат которого нельзя предсказать заранее.

Пространством элементарных исходов  $\Omega = \{\omega_1, \omega_2, \dots\}$  называется множество всех возможных результатов случайного эксперимента. Элементы этого множества  $\omega_i$  называют элементарными исходами.

Событиями называют подмножества множества  $\Omega$ . Говорят, что в результате эксперимента произошло событие  $A \subseteq \Omega$ , если в эксперименте произошел один из элементарных исходов, входящих в множество  $A$ .

Пусть пространство элементарных исходов состоит из конечного числа  $N$  элементов:  $\Omega = \{\omega_1, \omega_2, \dots, \omega_N\}$ , причем будем считать, что элементарные исходы равновозможны. Тогда вероятность любого из них принимается равной  $1/N$ . Если событие  $A = \{\omega_{i_1}, \omega_{i_2}, \dots, \omega_{i_k}\}$  состоит из  $k$  элементарных исходов, то вероятность этого события вычисляется как:

$$P(A) = \frac{|A|}{|\Omega|} = \frac{k}{N},$$

где  $|A|$  есть число элементов конечного множества  $A$ .

# Сведения из теории вероятностей

Определим операции объединения, пересечения и дополнения над событиями.

Пусть  $A$  и  $B$  - события.

1. Объединением  $A \cup B$  событий  $A$  и  $B$  называется событие, состоящее в том, что произошло либо  $A$ , либо  $B$ , либо оба события одновременно, т.е.  $A \cup B$  есть множество, содержащее как элементарные исходы, входящие в  $A$ , так и элементарные исходы, входящие в  $B$ .

2. Пересечением  $A \cap B$  событий  $A$  и  $B$  называется множество, содержащее элементарные исходы, входящие одновременно в  $A$  и в  $B$ .

3. Дополнением  $A \setminus B$  события  $B$  до  $A$  называется множество, содержащее элементарные исходы, входящие в  $A$ , но не входящие в  $B$ .

# Сведения из теории вероятностей

$A, B$  – некоторые события

Вероятность пересечения этих событий:

$$P(A \wedge B) = P(A/B)P(B) = P(B/A)P(A)$$

Вероятность объединения:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Формула условной вероятности.

Условной вероятностью события  $A$ , при условии, что произошло событие  $B$ , называется число:

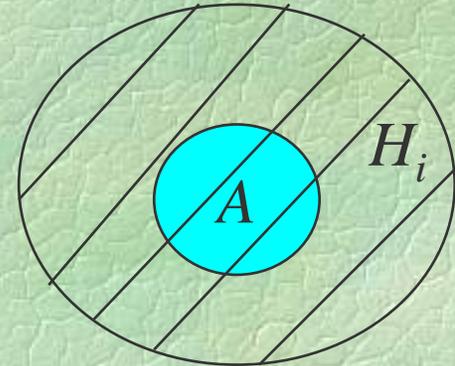
$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

# Сведения из теории вероятностей

Говорят, что события  $H_1, H_2, \dots, H_n$  образуют полную группу событий относительно события  $A$ , если:

$$1) H_i \cap H_j = \emptyset, \forall i \neq j.$$

$$2) \bigcup_{i=1}^n H_i \supseteq A.$$



События  $H_1, H_2, \dots, H_n$  часто называют гипотезами.

Формула полной вероятности.

Вероятность события  $A$  может быть вычислена по следующей формуле:

$$P(A) = \sum_{i=1}^n P(A|H_i)P(H_i).$$

# Теорема Байеса. MAP-гипотеза.

Пусть  $H_1, H_2, \dots, H_n$  образуют полную группу событий относительно события  $A$ , причем  $P(A) > 0$ . Тогда условная вероятность того, что имело место событие  $H_k, k = 1, \dots, n$ , если в результате эксперимента наблюдалось событие  $A$ , может быть вычислена по формуле:

$$P(H_k | A) = \frac{P(H_k)P(A|H_k)}{\sum_{i=1}^n P(H_i)P(A|H_i)}$$

Формула Байеса предназначена для вычисления вероятности апостериорных обстоятельств  $P(H_k | A)$ , т.е. обстоятельств, возникших после того (*a posteriori*), как произошло событие  $A$ .

В отличие от апостериорной априорная вероятность (*a priori*, до того как) определяется накопленным опытом. Эта величина обычно используется для предсказания исходов до их реального наступления.

**Определение.** MAP-гипотезой (*maximum a posteriori*) называется такая гипотеза:

$$h_{MAP} = \arg \max_{h \in H} P(h / D) = \arg \max_{h \in H} \frac{P(D / h)P(h)}{P(D)} = \arg \max_{h \in H} P(D / h)P(h).$$

В условиях, когда недостаточно оснований, для того, чтобы предпочесть одну гипотезу другой, полагают:

$$P(h_i) = P(h_j),$$

всем гипотезам  $h \in H$  приписывают одинаковую априорную вероятность, тогда;

$$h_{MAP} = h_{ML} = \arg \max_{h \in H} P(D / h)$$

# Пример обнаружения МАР-гипотезы

Пусть имеется две гипотезы:

$h_1 : cancer$  - у пациента имеется определённая форма рака;

$h_2 : \neg cancer$  - пациент здоров

В лаборатории при обработке анализа получают два вида результата, маркеры “+” и “-”. Причём верный результат в лаборатории получают с некоторой вероятностью, а именно:

$P(+ / cancer) = 0.98$ , тогда  $P(- / cancer) = 0.02$

$P(+ / \neg cancer) = 0.03$ ,  $P(- / \neg cancer) = 0.97$

Статистика по населению утверждает, что рак проявляется с вероятностью:

$P(cancer) = 0.008$ , и, соответственно,  $P(\neg cancer) = 0.992$ .

Требуется установить, болен ли раком пациент, у которого результатом лабораторного теста является положительный маркер.

В данном случае априорные вероятности гипотез (о присутствии/отсутствии рака) не равны, поэтому воспользуемся формулой для обнаружения гипотезы, обладающей максимальной апостериорной вероятностью.

Вычислим необходимые произведения:

$P(+ / cancer)P(cancer) = 0.98 \cdot 0.008 = 0.0078$ ,

$P(+ / \neg cancer)P(\neg cancer) = 0.03 \cdot 0.992 = 0.0298$ .

Итак,  $h_{MAP} : \neg cancer$

# NAÏVE BAYES CLASSIFIER

Наивный классификатор Байеса применяется к задачам, в которых обучающие примеры  $x \in X$  представлены конъюнкцией значений признаков  $\langle a_1, \dots, a_N \rangle$ . Целевая функция приписывает некоторый класс  $v_j \in V$  обучающему примеру  $x \in X$ .

Найдём класс, обладающий максимальной апостериорной вероятностью для некоторого поднабора значений признаков  $\langle a_1, \dots, a_n \rangle$ .

$$v_{MAP} = \arg \max_{v_j \in V} P(v_j / a_1, \dots, a_n) = \arg \max_{v_j \in V} \frac{P(a_1, \dots, a_n / v_j) P(v_j)}{P(a_1, \dots, a_n)} = \arg \max_{v_j \in V} P(a_1, \dots, a_n / v_j) P(v_j).$$

В данном произведении, множитель  $P(v_j)$  легко оценивается из таблицы обучающих данных как часть примеров, принадлежащих классу  $v_j$ . Для оценки оставшегося множителя в произведении требуется большой объем обучающих данных, поэтому наивный классификатор Байеса основан на упрощающем предположении о независимости признаков  $a_1, \dots, a_N$ . Это предположение позволяет переписать искомую вероятность в следующем виде:

$$P(a_1, \dots, a_n / v_j) = \prod_i P(a_i / v_j).$$

Тогда

$$v_{MAP} = v_{NB} = \arg \max_{v_j \in V} \prod_i P(v_j) P(a_i / v_j)$$

# Обучающие данные

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

# Пример применения классификатора

〈 *Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong, PlayTennis = ?* 〉

Вычисляем

$$v_{NB} = \arg \max_{v_j \in \{yes, no\}} \prod_i P(v_j) P(a_i / v_j) =$$

$$\arg \max_{v_j \in V} P(v_j) P(Outl = Sunny / v_j) P(Temp = Cool / v_j) P(Hum = High / v_j) P(Wind = Strong / v_j)$$

В данной записи содержится 10 вероятностей, которые необходимо оценить из таблицы данных.

Для значений целевой функции:

$$P(PlayTennis = yes) = 9/14 = 0.64$$

$$P(PlayTennis = no) = 5/14 = 0.36$$

Вычислим условные вероятности для фиксированных значений признаков:

$$P(Wind = Strong / PlayTennis = yes) = 3/9 = 0.33$$

$$P(Wind = Strong / PlayTennis = no) = 3/5 = 0.60$$

Вероятности для других признаков вычисляются аналогичным образом.

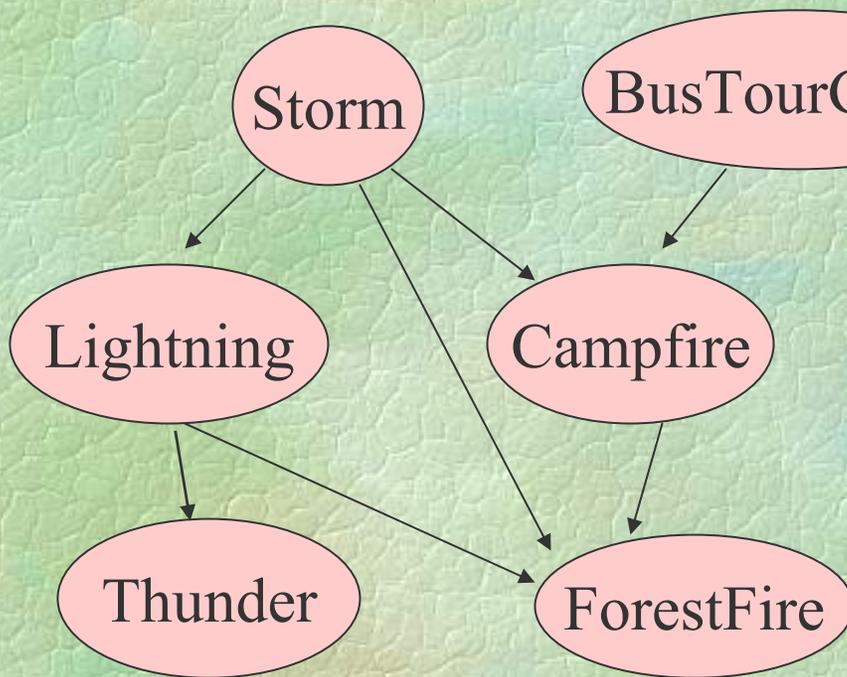
Окончательно имеем;

$$P(yes) P(Sunny / yes) P(Cool / yes) P(High / yes) P(Strong / yes) = 0.0053$$

$$P(no) P(Sunny / no) P(Cool / no) P(High / no) P(Strong / no) = 0.0206 .$$

Итак, классификатор Байеса склоняется к значению целевой концепции *PlayTennis = no*.

# Байесовские сети



	S.B	S.~B	~S.B	~S.~B
C	0.4	0.1	0.8	0.2
~C	0.6	0.9	0.2	0.8

$P(\text{Campfire} = \text{true} \mid \text{Storm} = \text{false}, \text{BusTourGroup} = \text{false}) = 0.2$

Выполнено условие независимости признака от НЕ предшественников:

$$(\forall x_i, y_j, z_k) P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k)$$

Совместное распределение значений  $\langle y_1, \dots, y_n \rangle$  признаков  $\langle Y_1, \dots, Y_n \rangle$ :

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i \mid \text{Parents}(Y_i)),$$

$\text{Parents}(Y_i)$  – непосредственные предшественники признака  $Y_i$ .

# Modeling splice sites with Bayes networks

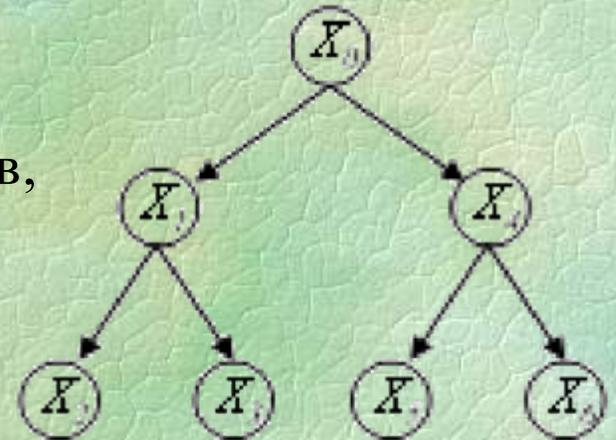
D. Cai, A. Delcher, B. Kao and S. Kasif

Каждая позиция нуклеотидной последовательности длины  $N$  может быть представлена дискретной случайной величиной  $X_i$ , признаком, принимающим значения из множества  $\{a, c, g, t\}$ .

Модель Байеса обобщает Марковскую модель первого порядка, допуская любые попарные зависимости нуклеотидных позиций.

В Байесовских сетях каждая позиция имеет только одного родителя, тогда как сама может быть родителем для нескольких позиций.

Совместное распределение значений признаков, изображенных на рисунке, вычисляется следующим образом:



$$P(X_0, X_1, X_2, X_3, X_4, X_5, X_6) =$$

$$P(X_0)P(X_1|X_0)P(X_2|X_0)P(X_3|X_1)P(X_4|X_1)P(X_5|X_2)P(X_6|X_2)$$

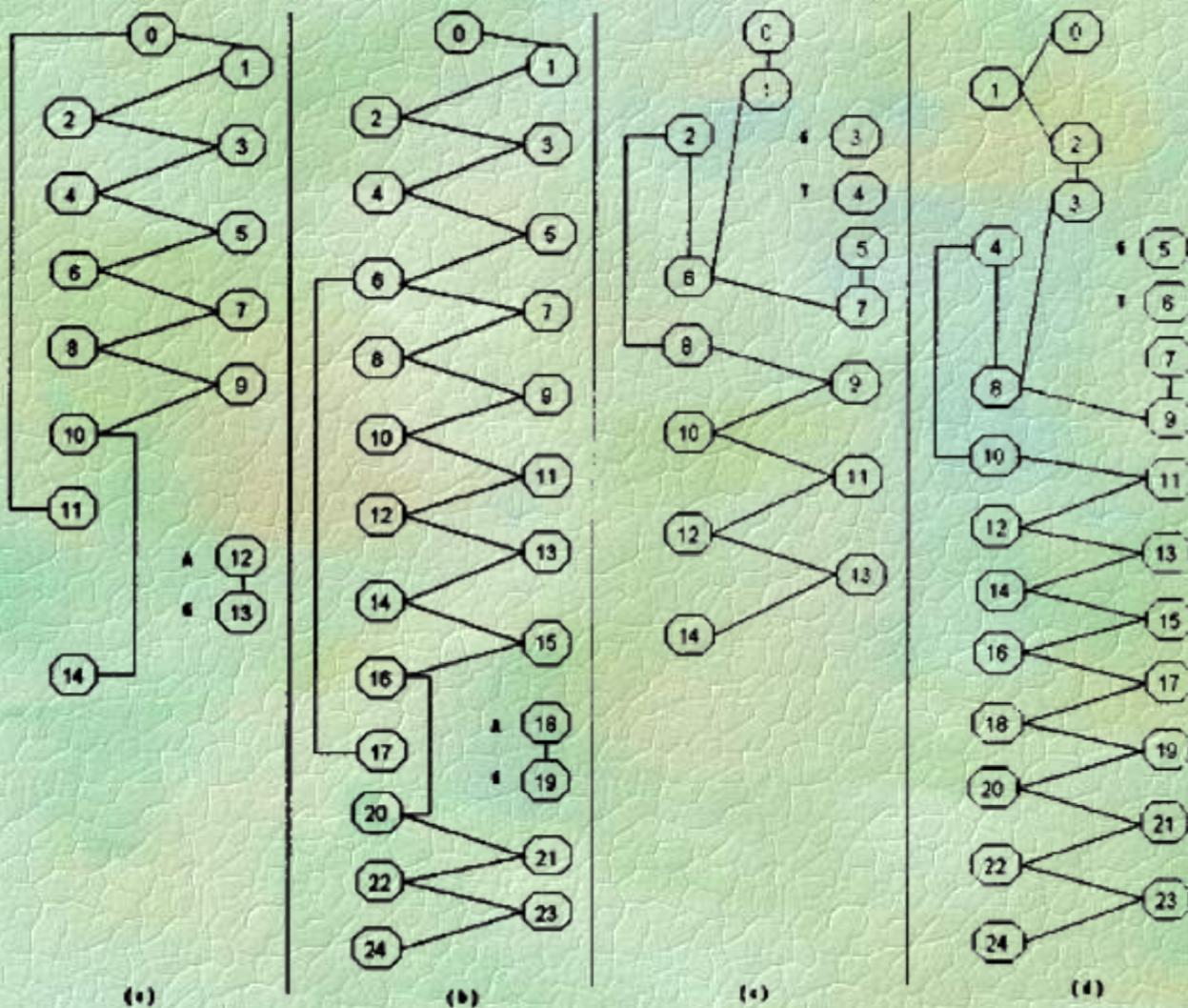
# Извлечение байесовской модели из данных

Для того, чтобы восстановить вероятностную модель сети из данных, рассматривают полный граф на  $N$  узлах, в котором узел  $X$  соответствует некоторой позиции нуклеотидной последовательности и все узлы графа соединены ребрами. Каждому ребру приписывается вес, равный количеству информации, содержащейся в одной позиции  $X$  относительно другой  $Y$  (Cai *et al.*, 2000).

$$I(X, Y) = I(Y, X) = \sum_{i,j} P(X = x_i, Y = y_j) \log_2 \frac{P(X = x_i, Y = y_j)}{P(X = x_i)P(Y = y_j)}$$

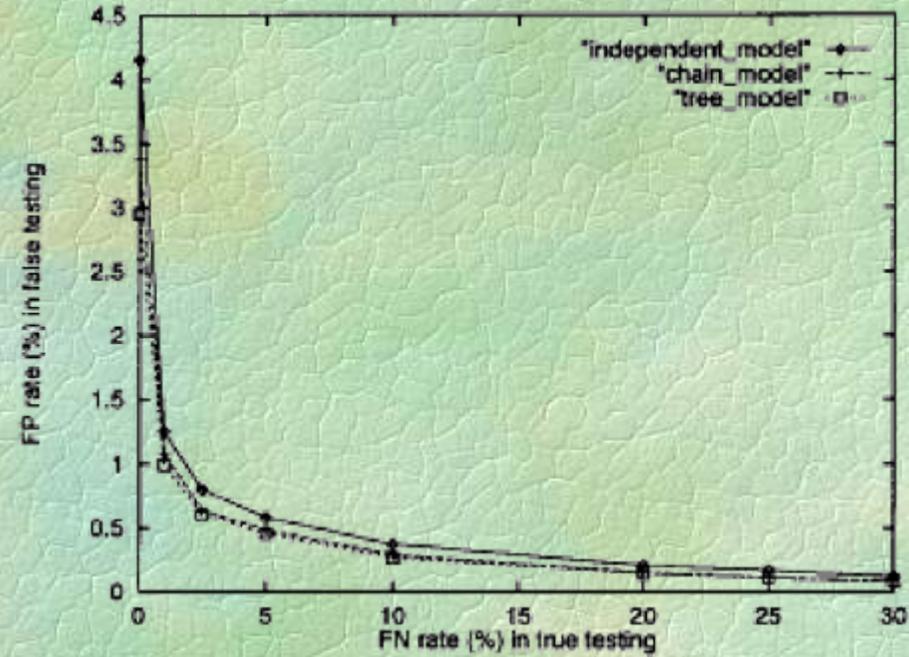
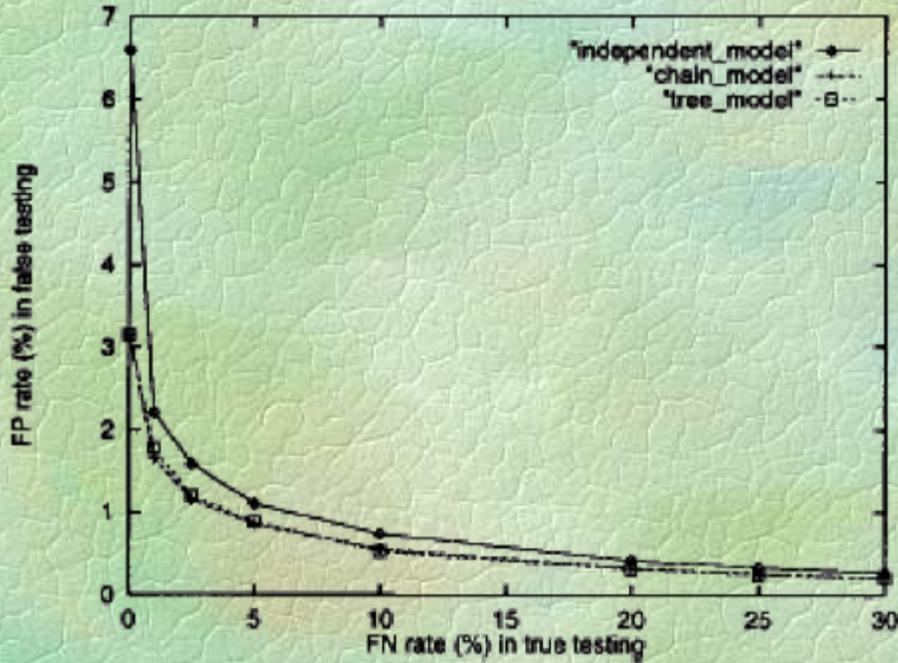
Задача состоит в нахождении такого ациклического подграфа, содержащего все вершины, у которого суммарный вес ребер будет максимальным (Wu, 2004).

# Байесовские модели донорных и акцепторных сайтов сплайсинга



Модели акцепторных (1 и 2) и донорных (3 и 4) сайтов сплайсинга, полученных для скользящего окна размером 15 и 25 нк

# Сравнение трех моделей сайтов

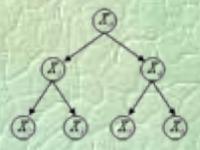


Class	0	1	2	3	4	5	6	7	8	9
0	0.99	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
1	0.01	0.99	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
2	0.01	0.01	0.99	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.01	0.01	0.01	0.99	0.01	0.01	0.01	0.01	0.01	0.01
4	0.01	0.01	0.01	0.01	0.99	0.01	0.01	0.01	0.01	0.01
5	0.01	0.01	0.01	0.01	0.01	0.99	0.01	0.01	0.01	0.01
6	0.01	0.01	0.01	0.01	0.01	0.01	0.99	0.01	0.01	0.01
7	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.99	0.01	0.01
8	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.99	0.01
9	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.99

Модель, основанная на предположении о независимости нк позиций (весовая матрица)



Марковская модель первого порядка (цепь)



Вероятностное дерево Байса

# Применение Байесовских сетей в биоинформатике

- моделирования сайтов сплайсинга последовательностей ДНК
- предсказания вторичной структуры белка
- симуляции генной сети и отслеживания её динамики

Сети Байеса способны поддерживать данные с шумами, оперировать неполными данными, с пробелами. Как и в случае решающих деревьев, и индуктивно-логических правил, такой способ интерпретации фактов как граф позволяет наиболее естественно проверить гипотезу типа “если, то”. В отличие от решающих деревьев, сети Байеса менее склонны к переобучению.