МАШИННОЕ ОБУЧЕНИЕ И АНАЛИЗ ДАННЫХ

(Machine Learning and Data Mining)

Н. Ю. Золотых

http://www.uic.unn.ru/~zny/ml

Лекция 10

Нейронные сети

10.1. Нейронные сети

10.1. Нейронные сети

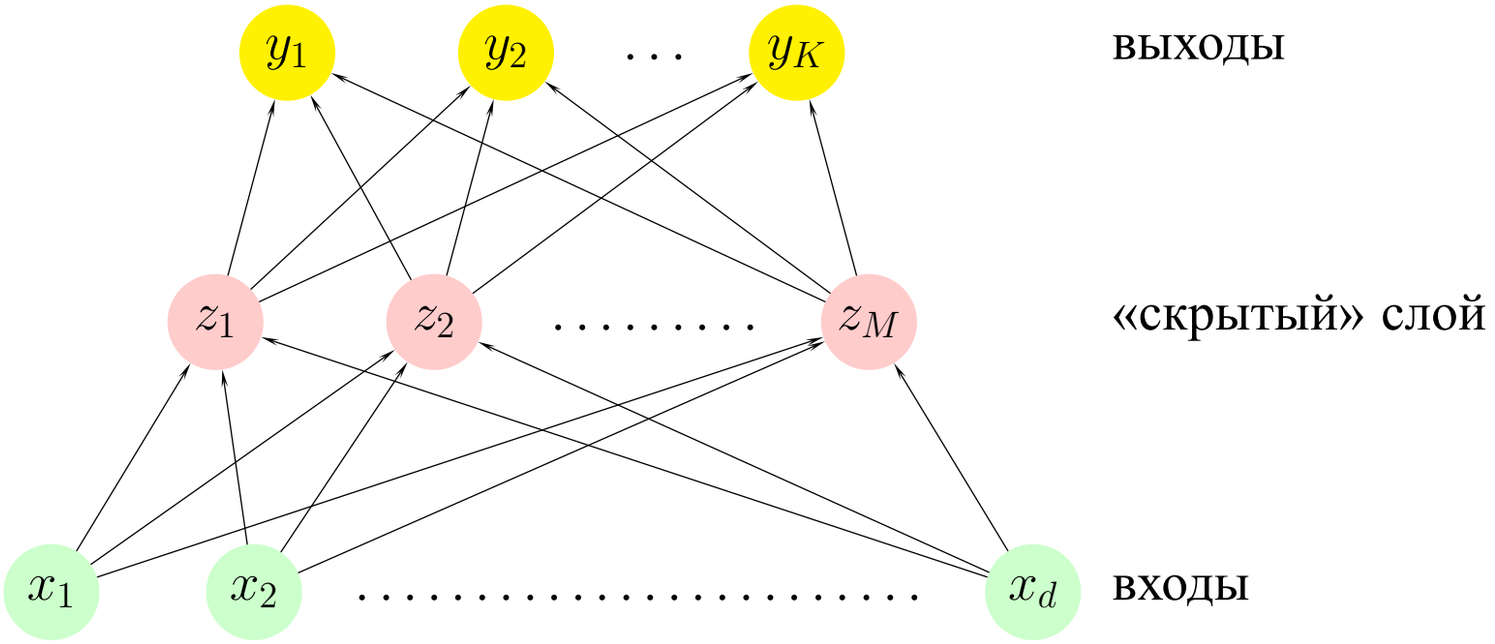
Нейронная сеть — (очень простая) модель мозга, т. е. множества взаимодействующих между собой нейронов.

10.1. Нейронные сети

Нейронная сеть — (очень простая) модель мозга, т. е. множества взаимодействующих между собой нейронов.

Нейронная сеть (или *искусственная нейронная сеть* — ориентированный граф, вершинам которого соответствуют функции (*функции активации*), а каждой входящей в вершину дуге (*синапсу*) — ее аргумент. Вход нейрона — дендрит. Выход — аксон

Пример «трехслойной» нейронной сети:



С математической точки зрения нейронная сеть графически задает суперпозицию функций активации.

С математической точки зрения нейронная сеть графически задает суперпозицию функций активации.
Небольшая классификация нейронных сетей:

- сети прямого распространения (feed-forward NN) — отсутствуют орциклы;
- рекуррентные нейронные сети, или сети с обратной связью (recurrent NN) — присутствуют орциклы.

Рекуррентные нейронные сети используют, например, для предсказания временных рядов.

С математической точки зрения нейронная сеть графически задает суперпозицию функций активации.

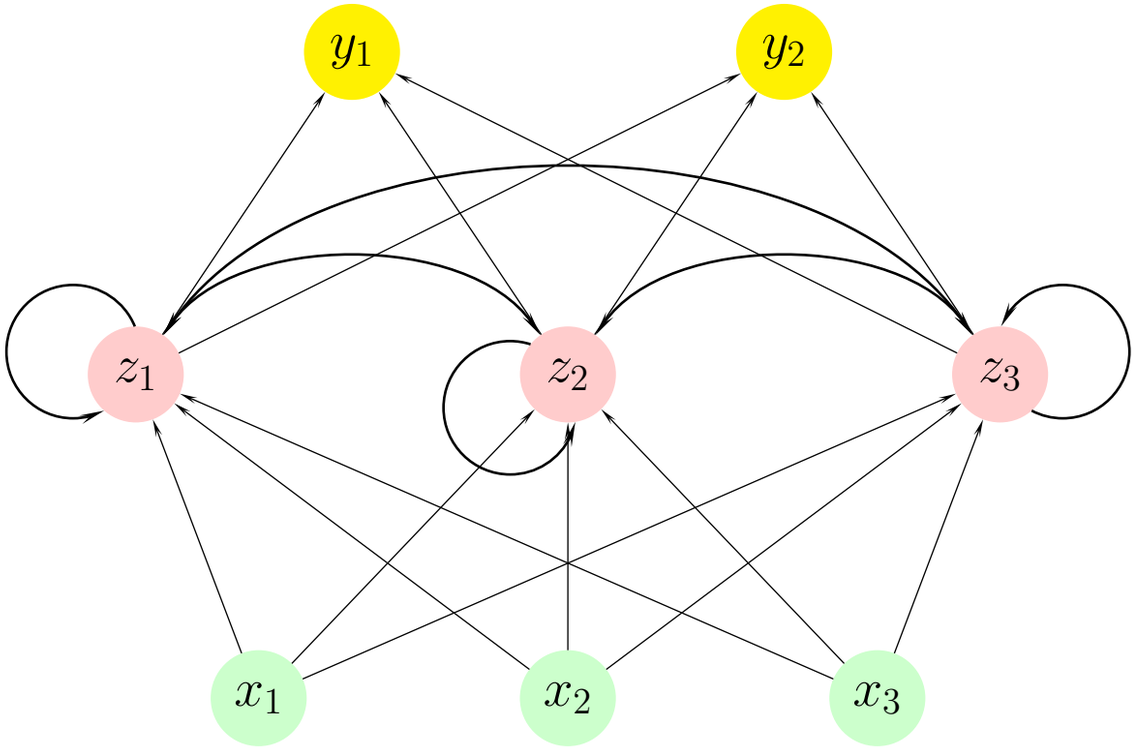
Небольшая классификация нейронных сетей:

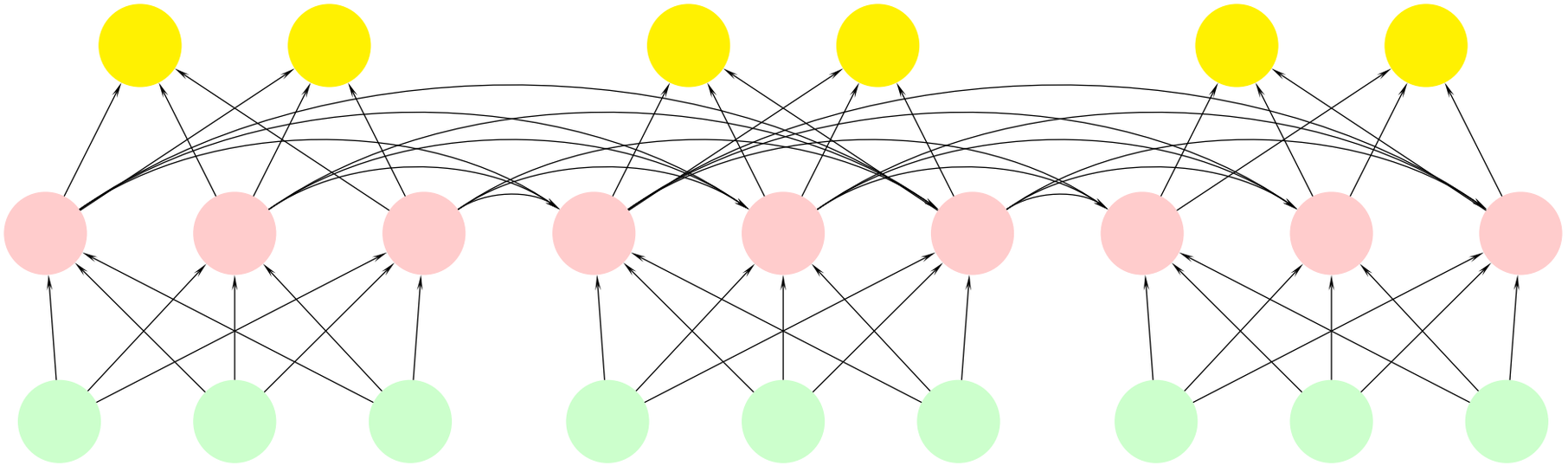
- сети прямого распространения (feed-forward NN) — отсутствуют орциклы;
- рекуррентные нейронные сети, или сети с обратной связью (recurrent NN) — присутствуют орциклы.

Рекуррентные нейронные сети используют, например, для предсказания временных рядов.

Если узлы сети можно разбить на группы (слои), которые удается пронумеровать так, что дуги будут вести только от вершин из i-го слоя в вершины $(i+1)$-го слоя, то нейронная сеть называется *многоуровневой* (или *многослойным персептроном*)).

Пример рекуррентной сети





$t-1$ t $t+1$

В качестве функций активации обычно берут пороговые и близкие к ним функции:

Пороговая функция (в чистом виде обычно не используется, так как разрывна):

$$g(x_1, x_2, \ldots, x_q) = \begin{cases} 1, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q \geq 0, \\ 0, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q < 0, \end{cases}$$

В качестве функций активации обычно берут пороговые и близкие к ним функции:

Пороговая функция (в чистом виде обычно не используется, так как разрывна):

$$g(x_1, x_2, \ldots, x_q) = \begin{cases} 1, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q \geq 0, \\ 0, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q < 0, \end{cases}$$

Сигмоидальная (*логит* или *логистическая*) функция (наиболее популярная):

$$g(x_1, x_2, \ldots, x_q) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)}}$$

В качестве функций активации обычно берут пороговые и близкие к ним функции:

Пороговая функция (в чистом виде обычно не используется, так как разрывна):

$$g(x_1, x_2, \ldots, x_q) = \begin{cases} 1, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q \geq 0, \\ 0, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q < 0, \end{cases}$$

Сигмоидальная (*логит* или *логистическая*) функция (наиболее популярная):

$$g(x_1, x_2, \ldots, x_q) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)}}$$

Арктангенс:

$$g(x_1, x_2, \ldots, x_q) = \frac{1}{2} + \frac{1}{\pi} \operatorname{arctg}(\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)$$

В качестве функций активации обычно берут пороговые и близкие к ним функции:

Пороговая функция (в чистом виде обычно не используется, так как разрывна):

$$g(x_1, x_2, \ldots, x_q) = \begin{cases} 1, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q \geq 0, \\ 0, & \text{если } \beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q < 0, \end{cases}$$

Сигмоидальная (*логит* или *логистическая*) функция (наиболее популярная):

$$g(x_1, x_2, \ldots, x_q) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)}}$$

Арктангенс:

$$g(x_1, x_2, \ldots, x_q) = \frac{1}{2} + \frac{1}{\pi} \operatorname{arctg}(\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)$$

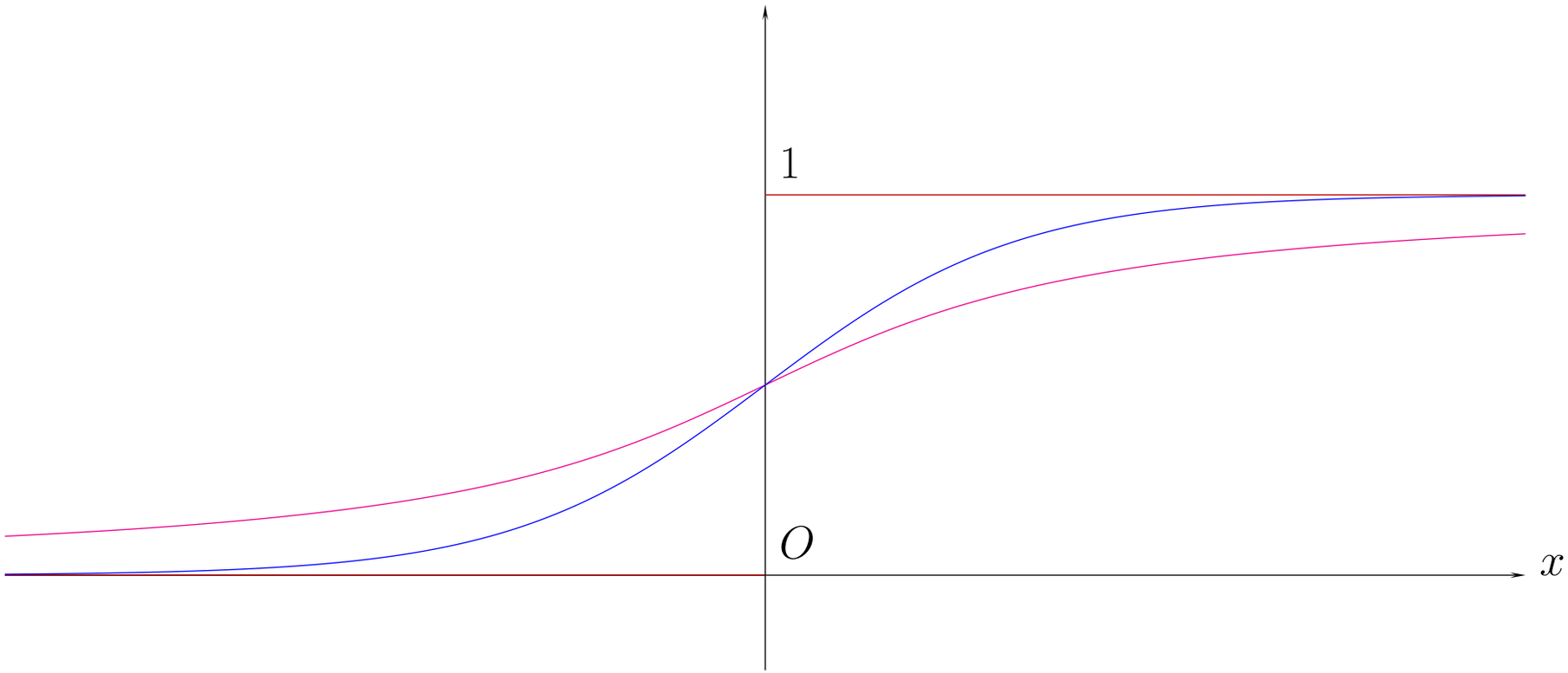
В специальном виде нейронных сетей — радиальных нейронных сетях (radial basis function network — RBF network) — используют радиальную функцию (Гауссиан)

$$g(x) = e^{-\beta \|x - c\|}.$$

Элементарная пороговая функция $y = I(x \geq 0) = \begin{cases} 1, & \text{если } x \geq 0, \\ 0, & \text{если } x < 0, \end{cases}$

Элементарный сигмоид (логит или логистическая функция) $y = \sigma(x) = \dfrac{1}{1 + e^{-x}}$

Арктангенс: $y = \dfrac{1}{2} + \dfrac{1}{\pi} \operatorname{arctg} x$



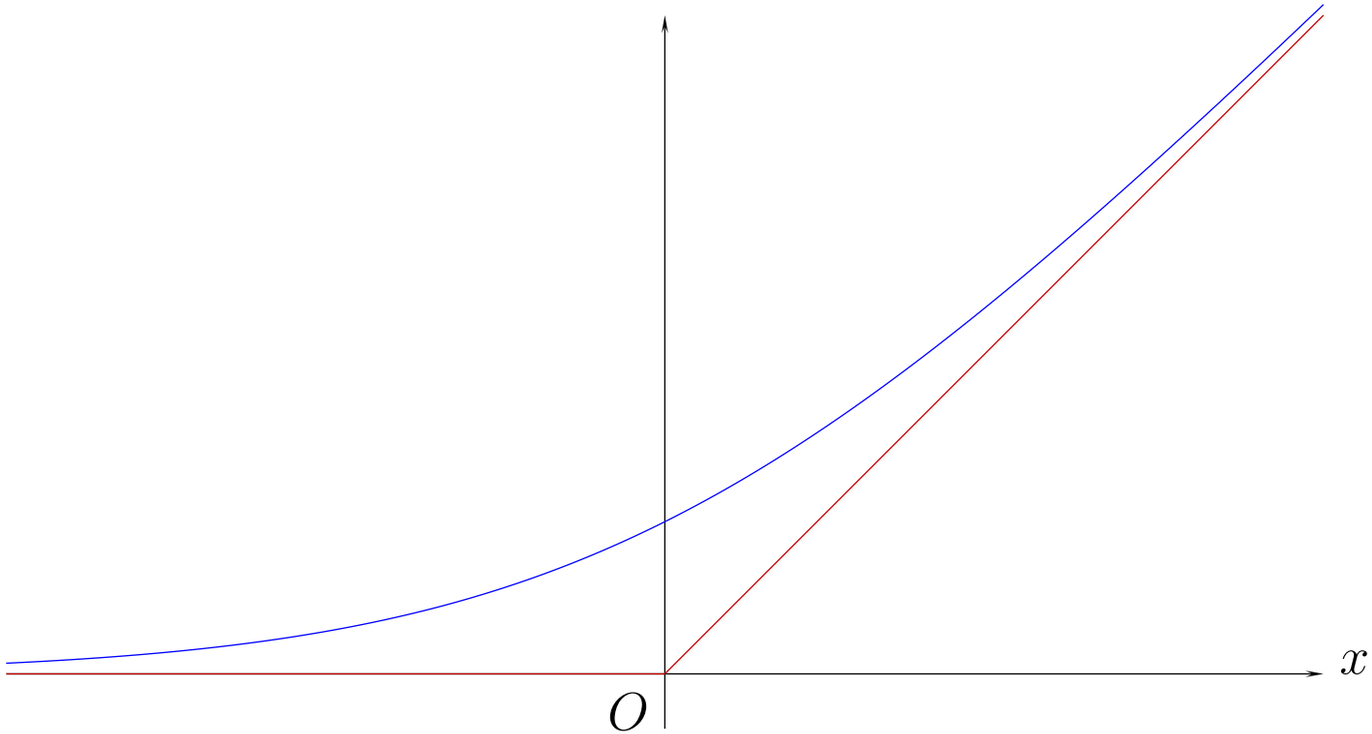
Используют и другие функции, например, в настоящее время самая популярная – *положительная срезка линейной функции* (linear rectifier):

$$g(x_1, x_2, \ldots, x_q) = (\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)_+$$

где $(x)_+ = \max\{0, x\}$, или ее сглаженный вариант *softplus*:

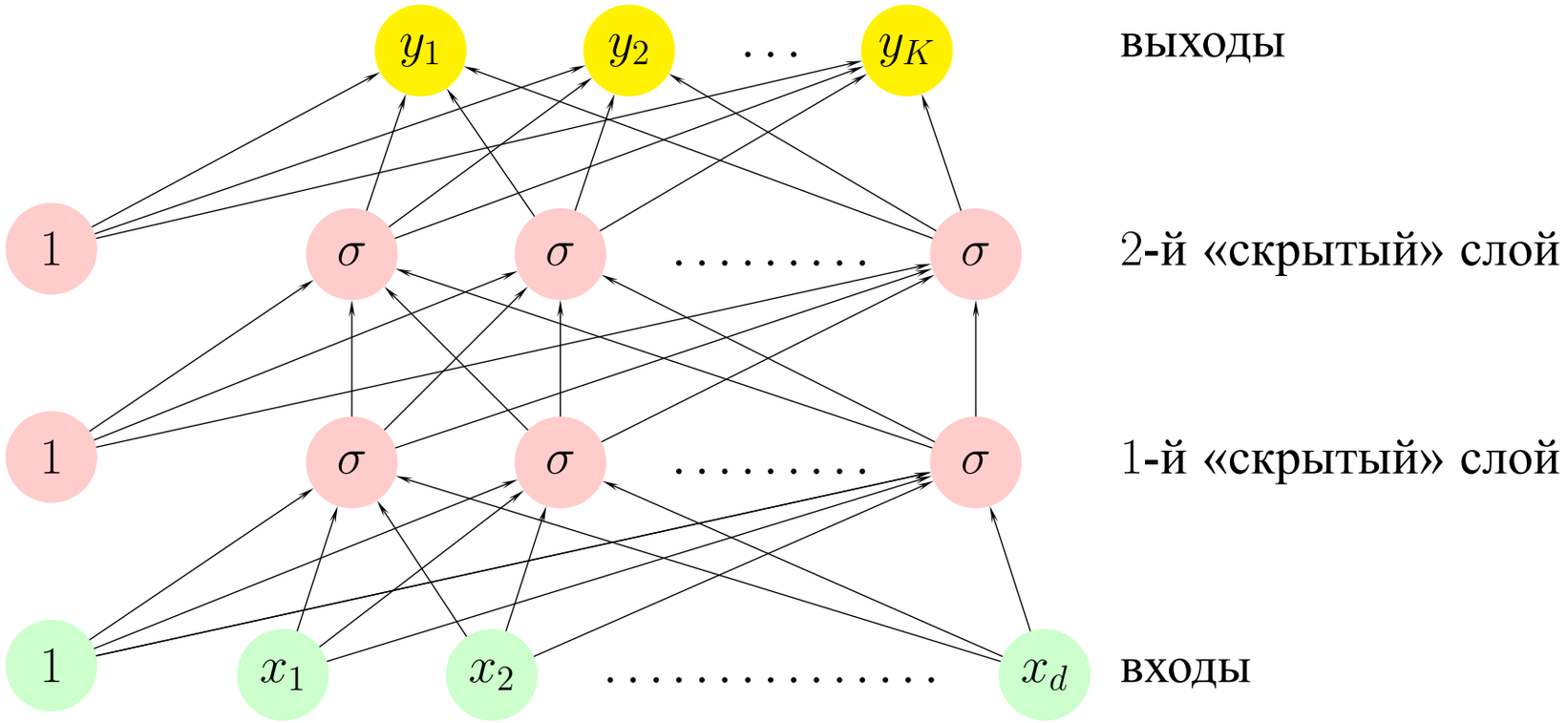
$$g = \ln\bigl(1 + \exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_q x_q)\bigr)$$

— эти функции успешно применяются для решения проблемы исчезающего градиента (см. ниже).



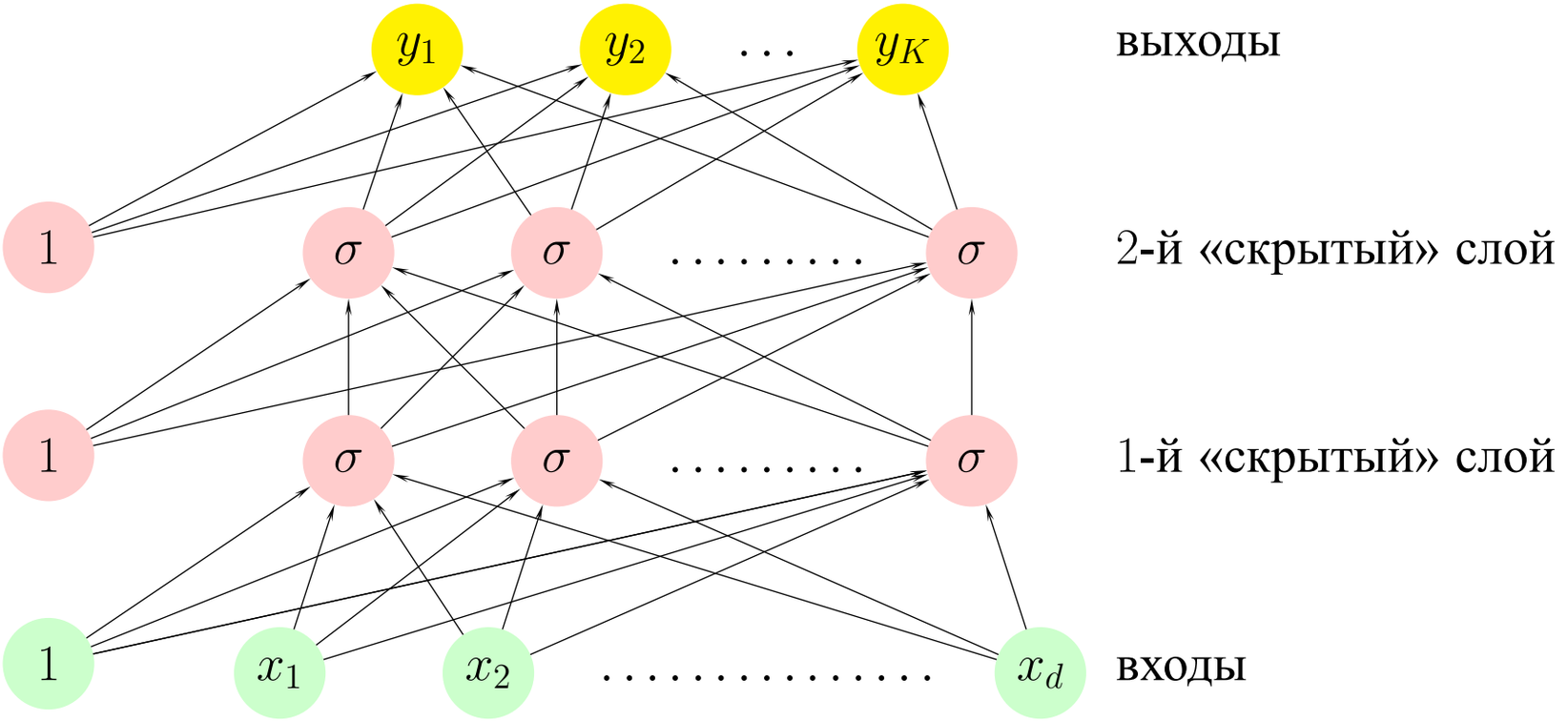
Нейронные сети часто изображают с дополнительными «единичными» нейронами.

Веса указываются рядом с соответствующими дугами.



Нейронные сети часто изображают с дополнительными «единичными» нейронами.

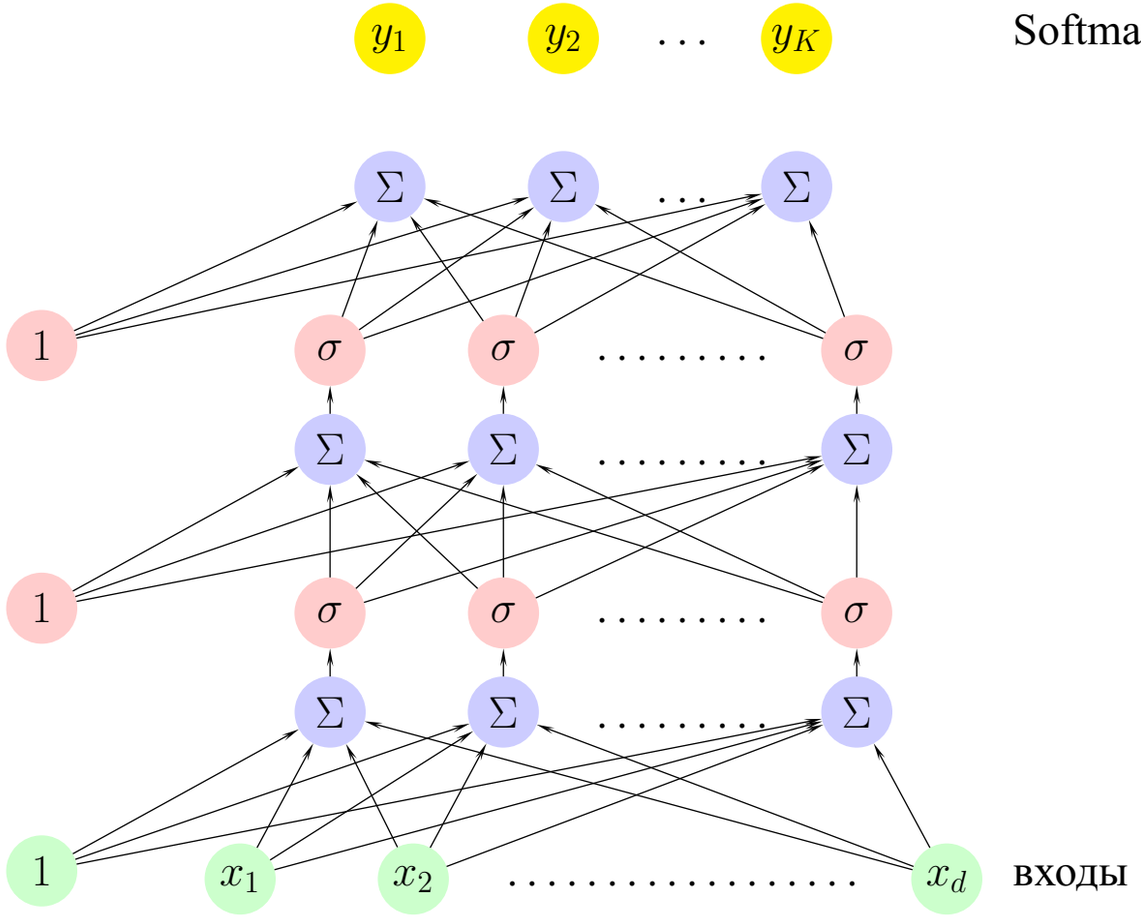
Веса указываются рядом с соответствующими дугами.



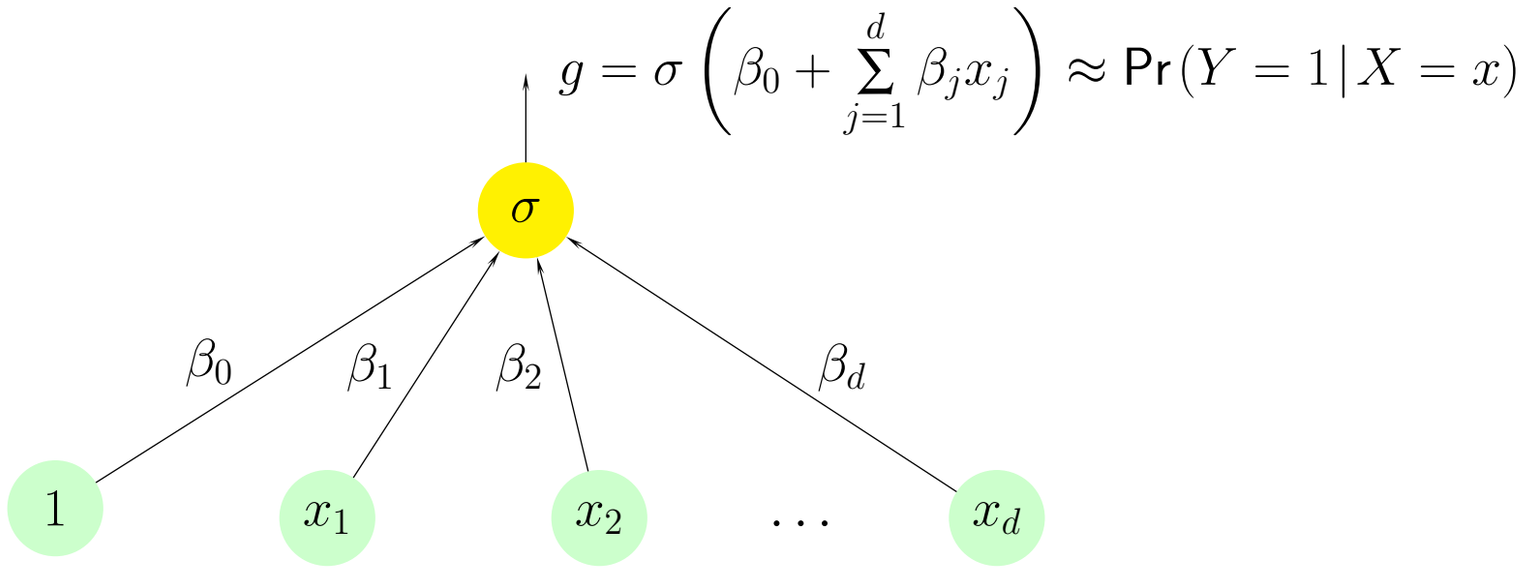
Таким образом, выходы из каждого узла (нейрона) умножаются на соответствующие веса и складываются.

Далее к полученному результату s применяется функция $\sigma(s)$.

Иногда отдельно изображают суммирующие элементы и элементы, вычисляющие σ:

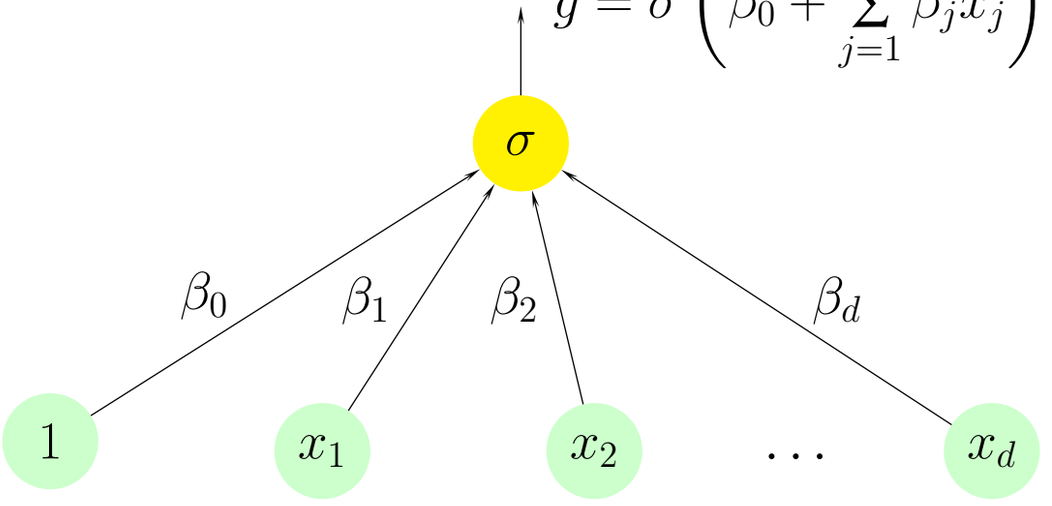


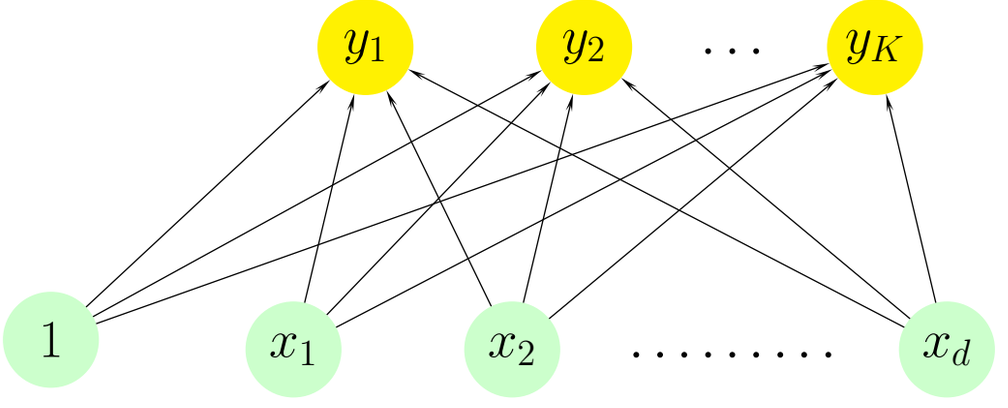
«Двуслойная» нейронная сеть (с логистической функцией активации σ) — это логистическая регрессия.



«Двуслойная» нейронная сеть (с логистической функцией активации σ) — это логистическая регрессия.

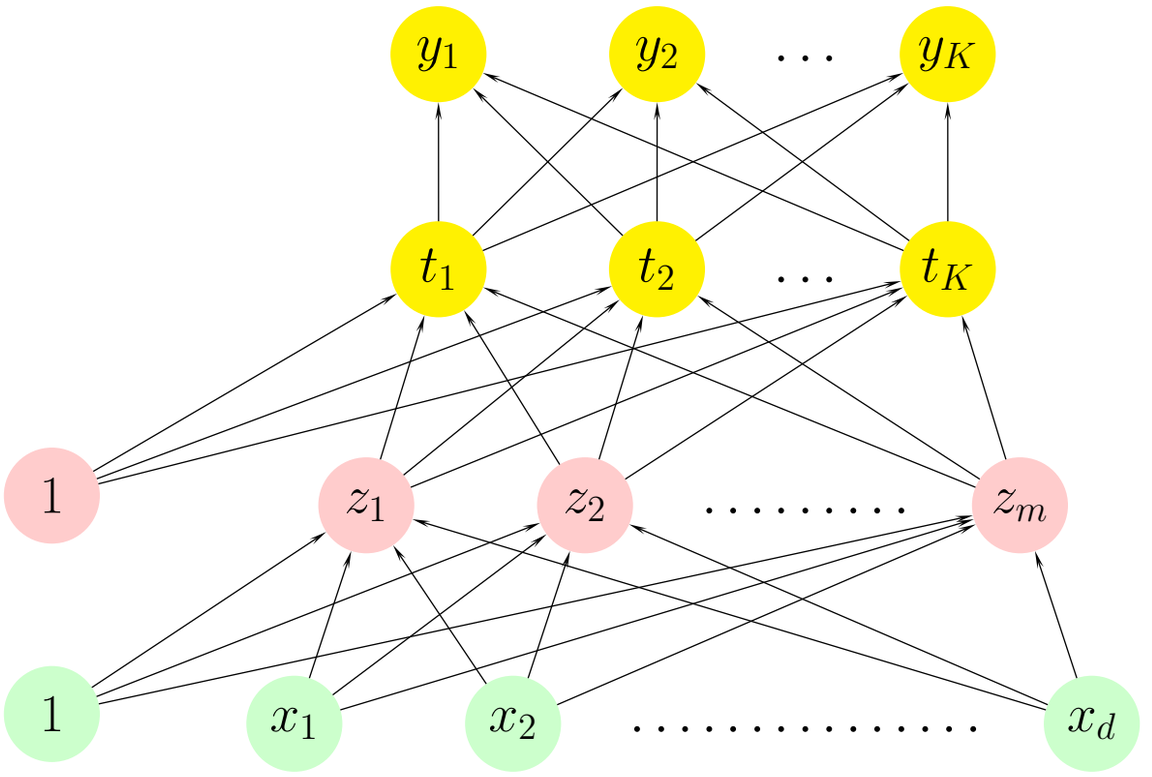
$$g = \sigma\left(\beta_0 + \sum_{j=1}^{d} \beta_j x_j\right) \approx \mathsf{Pr}(Y = 1 \,|\, X = x)$$





$$y_k = \frac{\exp\left(\beta_{k0} + \sum_{j=1}^{d} \beta_{kj} x_j\right)}{\sum_{\ell} \exp\left(\beta_{\ell 0} + \sum_{j=1}^{d} \beta_{\ell j} x_j\right)} \approx \mathsf{Pr}(k \,|\, x)$$

Трехслойная («ванилла») нейронная сеть для задачи классификации

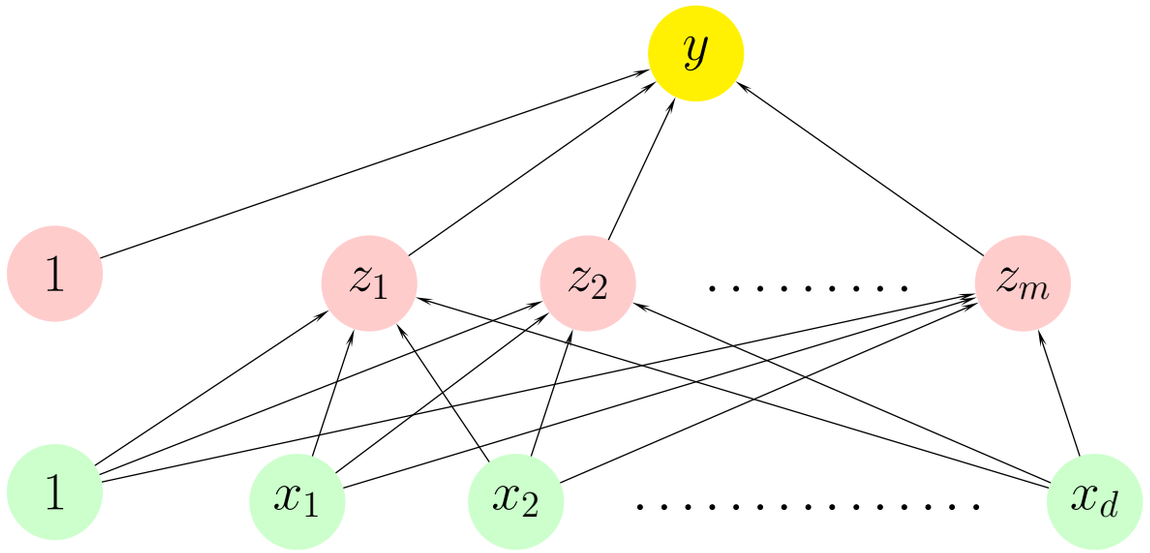


$$y_k = \frac{e^{t_k}}{\sum\limits_{\ell} e^{t_\ell}} \approx \mathsf{Pr}(k \,|\, x)$$

$$t_k = \gamma_{k0} + \sum_{i=1}^{m} \gamma_{ki} z_i$$

$$z_i = \sigma\left(\beta_{i0} + \sum_{j=1}^{d} \beta_{ij} x_j\right)$$

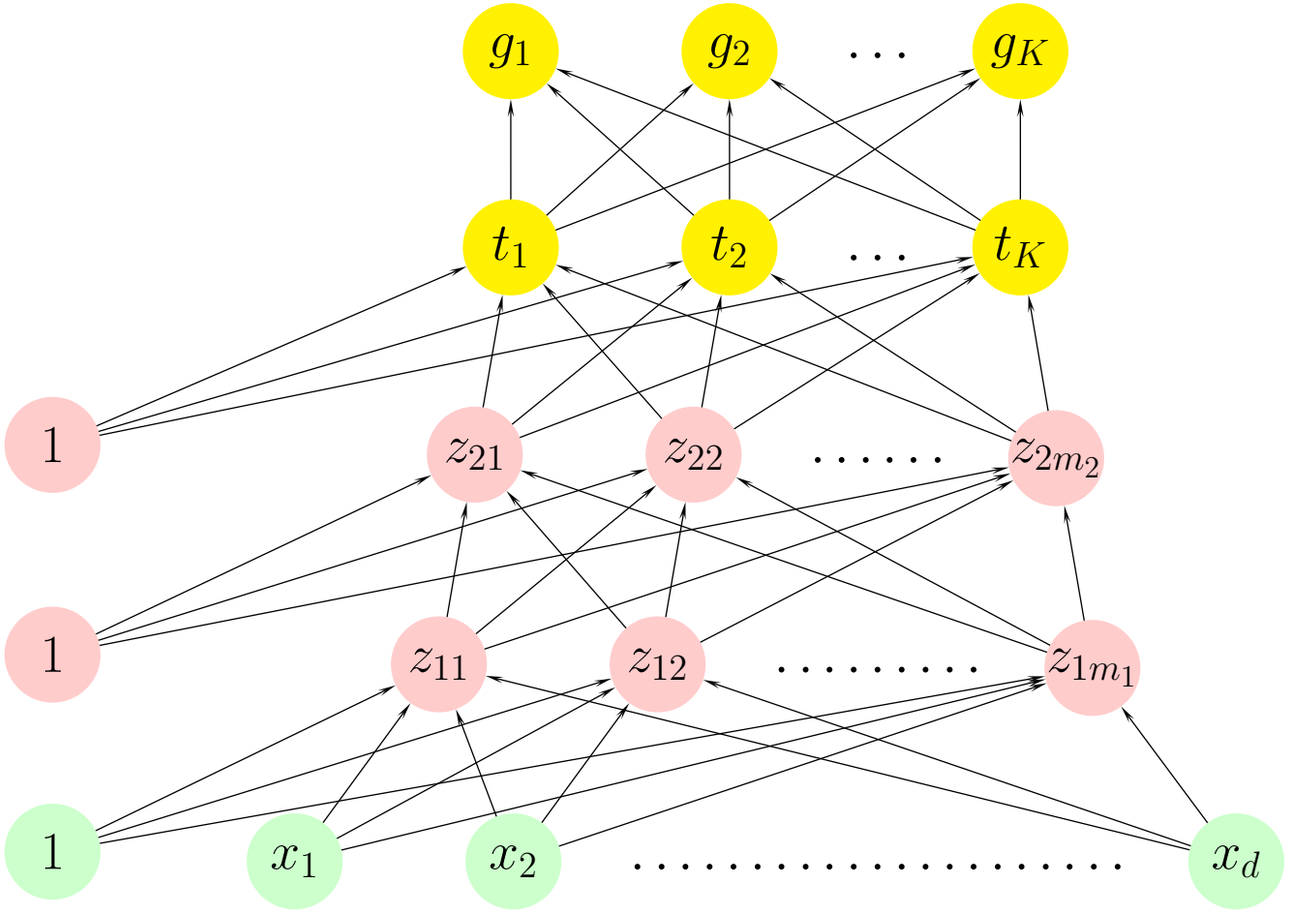
Трехслойная («ванилла») нейронная сеть для задачи восстановления регрессии



$$y = \gamma_0 + \sum_{i=1}^{m} \gamma_i z_i$$

$$z_i = \sigma\left(\beta_{i0} + \sum_{j=1}^{d} \beta_{ij} x_j\right)$$

«Четырехслойная» нейронная сеть для задачи классификации



$$g_k = \frac{e^{t_k}}{\sum\limits_{\ell=1}^{K} e^{t_\ell}} \approx \mathsf{Pr}(k \,|\, x)$$

$$t_k = \gamma_{k0} + \sum_{i=1}^{m_2} \gamma_{ki} z_{2i}$$

$$z_{2i} = \sigma\left(\beta_{2i0} + \sum_{j=1}^{m_1} \beta_{2ij} z_{1j}\right)$$

$$z_{1i} = \sigma\left(\beta_{1i0} + \sum_{j=1}^{d} \beta_{1ij} x_j\right)$$

$$z_1 = \sigma(B_1 x), \quad z_2 = \sigma(B_2 z_1), \quad t = B_3 z_2, \quad g = \mathrm{softmax}(t)$$

Почему нейронные сети?

Почему нейронные сети?

- $\&$, $\vee$ — пороговые функции. На самом деле из 16 булевых функций двух переменных не являются пороговыми только xor $\oplus$ (сложение по модулю 2) и его отрицание.

Почему нейронные сети?

- $\&$, $\vee$ — пороговые функции. На самом деле из 16 булевых функций двух переменных не являются пороговыми только xor $\oplus$ (сложение по модулю 2) и его отрицание.
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно для представления всех булевых функций, зависящих от d-переменных (а сколько всего элементов требуется?)

Почему нейронные сети?

- $\&$, $\vee$ — пороговые функции. На самом деле из 16 булевых функций двух переменных не являются пороговыми только xor $\oplus$ (сложение по модулю 2) и его отрицание.
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно для представления всех булевых функций, зависящих от d-переменных (а сколько всего элементов требуется?)
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно, чтобы выделять выпуклые многогранные области (конъюнкция полупространств), а 4-слойных — чтобы выделять невыпуклые, в т.ч. многосвязные области (дизъюнкция многогранных областей).

Почему нейронные сети?

- $\&$, $\vee$ — пороговые функции. На самом деле из 16 булевых функций двух переменных не являются пороговыми только xor $\oplus$ (сложение по модулю 2) и его отрицание.
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно для представления всех булевых функций, зависящих от d-переменных (а сколько всего элементов требуется?)
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно, чтобы выделять выпуклые многогранные области (конъюнкция полупространств), а 4-слойных — чтобы выделять невыпуклые, в т.ч. многосвязные области (дизъюнкция многогранных областей).
- 3-слойные нейронные сети с сигмоидальными функциями активации выделяют «сглаженные» многогранные области.

Почему нейронные сети?

- $\&$, $\vee$ — пороговые функции. На самом деле из 16 булевых функций двух переменных не являются пороговыми только xor $\oplus$ (сложение по модулю 2) и его отрицание.
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно для представления всех булевых функций, зависящих от d-переменных (а сколько всего элементов требуется?)
- 2-слойных нейронных сетей с пороговыми функциями активации достаточно, чтобы выделять выпуклые многогранные области (конъюнкция полупространств), а 4-слойных — чтобы выделять невыпуклые, в т.ч. многосвязные области (дизъюнкция многогранных областей).
- 3-слойные нейронные сети с сигмоидальными функциями активации выделяют «сглаженные» многогранные области.

Play with NN: http://playground.tensorflow.org

Отступление. История вопроса

Отступление. История вопроса

- Теорема Вейерштрасса: любую непрерывную функцию можно аппроксимировать полиномами (но сколько их достаточно?). Есть обобщения этой теоремы Стоуна и Горбаня. Есть еще ряды Фурье.

Отступление. История вопроса

- Теорема Вейерштрасса: любую непрерывную функцию можно аппроксимировать полиномами (но сколько их достаточно?). Есть обобщения этой теоремы Стоуна и Горбаня. Есть еще ряды Фурье.
- Колмогоров, Арнольд: любая непрерывная функция $f(x_1, x_2, \ldots, x_d)$ от d аргументов представима в виде

$$f(x_1, x_2, \ldots, x_d) = \sum_{k=1}^{2d+1} h_k \left(\sum_{j=1}^{d} \varphi_{jk}(x_j) \right),$$

где h_k, φ_{jk} — непрерывные функции, причем φ_{jk} не зависят от f (13-я проблема Гильберта). Таким образом, 3 слоев достаточно (но функции активации не пороговые и не близкие к ним.)

«Универсальная теорема об аппроксимации»

Теорема 10.1 *Пусть σ — ограниченная, монотонно возрастающая, непрерывная функция. Тогда для любой непрерывной функции $f^* : [0,1]^d \to \mathbf{R}$, для любого $\varepsilon > 0$ существуют M, α_m $(m = 1, 2, \ldots, M)$, w_{mj} $(m = 1, 2, \ldots, M;\ j = 0, 1, \ldots, d)$, такие, что функция*

$$f(x_1, \ldots, x_d) = \sum_{m=1}^{M} \alpha_m \sigma \left(w_{m0} + \sum_{j=1}^{d} w_{mj} x_j \right)$$

является ε-аппроксимацией функции f, т. е. $|f(x) - f^*(x)| \leq \varepsilon$.

«Универсальная теорема об аппроксимации»

Теорема 10.1 *Пусть σ — ограниченная, монотонно возрастающая, непрерывная функция. Тогда для любой непрерывной функции $f^* : [0,1]^d \to \mathbf{R}$, для любого $\varepsilon > 0$ существуют M, α_m $(m = 1, 2, \ldots, M)$, w_{mj} $(m = 1, 2, \ldots, M;\ j = 0, 1, \ldots, d)$, такие, что функция*

$$f(x_1, \ldots, x_d) = \sum_{m=1}^{M} \alpha_m \sigma \left(w_{m0} + \sum_{j=1}^{d} w_{mj} x_j \right)$$

является ε-аппроксимацией функции f, т. е. $|f(x) - f^*(x)| \leq \varepsilon$.

Таким образом, 3 слоев достаточно.

Но чему равно M — число узлов скрытого слоя?
(есть результаты, например, в терминах первых моментов преобразования Фурье)

10.2. Обучение нейронной сети (backpropagation)

[H. J. Kelley, 1960; A. E. Bryson, 1961; S. Dreyfus, 1962; S. Linnainmaa, 1970; **А. И. Галушкин**, 1974; **P. Werbos**, 1974; С. И. Барцев, В. А. Охонин, 1986; **D. E. Rumelhart et. al.**, 1986]

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку. Минимизируем эмпирический риск.

10.2. Обучение нейронной сети (backpropagation)

[H. J. Kelley, 1960; A. E. Bryson, 1961; S. Dreyfus, 1962; S. Linnainmaa, 1970; **А. И. Галушкин**, 1974; **P. Werbos**, 1974; С. И. Барцев, В. А. Охонин, 1986; **D. E. Rumelhart et. al.**, 1986]

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку. Минимизируем эмпирический риск.

В задачах восстановления регрессии, например, минимизируют сумму квадратов:

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\frac{1}{2}\Big(y^{(i)} - f(x^{(i)})\Big)^2}_{R^{(i)}} \to \min \qquad \text{(a)}$$

10.2. Обучение нейронной сети (backpropagation)

[H. J. Kelley, 1960; A. E. Bryson, 1961; S. Dreyfus, 1962; S. Linnainmaa, 1970; **А. И. Галушкин**, 1974; **P. Werbos**, 1974; С. И. Барцев, В. А. Охонин, 1986; **D. E. Rumelhart et. al.**, 1986]

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку. Минимизируем эмпирический риск.

В задачах восстановления регрессии, например, минимизируют сумму квадратов:

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\frac{1}{2}\Big(y^{(i)} - f(x^{(i)})\Big)^2}_{R^{(i)}} \to \min \qquad \text{(a)}$$

В задачах классификации обычно минимизируют *кросс-энтропию*:

$$R(w) = -\frac{1}{N}\sum_{i=1}^{N} \underbrace{\sum_{k=1}^{K} I(y^{(i)} = k) \ln g_k(x^{(i)})}_{R^{(i)}} \to \min \qquad \text{(b)}$$

10.2. Обучение нейронной сети (backpropagation)

[H. J. Kelley, 1960; A. E. Bryson, 1961; S. Dreyfus, 1962; S. Linnainmaa, 1970; **А. И. Галушкин**, 1974; **P. Werbos**, 1974; С. И. Барцев, В. А. Охонин, 1986; **D. E. Rumelhart et. al.**, 1986]

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку. Минимизируем эмпирический риск.

В задачах восстановления регрессии, например, минимизируют сумму квадратов:

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\frac{1}{2}\Big(y^{(i)} - f(x^{(i)})\Big)^2}_{R^{(i)}} \to \min \qquad \text{(a)}$$

В задачах классификации обычно минимизируют *кросс-энтропию*:

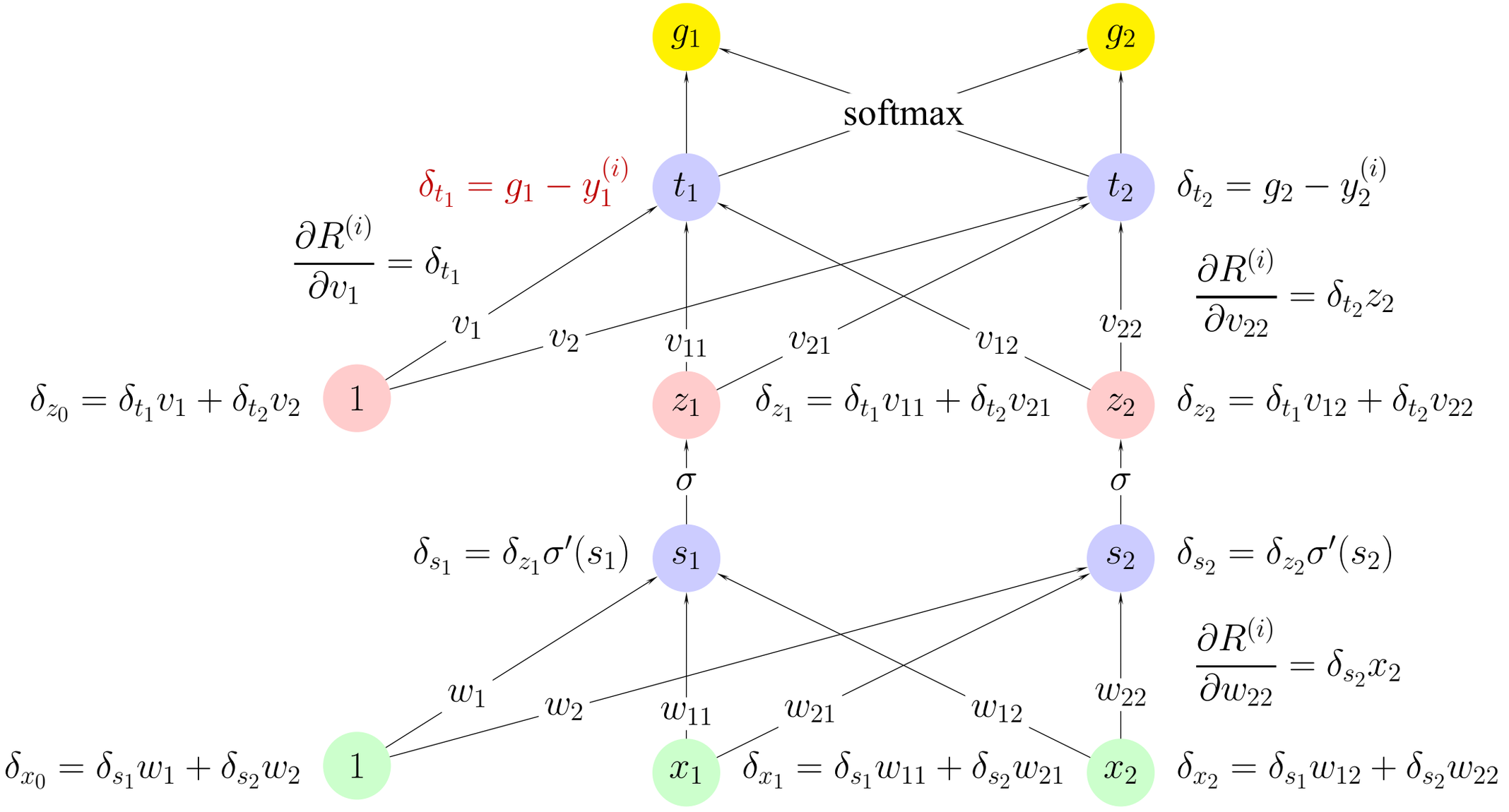
$$R(w) = -\frac{1}{N}\sum_{i=1}^{N} \underbrace{\sum_{k=1}^{K} I(y^{(i)} = k) \ln g_k(x^{(i)})}_{R^{(i)}} \to \min \qquad \text{(b)}$$

Задача минимизации $R(w)$ решается численными методами: обычно методом градиентного спуска и его вариантами, но можно пытаться использовать и другие методы.

Нужно уметь вычислять производную.

Нужно уметь вычислять производную.

Алгоритм *обратного распространения ошибки* (*back propagation*) — это метод вычисления градиента $\partial R/\partial w$ ($\partial R^{(i)}/\partial w$). Главное — цепное правило (взятие производной сложной функции)!


g_1
g_2
softmax
t_1
t_2
$\delta_{t_1} = g_1 - y_1^{(i)}$
$\delta_{t_2} = g_2 - y_2^{(i)}$
$\frac{\partial R^{(i)}}{\partial v_1} = \delta_{t_1}$
$\frac{\partial R^{(i)}}{\partial v_{22}} = \delta_{t_2} z_2$
v_1
v_2
v_{11}
v_{21}
v_{12}
v_{22}
1
z_1
z_2
$\delta_{z_0} = \delta_{t_1} v_1 + \delta_{t_2} v_2$
$\delta_{z_1} = \delta_{t_1} v_{11} + \delta_{t_2} v_{21}$
$\delta_{z_2} = \delta_{t_1} v_{12} + \delta_{t_2} v_{22}$
σ
σ
s_1
s_2
$\delta_{s_1} = \delta_{z_1} \sigma'(s_1)$
$\delta_{s_2} = \delta_{z_2} \sigma'(s_2)$
$\frac{\partial R^{(i)}}{\partial w_{22}} = \delta_{s_2} x_2$
w_1
w_2
w_{11}
w_{21}
w_{12}
w_{22}
1
x_1
x_2
$\delta_{x_0} = \delta_{s_1} w_1 + \delta_{s_2} w_2$
$\delta_{x_1} = \delta_{s_1} w_{11} + \delta_{s_2} w_{21}$
$\delta_{x_2} = \delta_{s_1} w_{12} + \delta_{s_2} w_{22}$

$$g = \text{softmax}\big(V\sigma(Wx + w) + v\big)$$

$$R^{(i)} = \text{logloss}(g) = \text{logloss}\Big(\text{softmax}\overbrace{\big(V\underbrace{\sigma\underbrace{(Wx + w)}_{s}}_{z} + v\big)}^{t}\Big)$$

$$\delta_x = \frac{\partial R^{(i)}}{\partial x} = \underbrace{\overbrace{(g - y)}^{\delta_t} \cdot V \cdot \text{diag}\big(\sigma'(s)\big)}_{\delta_z} \cdot W$$

$$\frac{\partial R^{(i)}}{\partial W} = \delta_s \cdot x, \qquad \frac{\partial R^{(i)}}{\partial w} = \delta_s, \qquad \frac{\partial R^{(i)}}{\partial V} = \delta_t \cdot z, \qquad \frac{\partial R^{(i)}}{\partial v} = \delta_t$$

$$W \leftarrow W - \rho\frac{\partial R^{(i)}}{\partial W}, \qquad w \leftarrow w - \rho\frac{\partial R^{(i)}}{\partial w}, \qquad V \leftarrow V - \rho\frac{\partial R^{(i)}}{\partial V}, \qquad v \leftarrow v - \rho\frac{\partial R^{(i)}}{\partial v}$$

$$w_{\ell j}^{(r+1)} = w_{\ell j}^{(r)} - \rho \sum_{i=1}^{N} \frac{\partial R^{(i)}}{\partial w_{\ell j}}, \qquad w_{\ell j}^{(r+1)} = w_{\ell j}^{(r)} - \rho \frac{\partial R^{(i)}}{\partial w_{\ell j}}$$

$$w_{\ell j}^{(r+1)} = w_{\ell j}^{(r)} - \rho \sum_{i=1}^{N} \frac{\partial R^{(i)}}{\partial w_{\ell j}}, \qquad w_{\ell j}^{(r+1)} = w_{\ell j}^{(r)} - \rho \frac{\partial R^{(i)}}{\partial w_{\ell j}}$$

procedure BackPropagation (on-line)

Инициализировать $w_{\ell j}$ (например, случайными значениями)

repeat

for $i = 1, \ldots, N$

Прямой ход: подать на вход $x^{(i)}$, вычислить все z_j

Обратный ход:

for $j \in$ Outputs

(a, b) $\delta_j \leftarrow z_j - y_j^{(i)}$

for каждого уровня, начиная с предпоследнего

for каждого узла j текущего уровня

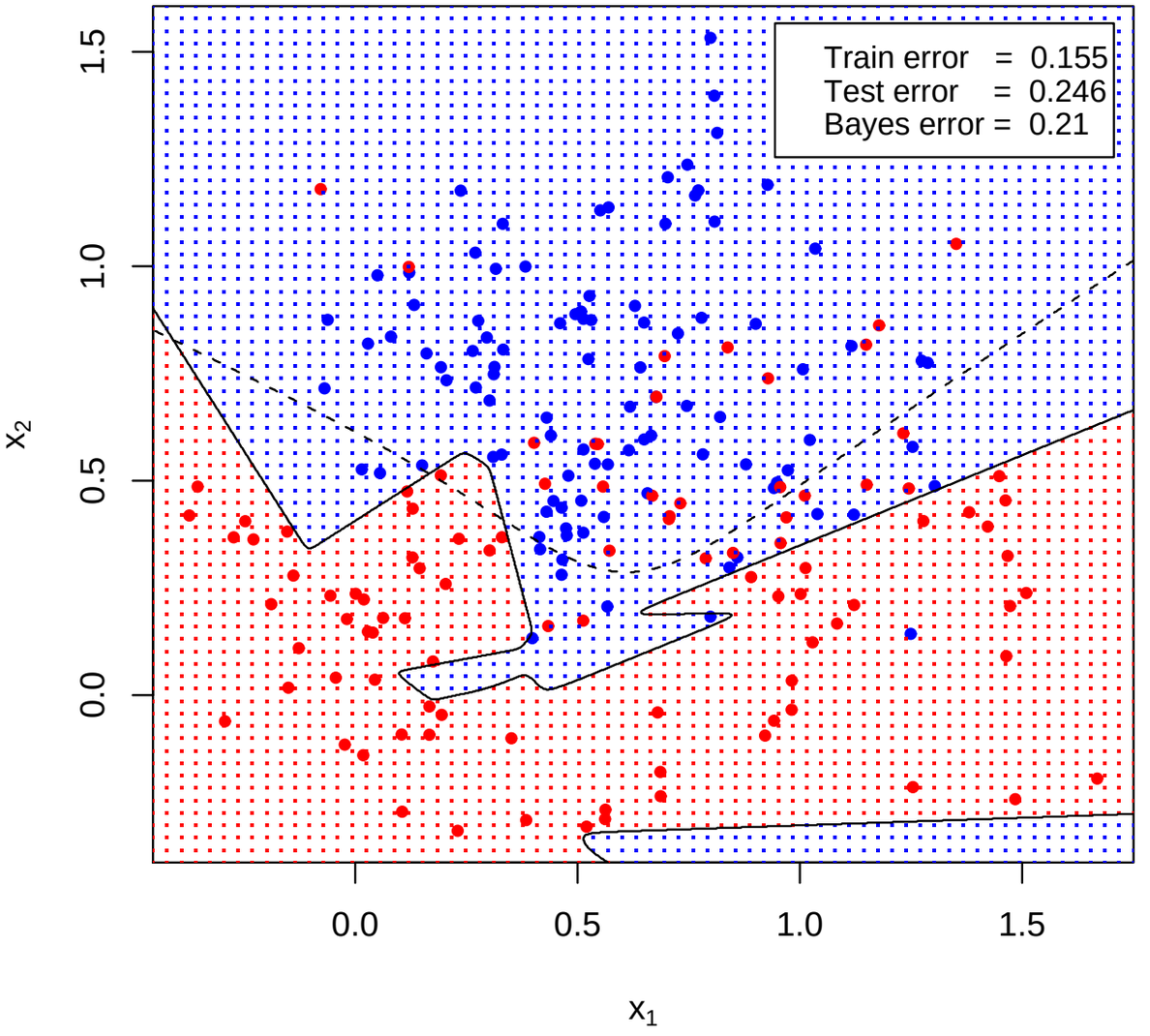
$\delta_j \leftarrow \sigma'(s_j) \sum_{k \in \text{Children}(j)} w_{jk} \delta_k$

for каждой дуги (ℓ, j)

$w_{\ell j} \leftarrow w_{\ell j} - \rho \delta_j z_l$

return $w_{\ell j}$

Классификация на основе «трехслойной» нейронной сети с 10 скрытыми узлами. Построенный классификатор сильно зависит от начальных значений.



10.3. Переобучение

10.3. Переобучение

Регуляризация — *weight decay*

10.3. Переобучение

Регуляризация — *weight decay*

Добавим штраф к функции ошибок: $R(w) + \lambda J(w)$, где

$$J(w) = \sum_{i\,i'} w_{i\,i'}^2, \qquad (\star)$$

$\lambda \geq 0$

10.3. Переобучение

Регуляризация — *weight decay*

Добавим штраф к функции ошибок: $R(w) + \lambda J(w)$, где

$$J(w) = \sum_{i\,i'} w_{i\,i'}^2, \qquad (\star)$$

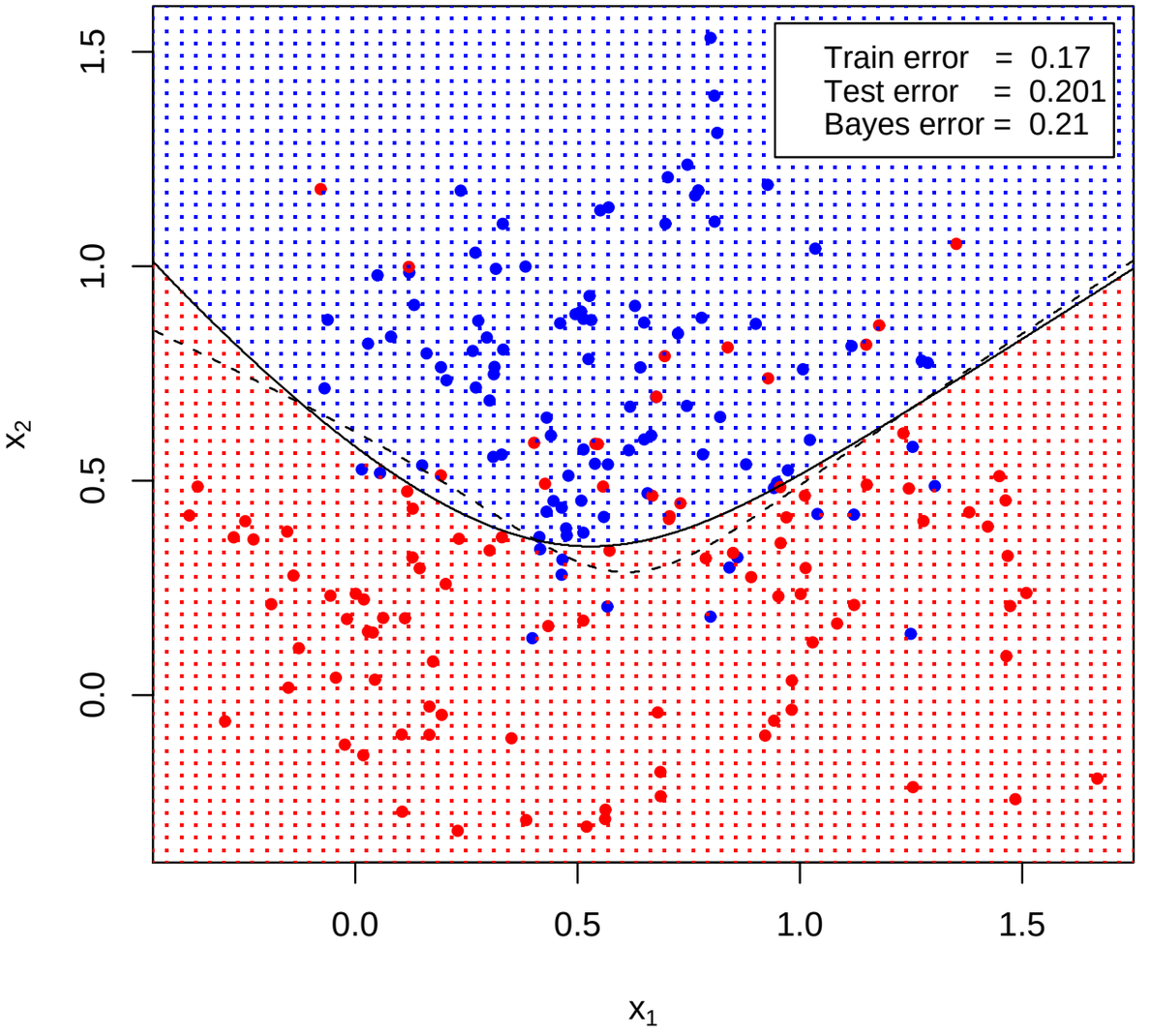
$\lambda \geq 0$

Вариант:

$$J(w) = \sum_{i\,i'} \frac{w_{i\,i'}^2}{1 + w_{i\,i'}^2},$$

— большие коэффициенты подвергаются бóльшему сокращению, чем это делает $(\star)$

Классификация на основе «трехслойной» нейронной сети с 10 скрытыми узлами и коэффициентом $\lambda = \text{decay weigth} = 0.1$



10.3.1. Dropout

Другой метод борьбы с переобучением (это тоже регуляризация).

«Отключение» нейронов на этапе обучения.

На каждой итерации обучения каждый нейрон на полносвязных слоях с некоторой вероятностью (например, 0.5) «отключается», т.е. не участвует ни в прямом, ни в обратном ходе.

В режиме предсказания все нейроны работают.

В примере `digits` выборка размера 1934 была случайным образом разбита на две группы по 967 объектов в каждой.

Для обучения использовалась «трехслойная» нейронная сеть с 15 элементами в скрытом слое.

100 итераций алгоритма BFGS

decay weight	*Ошибка*	
	на обучающей выборке	*на тестовой выборке*
0	0.000	0.111
0.1	0.000	0.078
0.2	0.000	0.035
0.3	0.000	0.037
0.4	0.000	0.039
0.5	0.000	0.037
0.6	0.000	0.035
0.7	0.000	0.029
0.8	0.000	0.036
0.9	0	0.037
1.0	0	0.033
5.0	0.007	0.039
10.0	0.027	0.047
50.0	0.249	0.267

Рукописные цифры, которые были не правильно классифицированы нейронной сетью с decay weight $= 0.7$. Цифра рядом с каждым изображением — ответ нейронной сети