

Machine Learning

# Классификация



Брак1.jpg



Брак2.jpg



Брак3.jpg



Брак4.jpg



Брак5.jpg



Брак6.jpg



Брак7.jpg



Брак8.jpg



Брак9.jpg



Брак10.jpg



Брак11.jpg



Брак12.jpg



Брак13.jpg



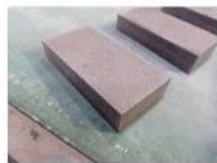
Брак14.jpg



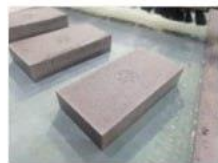
Брак15.jpg



Годный1.jpg



Годный2.jpg



Годный3.jpg



Годный4.jpg



Годный5.jpg



Годный6.jpg



Годный7.jpg



Годный8.jpg



Годный9.jpg

# Что такое классификация в машинном обучении?

- Классификация - это задача машинного обучения, которая выражается в предсказании дискретного значения.
- Классификация - это задача обучения с учителем, поэтому в датасете должны быть “правильные ответы” - значения целевой переменной.
- Классификация - самая распространенная задача машинного обучения на практике.
- Классификация бывает бинарной и множественной, одноклассовой и мультиклассовой.
- Примеры задач классификации - распознавание объектов, генерация текстов, подбор тематики текстов, идентификация объектов на изображениях, распознавание речи, машинный перевод и так далее.
- Почти любую практическую задачу машинного обучения можно сформулировать как задачу классификации.

Как определяется задача классификации?

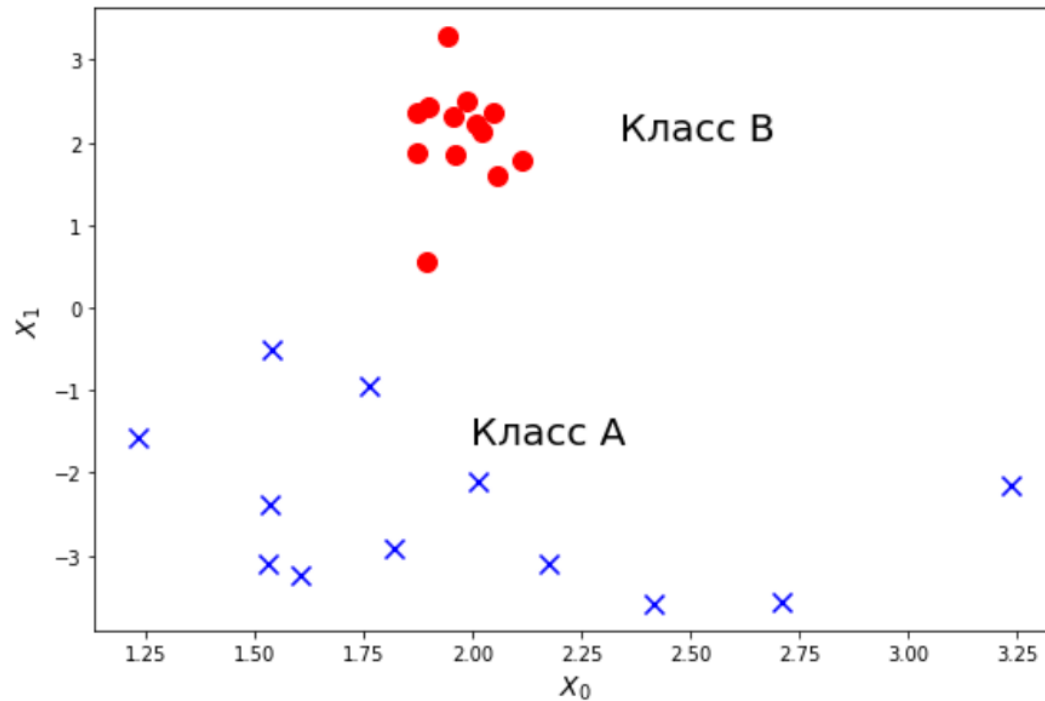
ВХОД  $x^{(i)} = (x_1, x_2, \dots, x_n)$

ВЫХОД  $y \in \{y_1, y_2, \dots, y_k\}$

Функция  
гипотезы  $y = h_{\theta}(x)$

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_m, y_m)$$

Как определяется задача классификации?



# Классификация

Email: Спам / Не спам?

Финансовые транзакции: Мошенничество (Да / Нет)?

Опухоли: Рак / Доброкачественная ?

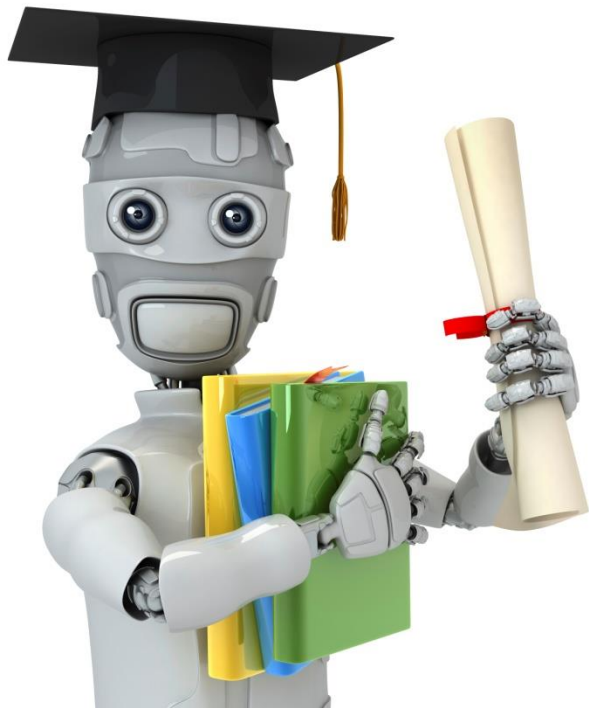
$$y \in \{0, 1\}$$

0: “Отрицательный класс”

1: “Положительный класс”

## Как определяется задача классификации?

1. На вход модели классификации подается вектор признаков объекта.
2. На выходе модель классификации предсказывает одно из конечного набора значений - метку класса объекта.
3. Мы часто будем изображать классификацию на графике, но имейте в виду, что на практике это обычно многомерная задача.
4. Обычно сначала рассматривается бинарная классификация, остальные типы строятся на ее основе.
5. Бинарная классификация - это про наличие или отсутствие какого-либо признака у объекта.



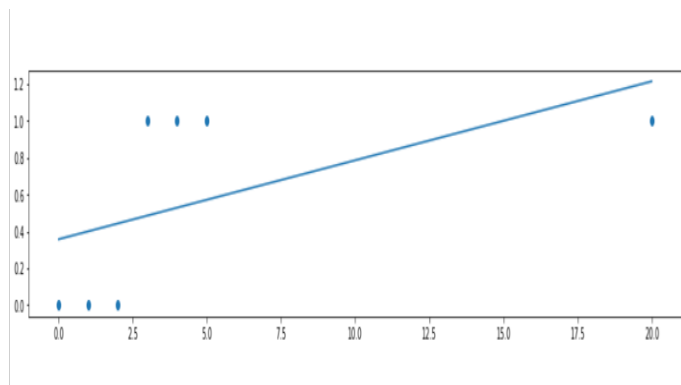
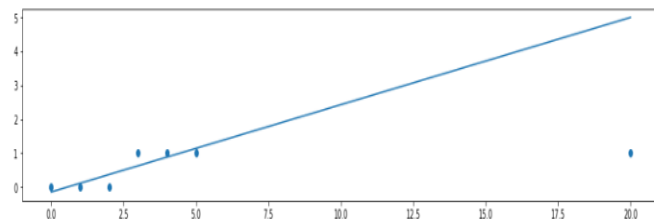
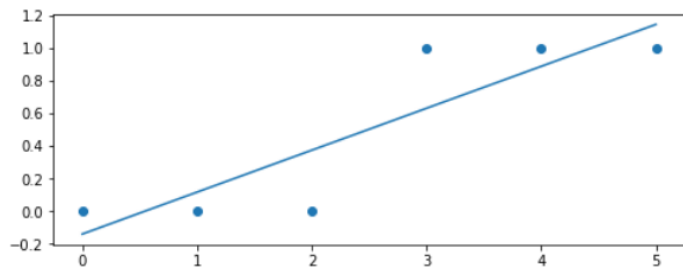
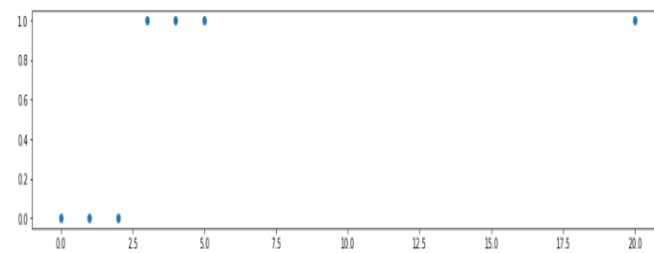
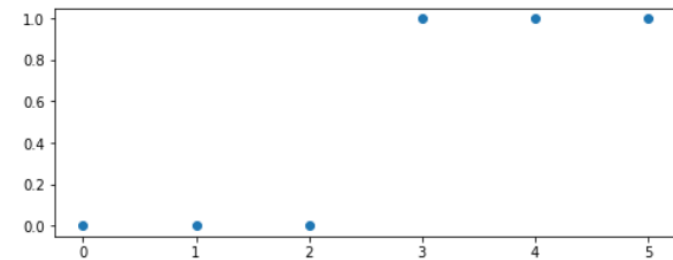
Machine Learning

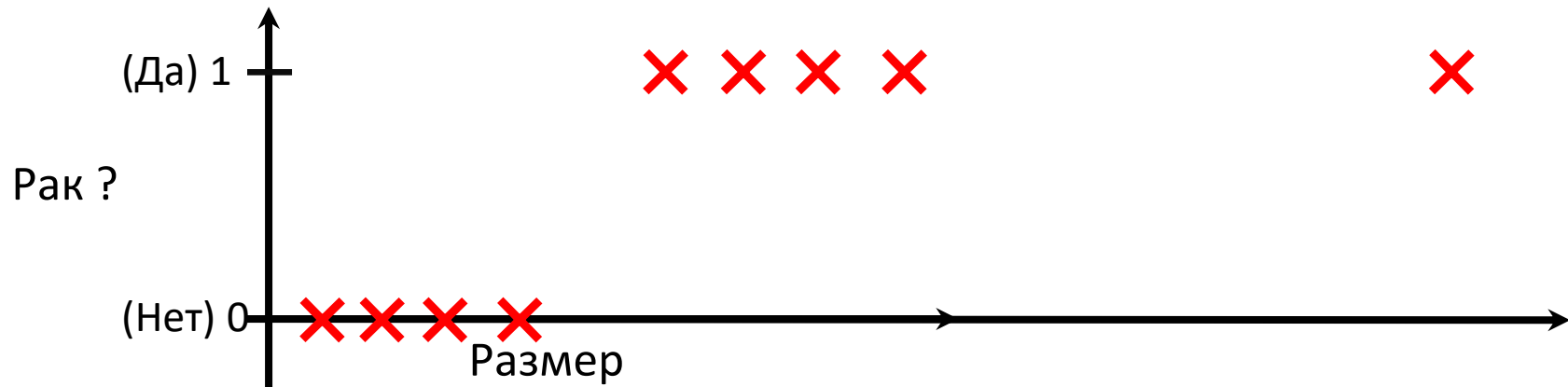
# Классификация

---

## Логистическая регрессия







Пороговый классификатор  $h_{\theta}(x)$  при 0.5:

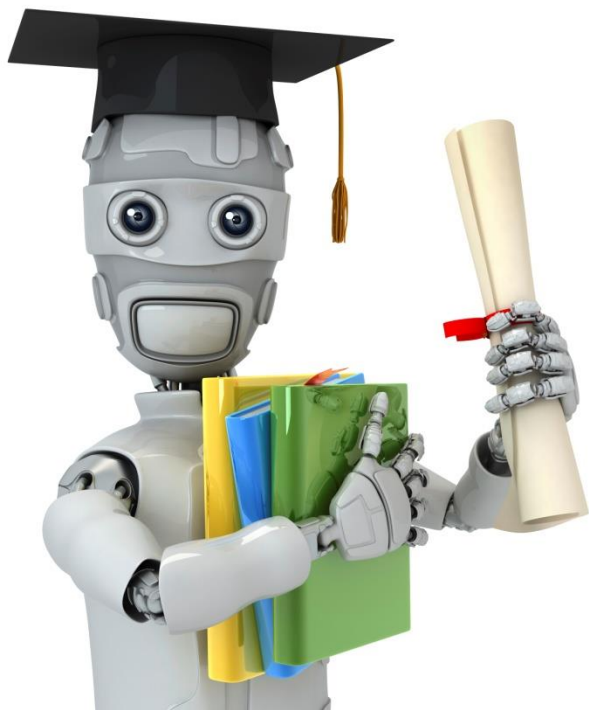
If  $h_{\theta}(x) \geq 0.5$ , predict "y = 1"

If  $h_{\theta}(x) < 0.5$ , predict "y = 0"

Классификация:  $y = 0$  or  $1$

$h_{\theta}(x)$  может быть  $> 1$  или  $< 0$

Логистическая регрессия:  $0 \leq h_{\theta}(x) \leq 1$



Machine Learning

# Логистическая регрессия

---

## Представление модели

## Модель логистической регрессии

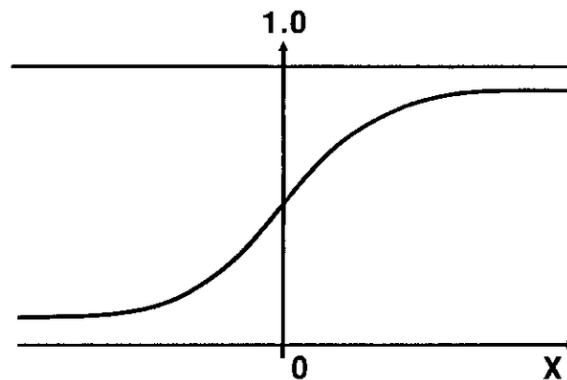
Нужно  $0 \leq h_{\theta}(x) \leq 1$

$$h_{\theta}(x) = \theta^T x$$

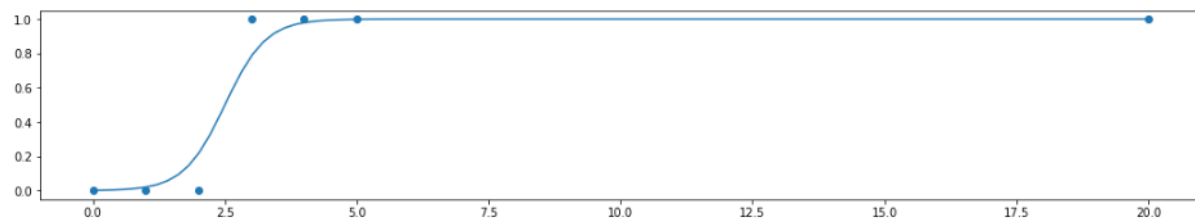
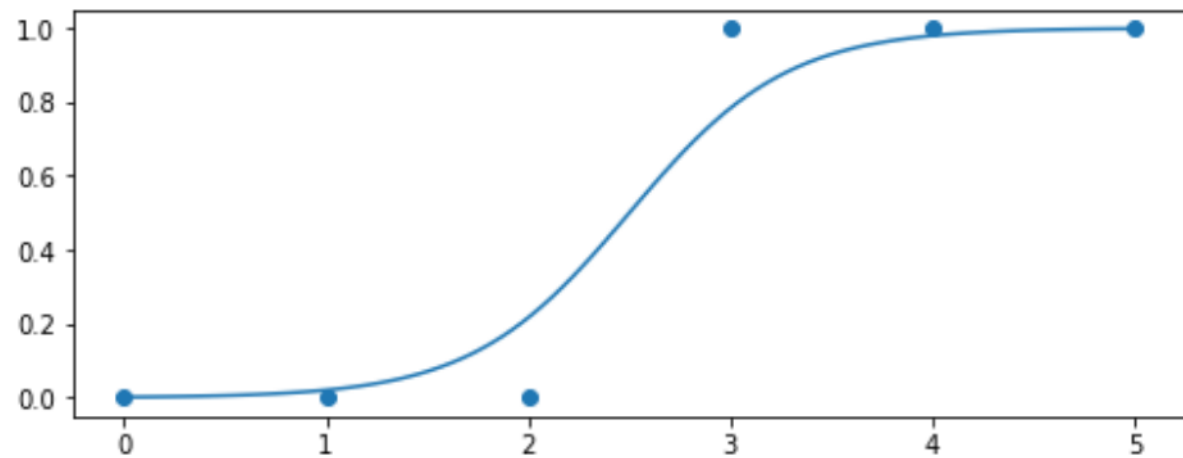
$$h_{\theta}(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid function  
Logistic function



*Sigmoid*  
 $f(x) = \frac{1}{1 + e^{-x}}$



## Выводы:

1. Логистическая регрессия - это самый простой алгоритм бинарной классификации.
2. Можно взять регрессионную модель и ввести пороговое значение.
3. Обычная регрессия плохо работает в задачах классификации за счет своей чувствительности и неограниченности.
4. Метод логистической регрессии основан на применении логистической или сигмоидной функции.
5. Результат работы логистической функции часто интерпретируется как вероятность отнесения объекта к положительному классу.
6. Для четкой классификации обычно выбирают некоторое пороговое значение, обычно - 0,5.

## Интерпретация результата

$h_{\theta}(x)$  = теоретическая вероятность  $y = 1$  при данном  $x$

Пример: Если  $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_{\theta}(x) = 0.7$$

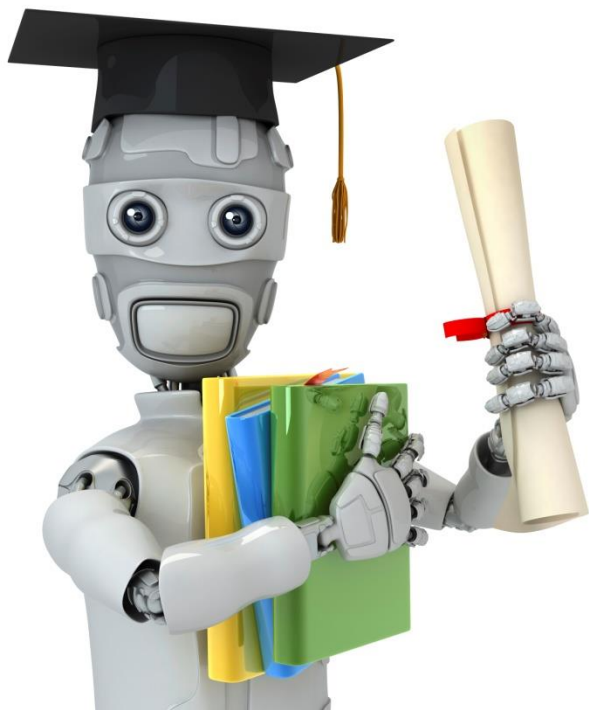
Пациенту сообщается, что 70% вероятность наличия рака

“вероятность  $y = 1$ , при данном  $x$ ,  
зависящая от  $\theta$ ”

$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$$

$$P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta)$$





Machine Learning

# Логистическая регрессия

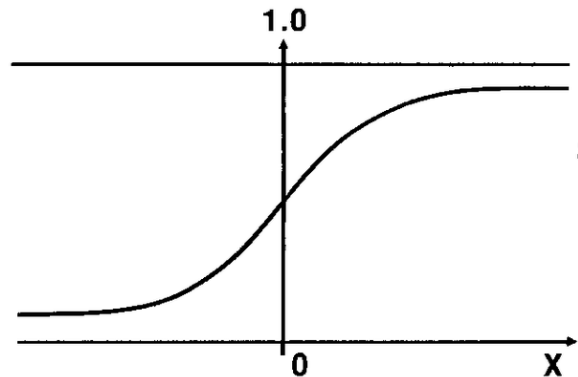
---

Граница принятия  
решения

# Логистическая регрессия

$$h_{\theta}(x) = g(\theta^T x)$$

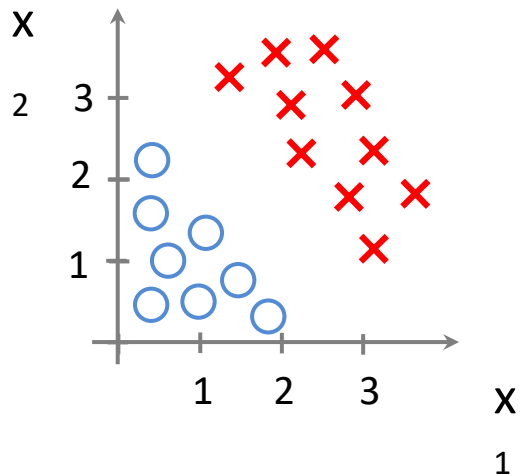
$$g(z) = \frac{1}{1+e^{-z}}$$



Предсказывается “ $y = 1$ ” if  $h_{\theta}(x) \geq 0.5$

предсказывается “ $y = 0$ ” if  $h_{\theta}(x) < 0.5$

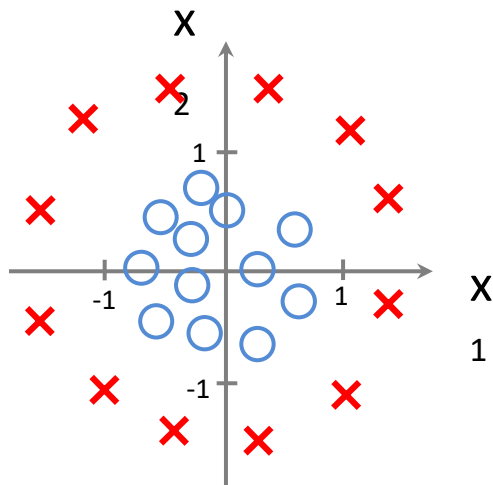
## Граница принятия решения



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$y = 1$  если  $-3 + x_1 + x_2 \geq 0$

## Non-linear decision boundaries



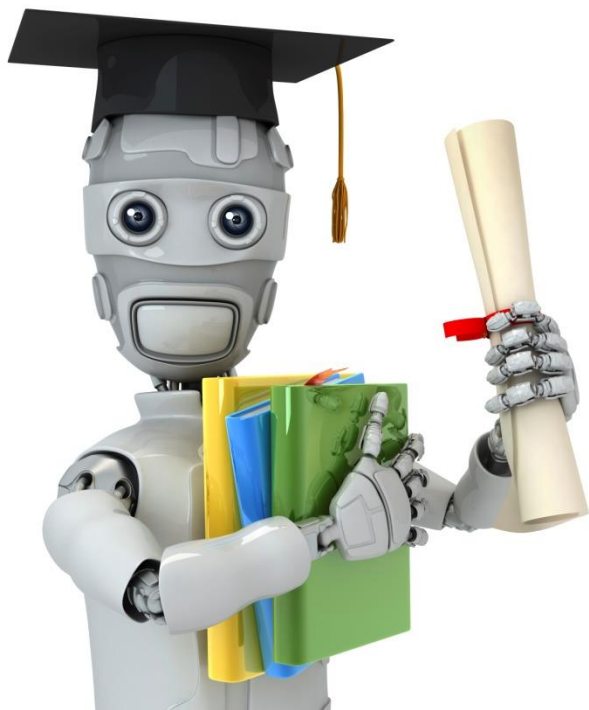
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$“y = 1” \text{ если } -1 + x_1^2 + x_2^2 \geq 0$$

$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

## Выводы:

1. Граница принятия решений - это область, отделяющая один класс от другого.
2. Форма границы принятия решения определяется видом используемой модели.
3. Данные бывают линейно разделимые или нет.
4. Логистическая регрессия - это линейный метод, поэтому она хорошо работает с линейно разделимыми данными.
5. Если данные линейно неразделимы можно попробовать ввести в модель полиномиальные признаки.



Machine Learning

# Логистическая регрессия

---

## Функция ошибки

Обучающая  
выборка:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

$m$  примеров  $x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

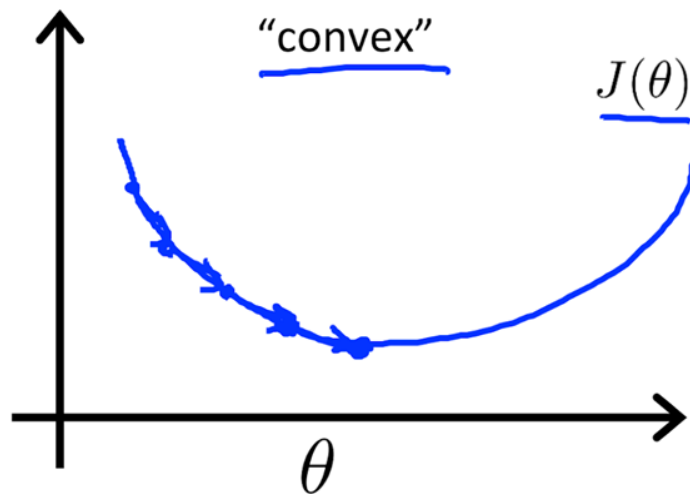
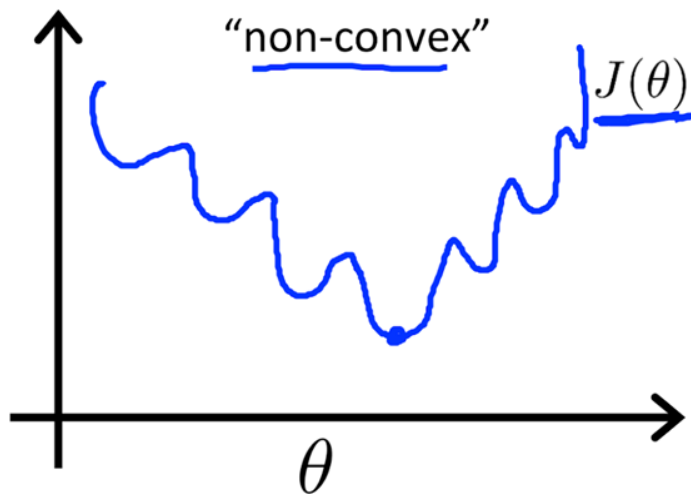
$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Как подобрать параметры  $\theta$  ?

## Функция ошибки

Линейная регрессия:  $J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$

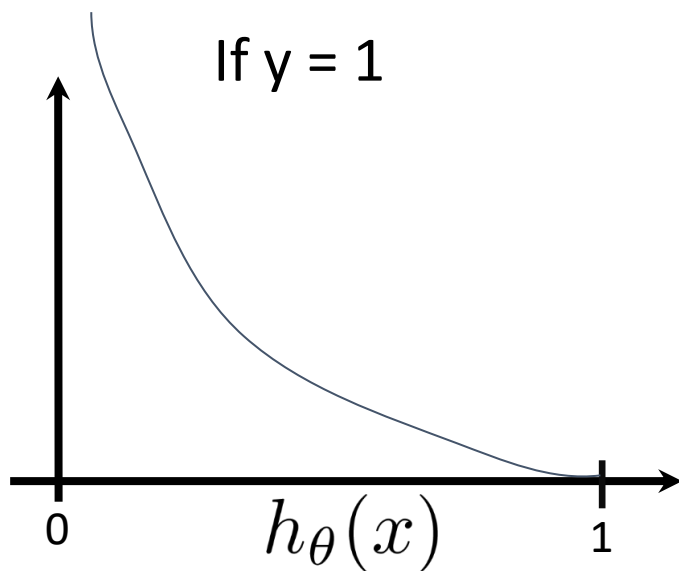
$$\text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$





## Функция ошибки логистической регрессии

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



Cost = 0 if  $y = 1, h_{\theta}(x) = 1$

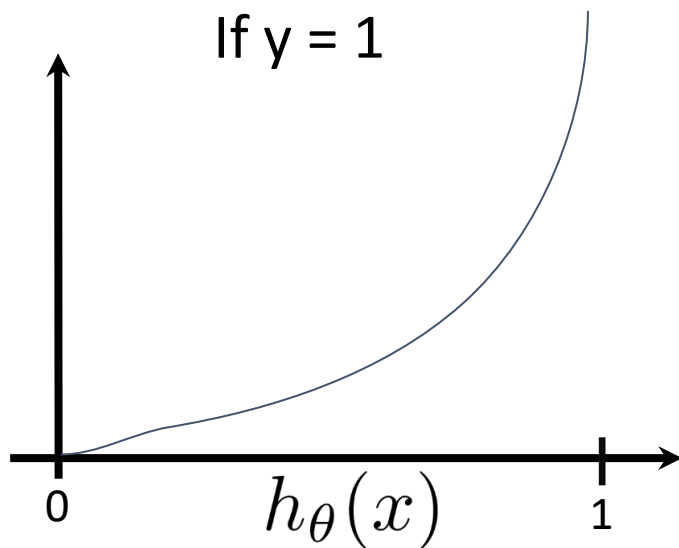
But as  $h_{\theta}(x) \rightarrow 0$

$\text{Cost} \rightarrow \infty$

Captures intuition that if  $h_{\theta}(x) = 0$ ,  
(predict  $P(y = 1|x; \theta) = 0$ ), but  $y = 1$ ,  
we'll penalize learning algorithm by a very  
large cost.

## Функция ошибки логистической регрессии

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



Cost = 0 if  $y = 1, h_{\theta}(x) = 1$

But as  $h_{\theta}(x) \rightarrow 0$

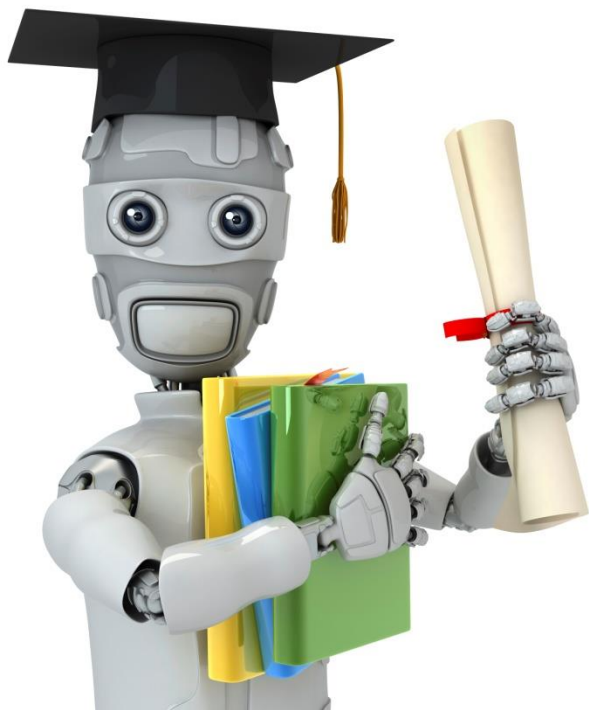
$\text{Cost} \rightarrow \infty$

Captures intuition that if  $h_{\theta}(x) = 0$ ,  
(predict  $P(y = 1|x; \theta) = 0$ ), but  $y = 1$ ,  
we'll penalize learning algorithm by a very  
large cost.

$$Cost(h_b(x), y) = -y \cdot \log(h_b(x)) - (1 - y)(1 - \log(h_b(x)))$$

$$J(\vec{b}) = -\frac{1}{m} \sum_{i=1}^m y_i \cdot \log(h_b(x_i)) + (1 - y_i)(1 - \log(h_b(x_i)))$$

$$\frac{\partial}{\partial b_i} J(\vec{b}) = \frac{1}{m} \sum_{i=1}^m (h_b(x_i) - y_i) x_i$$



Machine Learning

# Логистическая регрессия

---

Метод градиентного  
спуска

## Функция ошибки логистической регрессии

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

Note:  $y = 0$  or  $1$  always

## Функция ошибки логистической регрессии

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

Найти такие  $\theta$ :

$$\min_{\theta} J(\theta)$$

Предсказание новых значений  $x$  :

$$\text{Модель } h_{\theta}(x) = \frac{1}{1+e^{-\theta^T x}}$$

Алгоритм полностью идентичен!

$$\min_{\theta} J(\theta)$$

Повторить {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

(одновременное изменение  $\theta_j$  )

## Алгоритмы оптимизации

Имея  $\theta$ , и зная как вычислить в каждой точке

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$  (for  $j = 0, 1, \dots, n$ )

Алгоритмы оптимизации:

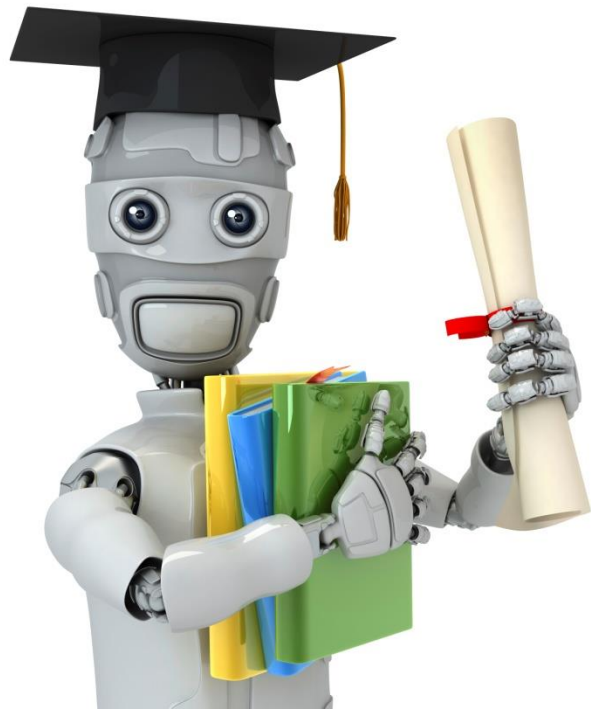
- Gradient descent
- [Conjugate gradient](#)
- [BFGS](#)
- L-BFGS

Преимущества:

- Не нужно подбирать  $\alpha$
- Чаще сходятся быстрее.

Недостатки:

- Более сложные



# Логистическая регрессия

---

Множественная классификация:  
один против всех

Machine Learning



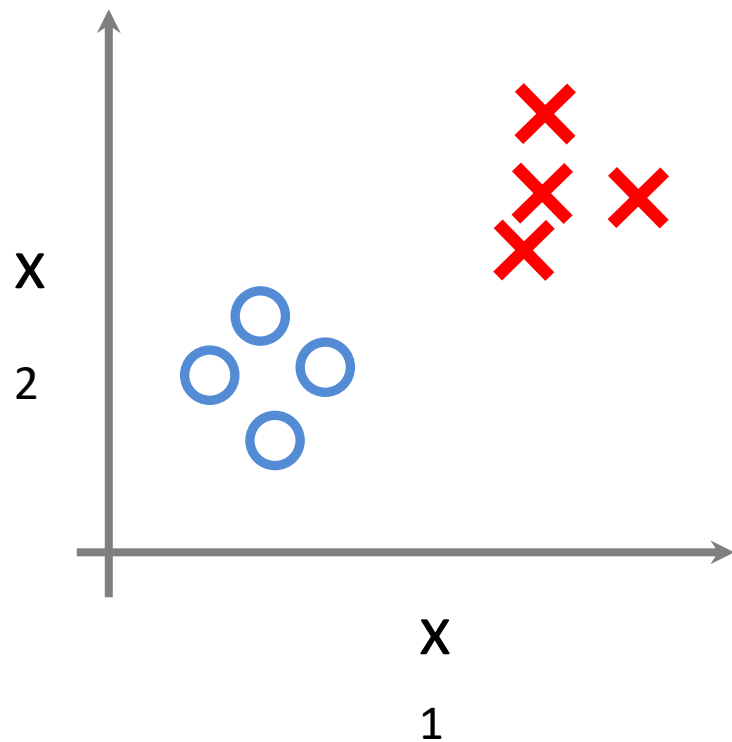
# Множественная классификация

Классификация сообщений: Рабочие, Личные, Семейные...

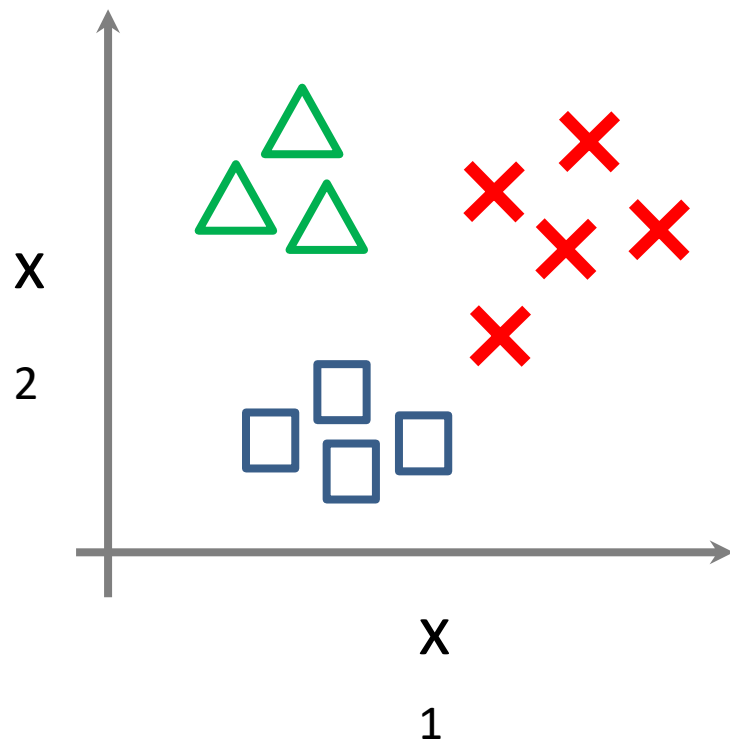
Диагностика: Здоров, Грипп, Простуда

Погода: Ясно, Пасмурно, Дождь, Снег

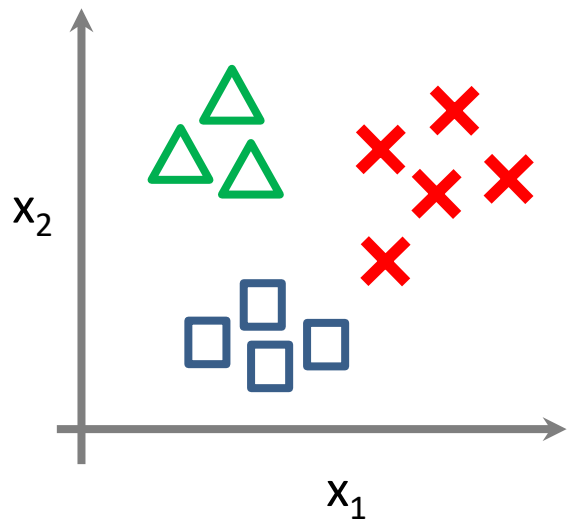
Бинарная классификация:




Множественная классификация:



## Один против всех (one-vs-rest):

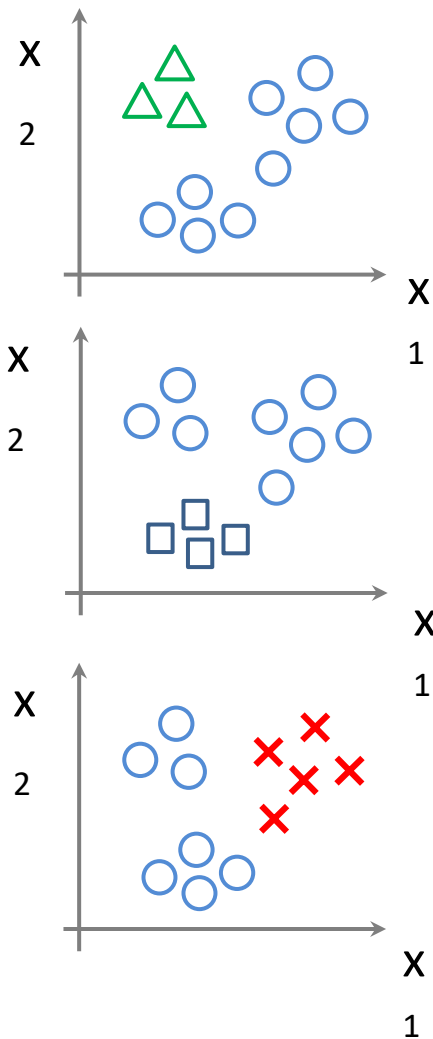


Класс 1: 

Класс 2: 

Класс 3: 

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$



## Один против всех

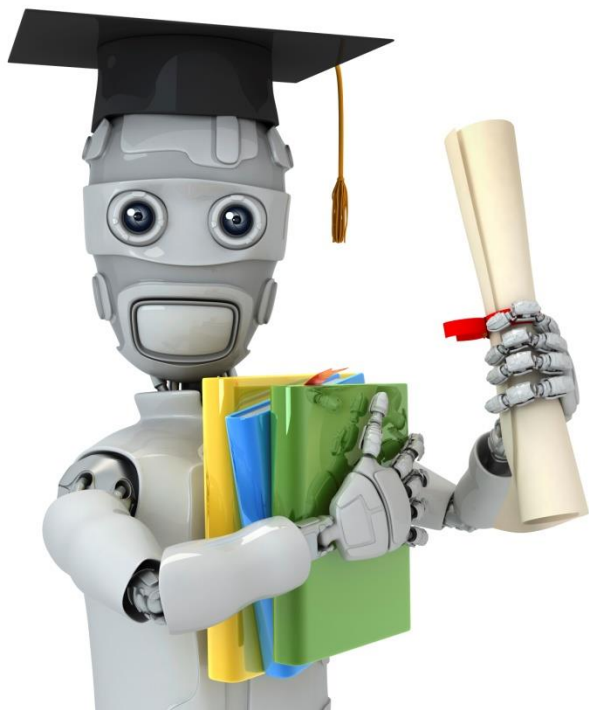
Обучить логистическую регрессию  $h_{\theta}^{(i)}(x)$  для каждого класса  $i$  чтобы предсказывать вероятность  $y = i$ .

При новом  $x$ , для предсказания, выбрать такой класс  $i$ , у которого:

$$\max_i h_{\theta}^{(i)}(x)$$

## Выводы:

1. Существуют методы классификации, которые сами по себе могут решать задачи множественной классификации.
2. Для тех, которые не умеют, существует алгоритм “один против всех”.
3. В нем строится столько бинарных моделей, сколько классов существует в задаче.
4. Данный алгоритм уже не зависит от выбора порогового значения.
5. Этот алгоритм еще может решать проблемы мультиклассификации.
6. Для задач с очень большим количеством классов этот алгоритм может быть неэффективен.



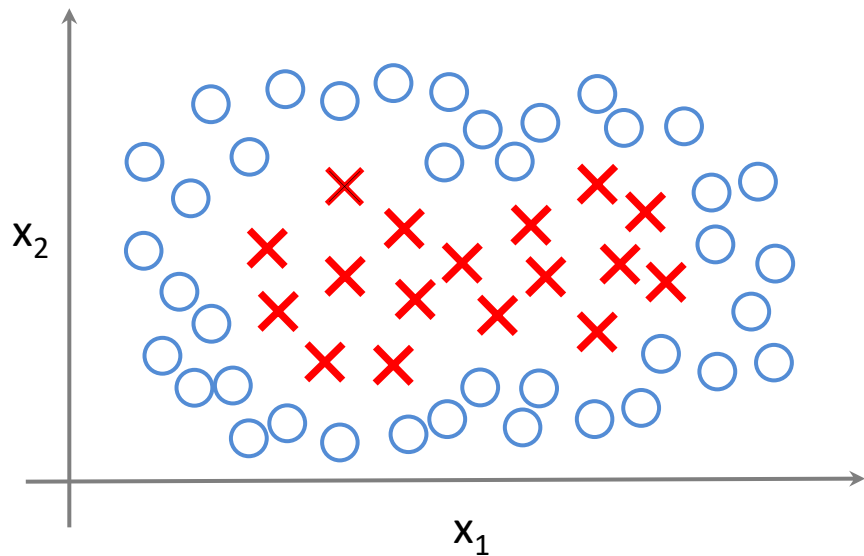
Machine Learning

# Метод опорных векторов

---

## Ядра

## Нелинейные границы принятия решений



Predict  $y = 1$  if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots \geq 0$$

Есть ли другой, более эффективный способ задать  $f_1, f_2, f_3, \dots$  ?

## Ядра и расстояния

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp \left( -\frac{\|x - l^{(1)}\|^2}{2\sigma^2} \right) = \exp \left( -\frac{\sum_{j=1}^n (x_j - l_j^{(1)})^2}{2\sigma^2} \right)$$

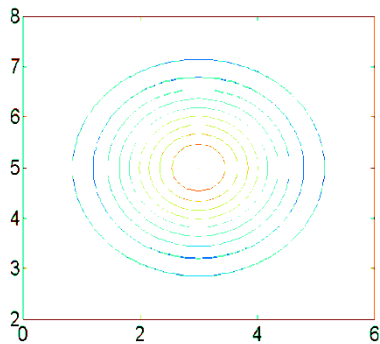
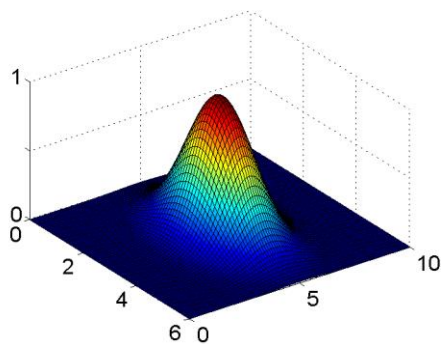
При данном  $x$ , вычислить новые признаки,  
основанные на близости к реперным точкам



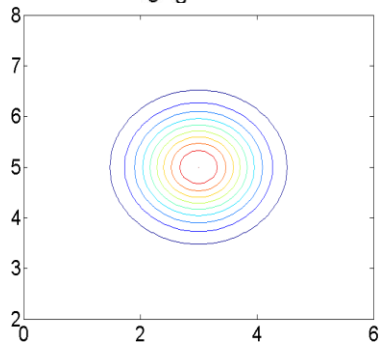
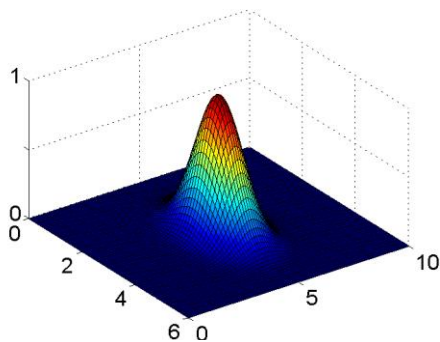
**Пример:**

$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp \left( -\frac{\|x - l^{(1)}\|^2}{2\sigma^2} \right)$$

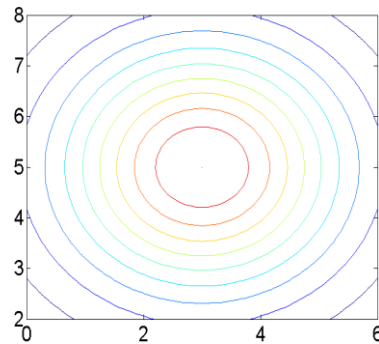
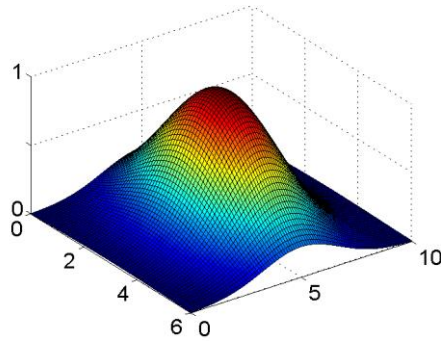
$$\sigma^2 = 1$$

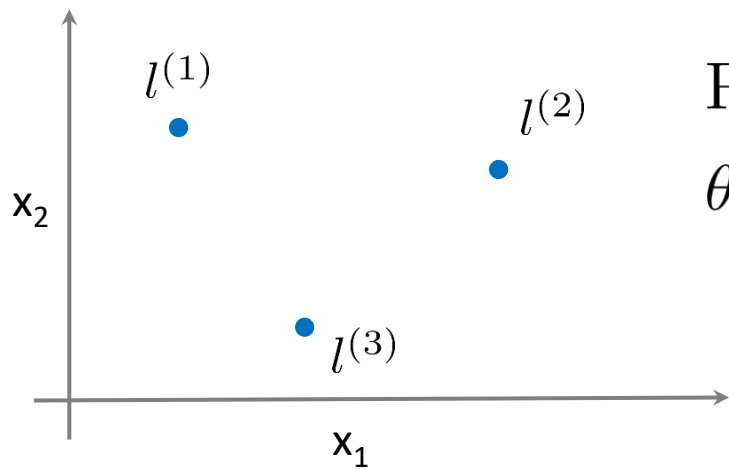


$$\sigma^2 = 0.5$$



$$\sigma^2 = 3$$

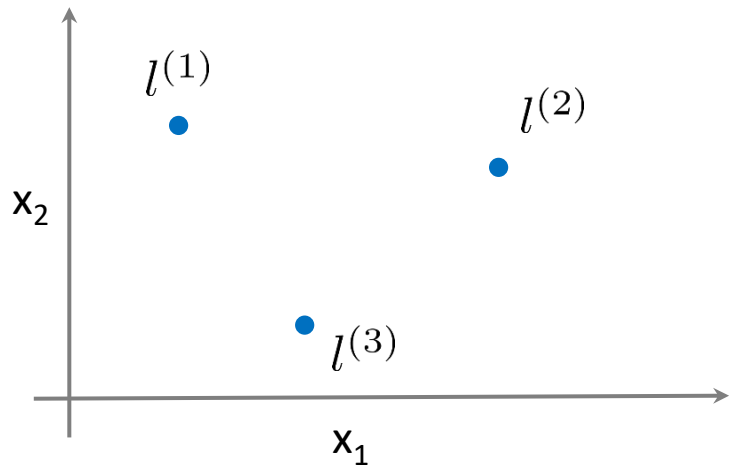




Predict “1” when

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

## Выбор реперных точек

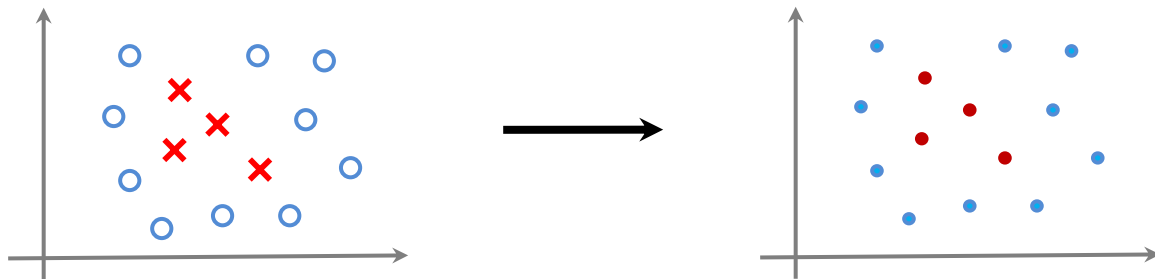


Given  $x$ :

$$f_i = \text{similarity}(x, l^{(i)})$$
$$= \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$$

Predict  $y = 1$  if  $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Where to get  $l^{(1)}, l^{(2)}, l^{(3)}, \dots$ ?



## SVM с ядром

Given  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$ ,  
choose  $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$ .

Given example  $x$ :

$$f_1 = \text{similarity}(x, l^{(1)})$$

$$f_2 = \text{similarity}(x, l^{(2)})$$

...

For training example  $(x^{(i)}, y^{(i)})$ :

## SVM с ядром

Модель: При данном  $x$  , вычислять признаки  $f \in \mathbb{R}^{m+1}$

Вывод: “ $y=1$ ” при  $\theta^T f \geq 0$

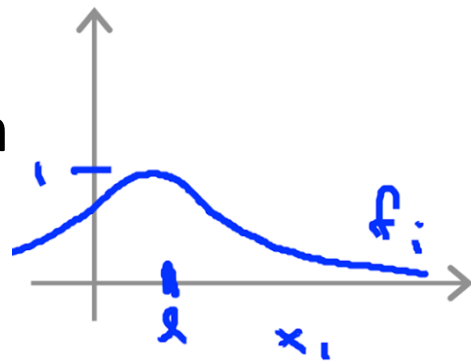
Обучение:

$$\min_{\theta} C \sum_{i=1}^m y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

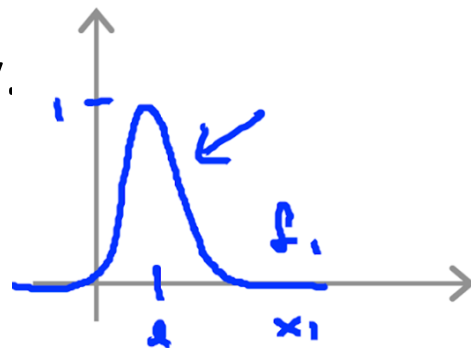
## Параметры SVM:

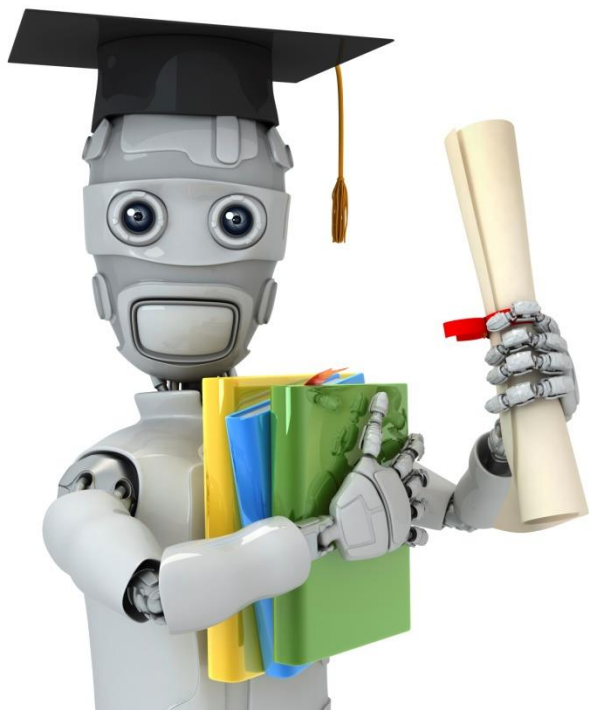
$C \left( = \frac{1}{\lambda} \right)$ . Large C: Lower bias, high variance.  
Small C: Higher bias, low variance.

Large  $\sigma^2$ : Features  $f_i$  vary more smooth  
Higher bias, lower variance.



Small  $\sigma^2$ : Features  $f_i$  vary less smoothly.  
Lower bias, higher variance.





Machine Learning

# Метод опорных векторов

---

Использование SVM

При обучении модели опорных векторов происходит подбор значений параметров  $\theta$  .

Необходимо указать:

Параметр регуляризации  $C$ .

Вид ядра (функцию расстояния):

SVM без ядра (линейное ядро)

Вывод “ $y = 1$ ” при  $\theta^T x \geq 0$

Гауссово ядро:

$$f_i = \exp \left( -\frac{\|x - l^{(i)}\|^2}{2\sigma^2} \right), \text{ где } l^{(i)} = x^{(i)}$$

Нужно выбрать  $\sigma^2$ .



**Функции ядра (расстояния):**

```
function f = kernel(x1,x2)
```

$$f = \exp \left( -\frac{||\mathbf{x1} - \mathbf{x2}||^2}{2\sigma^2} \right)$$

```
return
```

Замечание: при использовании гауссовых ядер необходимо предварительно нормализовать шкалы признаков

## Other choices of kernel

Не любая функция расстояния  $\text{similarity}(x, l)$  может быть функцией ядра (нужно удовлетворять “теореме Мерсера”, которая гарантирует, что методы оптимизации SVM будут сходиться).

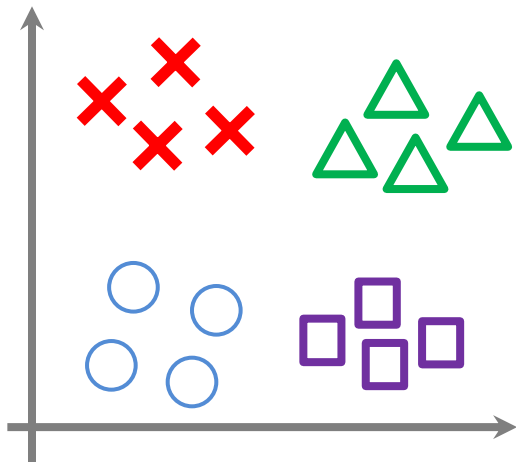
Существует множество готовых функций ядра:

- Полиномиальное ядро:

$$K(x, y) = (x^T y + c)^d$$

- Гауссовы ядра также называются RBF- ядра
- Более специализированные: строковые ядра, хи-квадрат, пересечение гистограмм, ...

## Множественная классификация



$$y \in \{1, 2, 3, \dots, K\}$$

Большинство готовых решений включают в себя реализацию множественной классификации.

Но можно использовать метод one-vs-all. (Обучить  $K$  моделей SVM, каждую для отделения класса  $1, 2, \dots, K$  от остальных) и выбирать класс  $i$ , для которого  $(\theta^{(i)})^T x$  выше.

## SVM и логистическая регрессия

$n$  = количество признаков ( $x \in \mathbb{R}^{n+1}$ ),  $m$  = количество точек.

При большом  $n$  (по сравнению с  $m$ ):

Лучше использовать логистическую регрессию или SVM без ядра

Если  $n$  мало, а  $m$  среднее:

Использовать SVM с гауссовым ядром

Если  $n$  мало, а  $m$  велико:

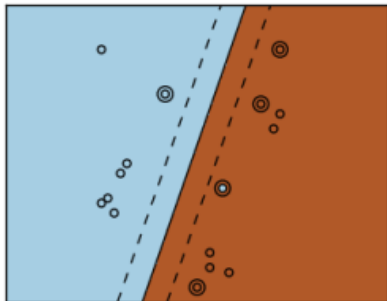
Собрать/создать больше признаков, а затем использовать логистическую регрессию или SVM без ядра

Нейронные сети обычно работают неплохо во всех случаях, но зачастую более трудоемки в обучении.

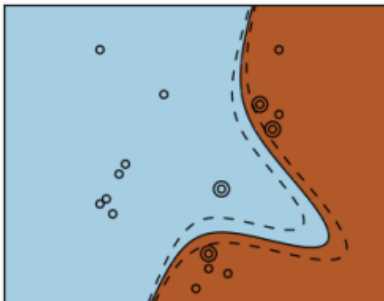
Гиперплоскость в спрямляющем пространстве соответствует нелинейной разделяющей поверхности в исходном.

Примеры с различными ядрами  $K(x, x')$

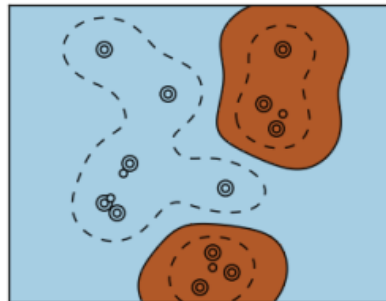
линейное  
 $\langle x, x' \rangle$

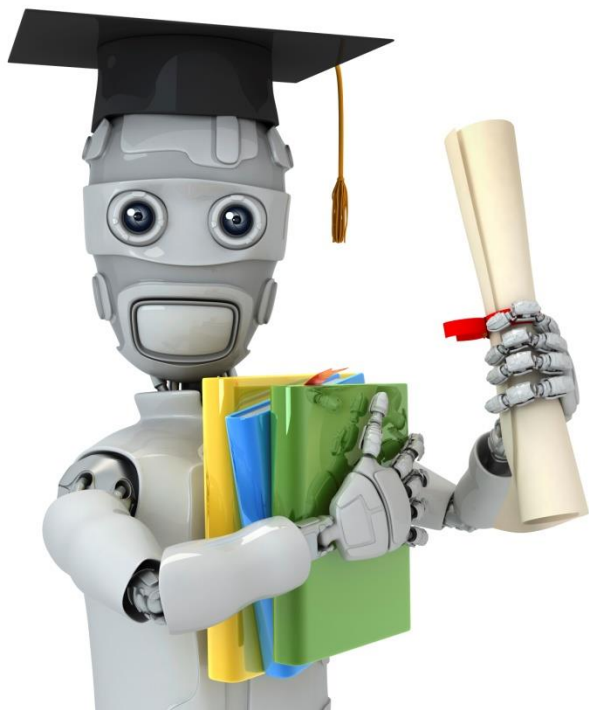


полиномиальное  
 $(\langle x, x' \rangle + 1)^d, d=3$



гауссовское (RBF)  
 $\exp(-\gamma \|x - x'\|^2)$





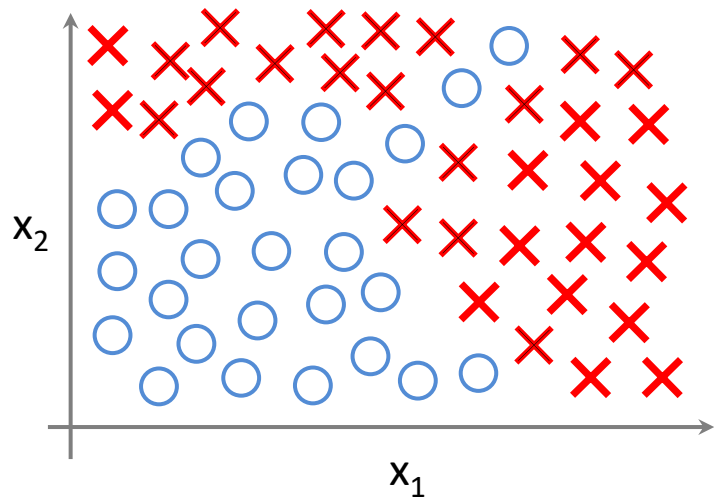
Machine Learning

# Neural Networks: Representation

---

## Non-linear hypotheses

## Non-linear Classification



$x_1$  = size

$x_2$  = # bedrooms

$x_3$  = # floors

$x_4$  = age

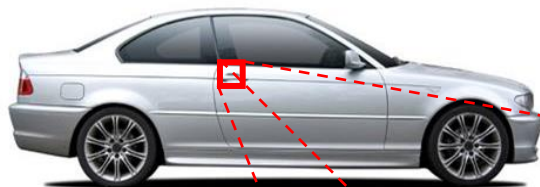
...

$x_{100}$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

# What is this?

You see this:



But the camera sees

this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

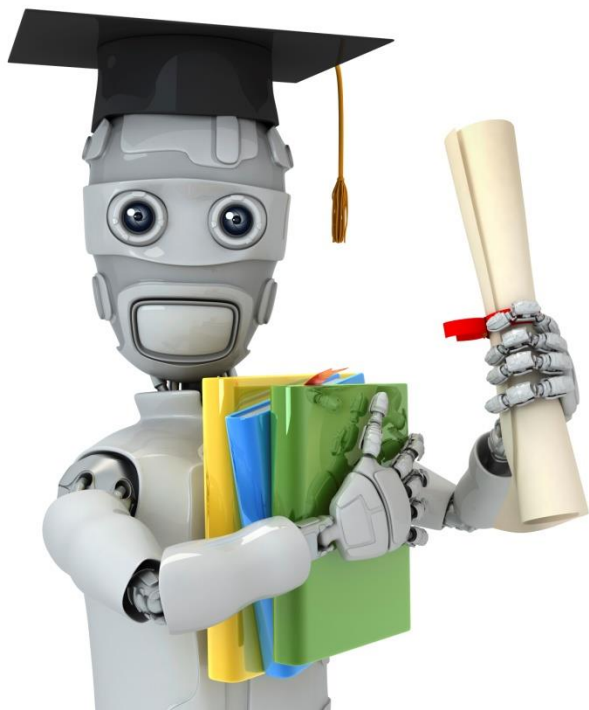


# Neural Networks

Origins: Algorithms that try to mimic the brain.

Was very widely used in 80s and early 90s; popularity diminished in late 90s.

Recent resurgence: State-of-the-art technique for many applications



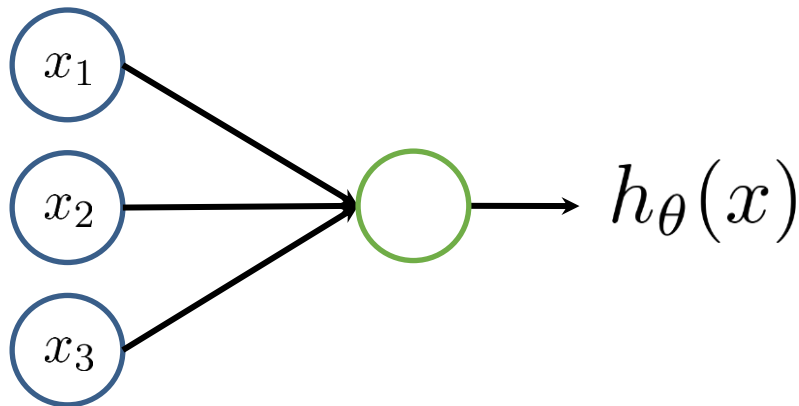
Machine Learning

# Neural Networks: Representation

---

## Model representation I

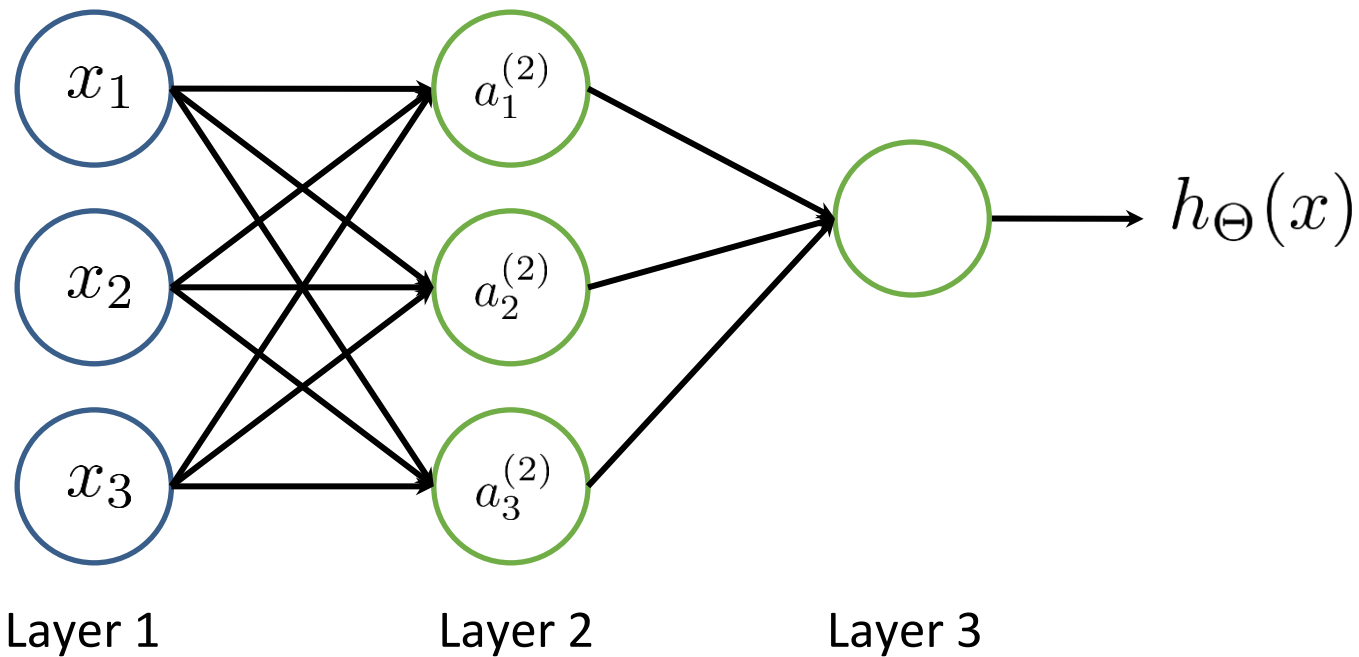
## Neuron model: Logistic unit



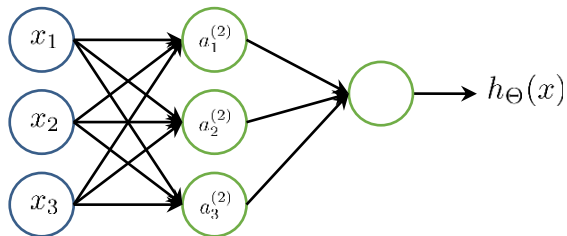
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

Sigmoid (logistic) activation function.

# Neural Network



# Neural Network



$a_i^{(j)}$  = “activation” of unit  $i$  in layer  $j$

$\Theta^{(j)}$  = matrix of weights controlling function mapping from layer  $j$  to layer  $j + 1$

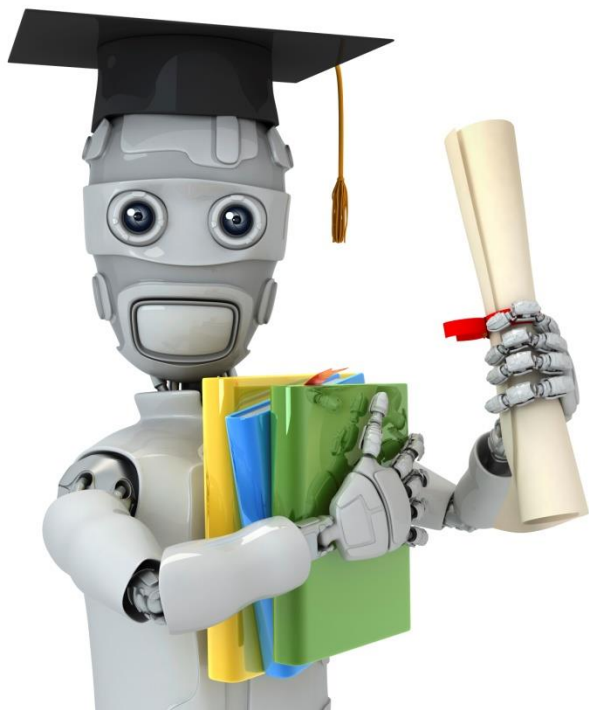
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has  $s_j$  units in layer  $j$ ,  $s_{j+1}$  units in layer  $j + 1$ , then  $\Theta^{(j)}$  will be of dimension  $s_{j+1} \times (s_j + 1)$ .



Machine Learning

# Neural Networks: Representation

---

## Multi-class classification

## Multiple output units: One-vs-all.



Pedestrian



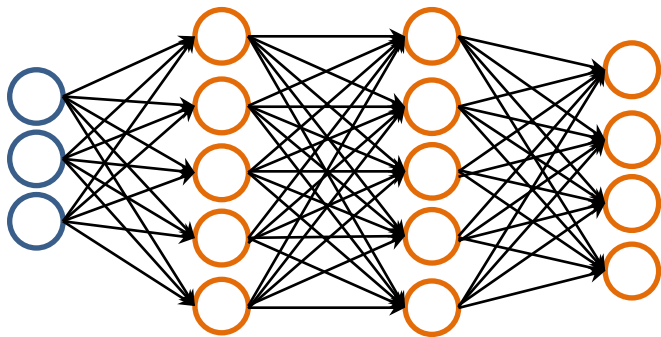
Car



Motorcycle



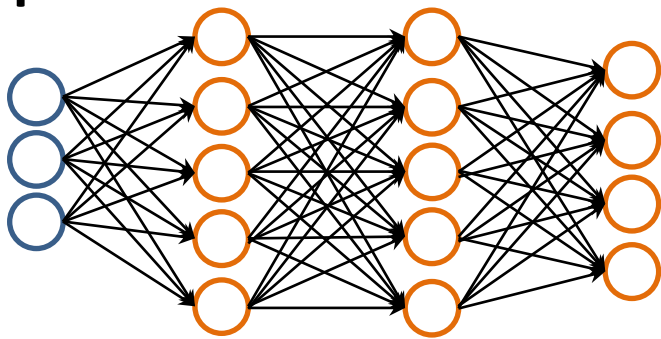
Truck



$$h_{\Theta}(x) \in \mathbb{R}^4$$

Want  $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ , etc.  
when pedestrian      when car      when motorcycle

## Multiple output units: One-vs-all.



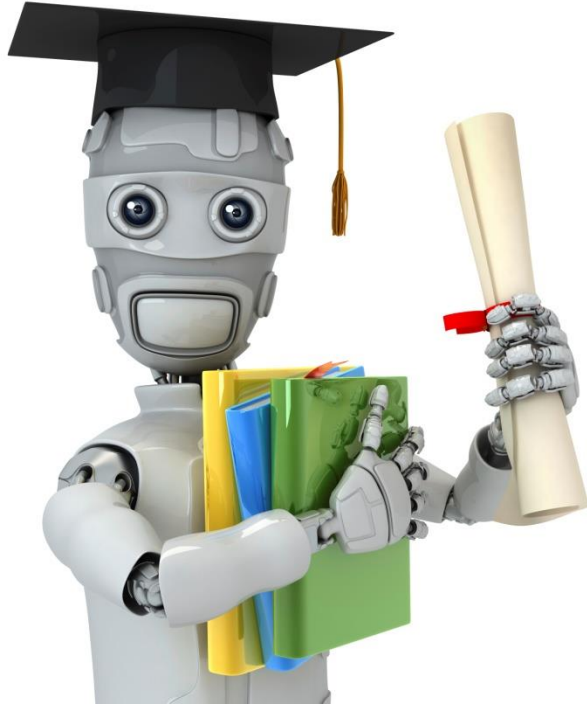
$$h_{\Theta}(x) \in \mathbb{R}^4$$

Want  $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ , etc.  
when pedestrian      when car      when motorcycle

Training set:  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$y^{(i)}$  one of  $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$   
pedestrian   car   motorcycle   truck





Machine Learning

# Neural Networks: Learning

---

## Backpropagation algorithm

## Gradient computation

$$J(\Theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log h_{\theta}(x^{(i)})_k + (1 - y_k^{(i)}) \log(1 - h_{\theta}(x^{(i)})_k) \right] \\ + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_j^{(l)})^2$$

$$\min_{\Theta} J(\Theta)$$

Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

# Gradient computation

Given one training example  $(x, y)$ :

Forward propagation:

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

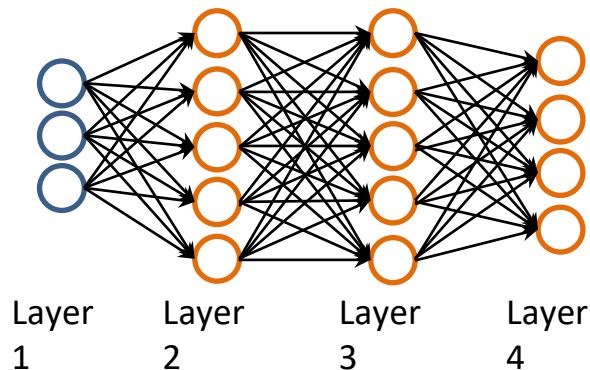
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$

$$z^{(4)} = \Theta^{(3)} a^{(3)}$$

$$a^{(4)} = h_{\Theta}(x) = g(z^{(4)})$$



# Gradient computation: Backpropagation algorithm

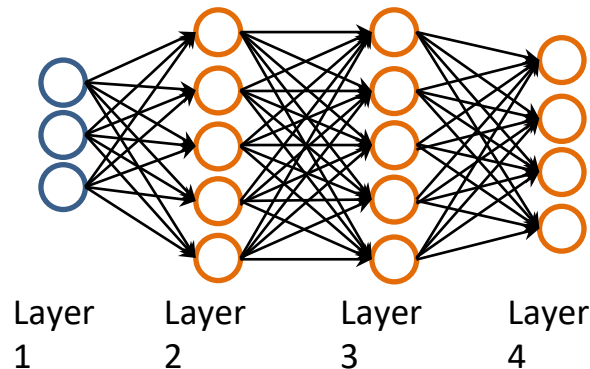
Intuition:  $\delta_j^{(l)}$  = “error” of node  $j$  in layer  $l$ .

For each output unit (layer  $L = 4$ )

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} \cdot * g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} \cdot * g'(z^{(2)})$$



## Backpropagation algorithm

Training set  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set  $\Delta_{ij}^{(l)} = 0$  (for all  $l, i, j$ ).

For  $i = 1$  to  $m$

Set  $a^{(1)} = x^{(i)}$

Perform forward propagation to compute  $a^{(l)}$  for  $l = 2, 3, \dots, L$

Using  $y^{(i)}$ , compute  $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute  $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \text{ if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \text{ if } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

# What is backpropagation doing?

$$J(\Theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \log(h_{\Theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))) \right] \\ + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

Focusing on a single example  $x^{(i)}$ ,  $y^{(i)}$ , the case of 1 output unit, and ignoring regularization ( $\lambda = 0$ ),

$$\text{cost}(i) = y^{(i)} \log h_{\Theta}(x^{(i)}) + (1 - y^{(i)}) \log h_{\Theta}(x^{(i)})$$

(Think of  $\text{cost}(i) \approx (h_{\Theta}(x^{(i)}) - y^{(i)})^2$ )

I.e. how well is the network doing on example  $i$ ?

# МЕТОД К БЛИЖАЙШИХ СОСЕДЕЙ

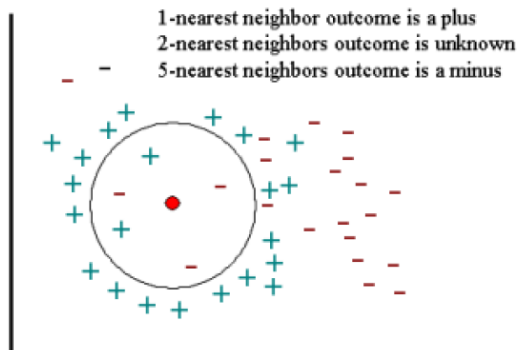
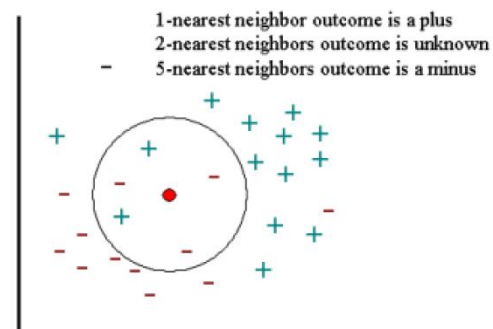
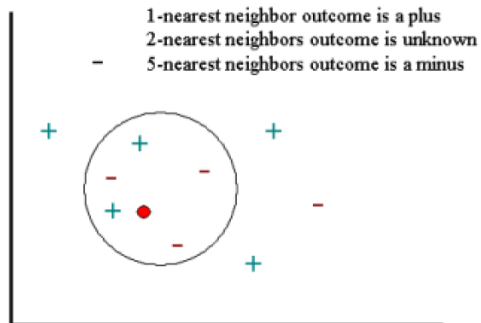


Метод  $k$  ближайших соседей (kNN —  $k$  nearest neighbours) метрический алгоритм для классификации объектов, основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.

### Алгоритм:

- 1 Вычислить расстояние до каждого из объектов обучающей выборки
- 2 Отобрать  $k$  объектов обучающей выборки, расстояние до которых минимально
- 3 Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди  $k$  ближайших соседей





## Достоинства:

- Простота реализации.
- Классификацию, проведенную алгоритмом, легко интерпретировать путем предъявления пользователю нескольких ближайших объектов.

## Недостатки:

- Необходимость хранения обучающей выборки целиком.
- Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки.

- Малые значения  $k$  приведут к тому, что “шум” (выбросы) будет существенно влиять на результаты.
- Большие значения усложняют вычисления и искажают логику ближайших соседей, в соответствии с которой ближайшие точки могут принадлежать одному классу (гипотеза компактности).
- Эвристика:  $k = \sqrt{n}$

# Деревья решений



*Деревья решений – это способ представления правил в иерархической, последовательной структуре, где каждому объекту соответствует единственный узел, дающий решение*

*Деревья решений – это логический алгоритм классификации, основанный на поиске конъюнктивных закономерностей.*



	возраст	наличие дома	доход	образование	кредит
$x_1$	32	нет	2000	среднее	нет
$x_2$	54	да	12000	высшее	да
$x_3$	73	нет	800	специальное	нет
...	...	...			
$x_{50}$	18	да	200	среднее	да



- 1 “Построение” или “создание” дерева (tree building):  
выбор **критерия расщепления** и **остановки**  
обучения
- 2 “Сокращение” дерева (tree pruning): **сокращения**  
дерева и **отсечение** некоторых его ветвей



- Расщепление должно разбивать исходное множество данных таким образом, чтобы объекты подмножеств, получаемых в результате этого разбиения, являлись представителями одного класса или же были максимально приближены к такому разбиению.
- Количество объектов из других классов, так называемых “примесей”, в каждом классе должно стремиться к минимуму.

