Лаборатория компьютерной графики и мультимедиа ВМК МГУ имени М.В. Ломоносова

Курс «Введение в компьютерное зрение и глубокое обучение»

Лекция №3
«Введение в машинное обучение»

Антон Конушин
Заведующий лабораторией компьютерной графики и мультимедиа ВМК МГУ

4 марта 2019 года

Классификация

Задача классификации – определить для объекта x его «класс» y из заданного конечного набора классов

Изображение x_1

Изображение x_2

Это лицо?

$y_1 = 1$ (да)

$y_2 = 0$ (нет)

Классификатор $f(x) = y$ – функция, отображающая x в соответствующий класс

Классификация изображений

Бинарная классификация

- На картинке лицо человека?
- Бинарный ответ

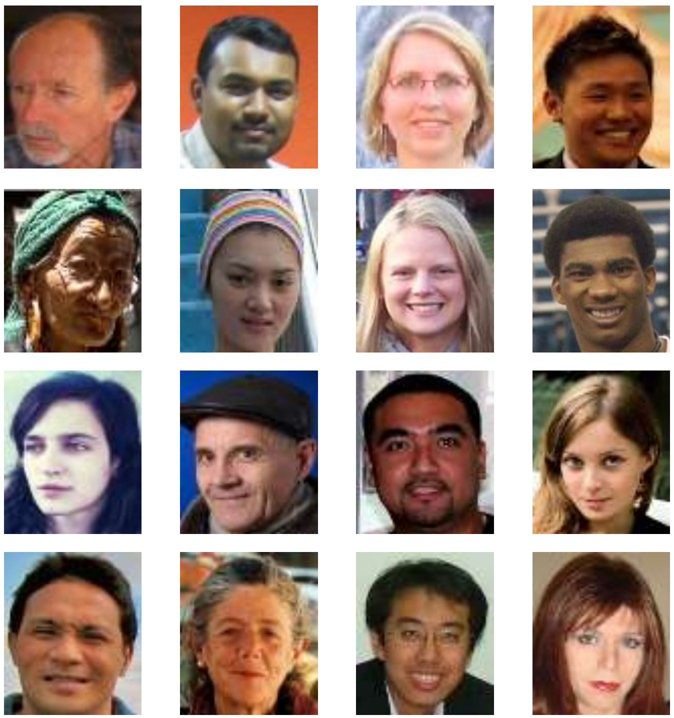
$$y \in [0,1], 1 \ - \text{да}, 0 \ - \text{нет}$$

Многоклассовая классификация

- К какому из заданных s классов относится данное изображение?
- Например этническое происхождение (раса) – европеец, азиат, негр и т.д.
- Ответ – метка класса $c \in [1, S]$

Что такое «лицо»?

- Соберём много примеров «лиц» и «не лиц»

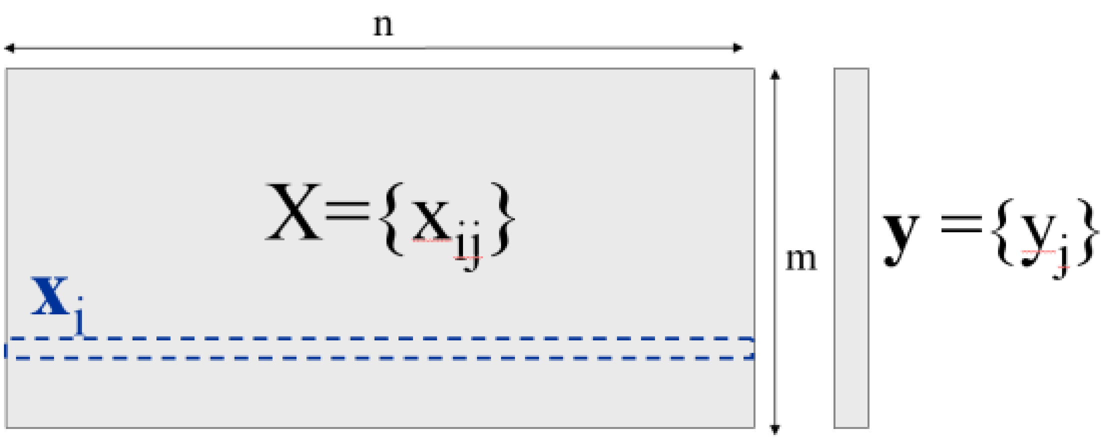


Лица

Не лица

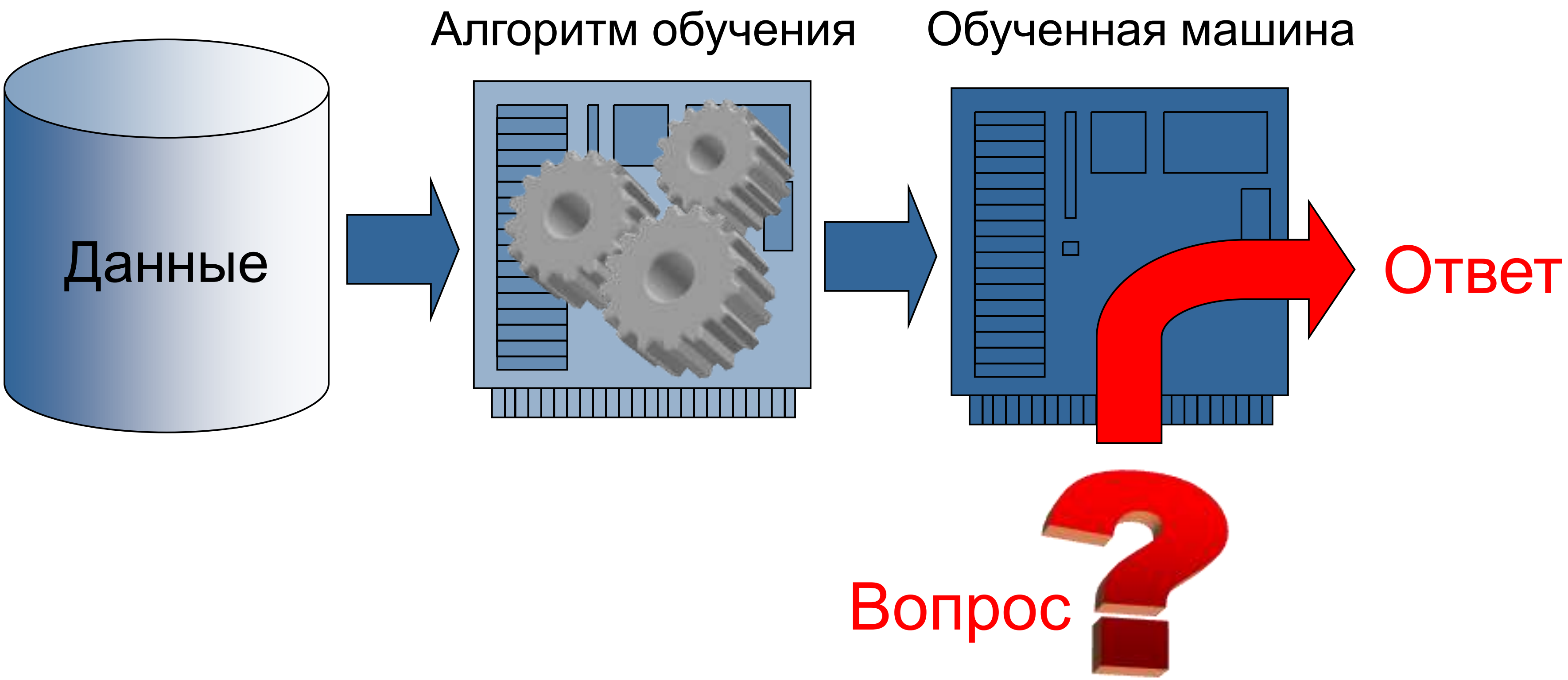
Построение классификатора

- Есть выборка данных X для которых известны Y
- Каждый объект из X с номером j можно описать вектором признаков $\boldsymbol{x}_j$ (набором характеристик)
- Всё множество известных наблюдений (конечное) можно записать в следующем виде (обучающая выборка):



Нужно построить функцию f(x) = y

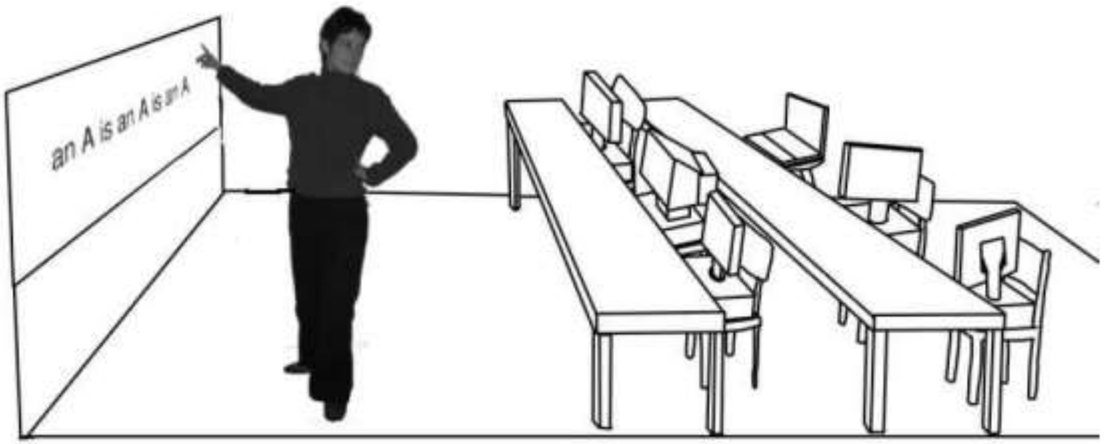
Как должно быть в идеале



Машинное обучение

Что такое машинное обучение?

- Обучение ≠ «заучивание наизусть»
  - Заучить наизусть – для машины не проблема
  - Мы хотим научить машину делать выводы!
  - Машина должна корректно работать на новых данных, которые мы ей раньше не давали
  - По конечному набору обучающих данных машина должна научиться делать выводы на новых данных



Машинное обучение

Развитие методов компьютерного зрения неразрывно связано с развитием методов машинного обучения

- Нейронные сети - Перспептрон (Розенблатт, 1958), когнитрон (Фукушима, 1975)
- Нейронные сети - обратное распространение ошибки (1980е)
- Метод опорных векторов (1990е)
- Бустинг (конец 1990х)
- Рандомизированный решающий лес (начало 2000х)
- Нейронные сети - «глубинное обучение» (2006 и далее)

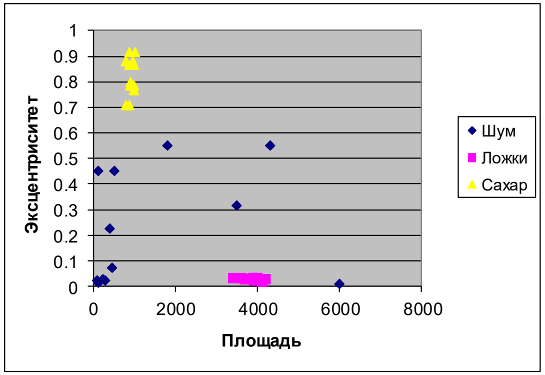
Планы на сегодня

Декомпозиция классификации изображений:

Изображение I

Признаки изображения → Анализ признаков → Ответ y

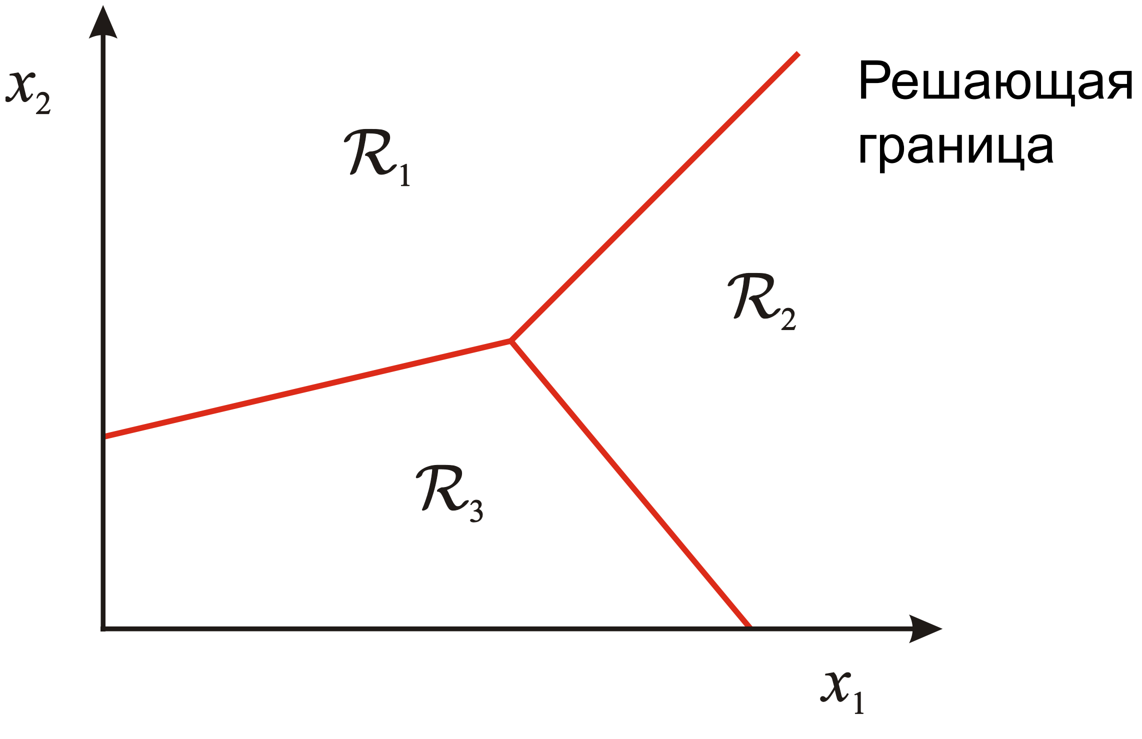
$$x = features\ (I) \quad y = analysis(x)$$



- Сегодня мы будем рассматривать вторую стадию - построение классификаторов
- На следующей лекции будем говорить про вектор-признаки и т.д.

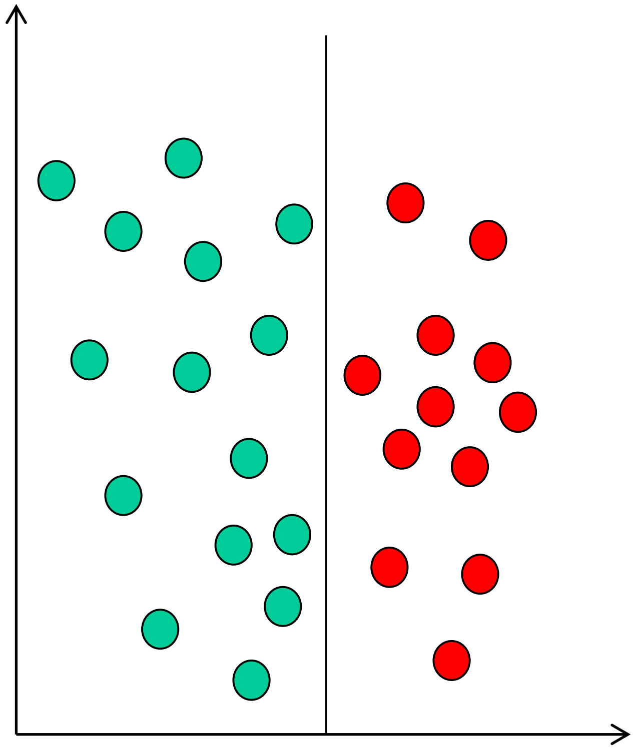
Задача классификации образов

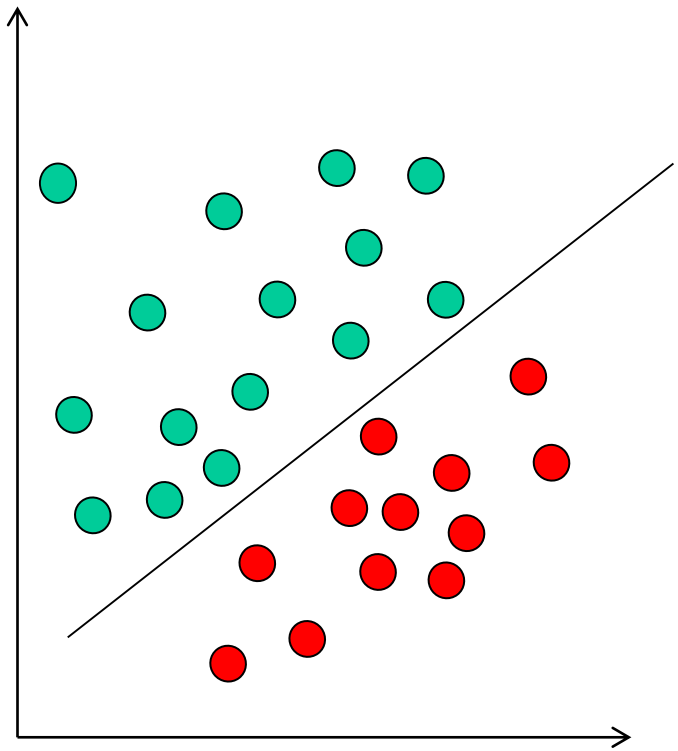
- Наша задача построить классификатор – функцию **y=f(x)**, которая для каждого вектора-признаков **x** даёт ответ **y**, какому классу принадлежит объект **x**
- Другое название - *решающее правило*
- Любое *решающее правило* делит пространство признаков на *решающие регионы* разделенные *решающими границами*



Пространство признаков

Простые классификаторы





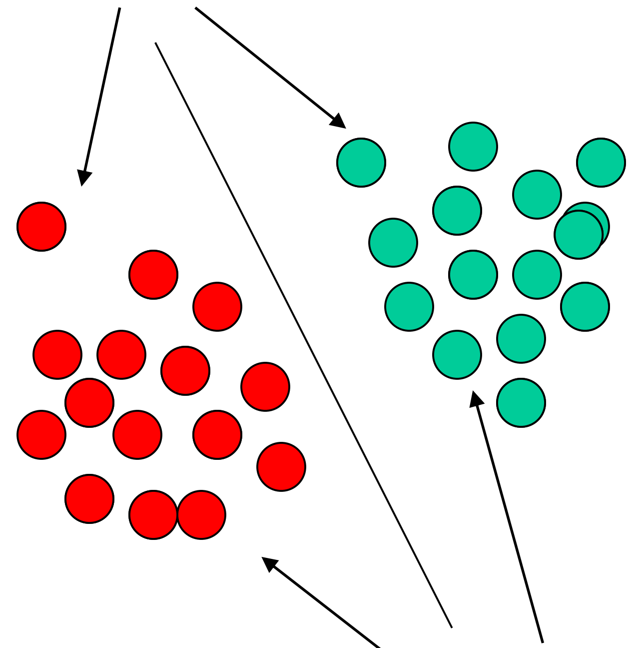
- Мы привели 2 примера параметрический семейств функций ***F,*** из которых мы будем искать конкретные наилучшие $\boldsymbol{f_t}$ (т.е. будем выбирать подходящий набор параметров ***t***)

Линейная классификация

- Данные являются линейно разделимыми, если их вектора признаков в пространстве признаков можно отделить друг от друга гиперплоскостью

- Т.е. существует такая гиперплоскость

$$f(x; w, b) = x \cdot \mathbf{w} + b$$

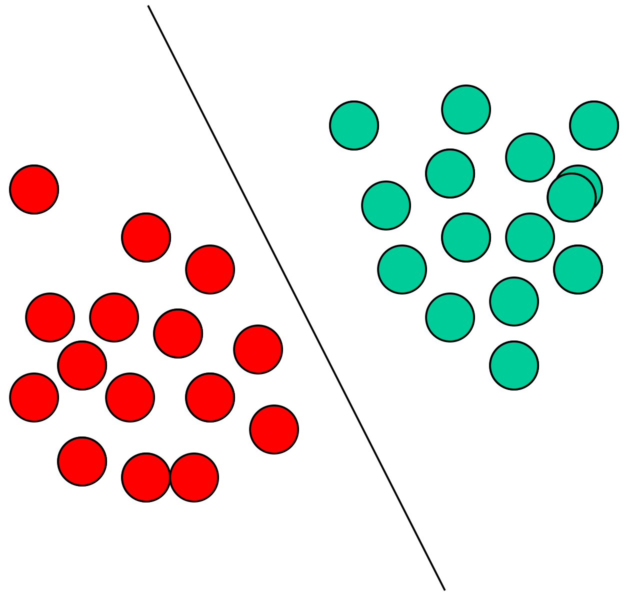


$$\mathbf{x}_i \; \textit{положительные} \; (y_i = 1): \qquad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

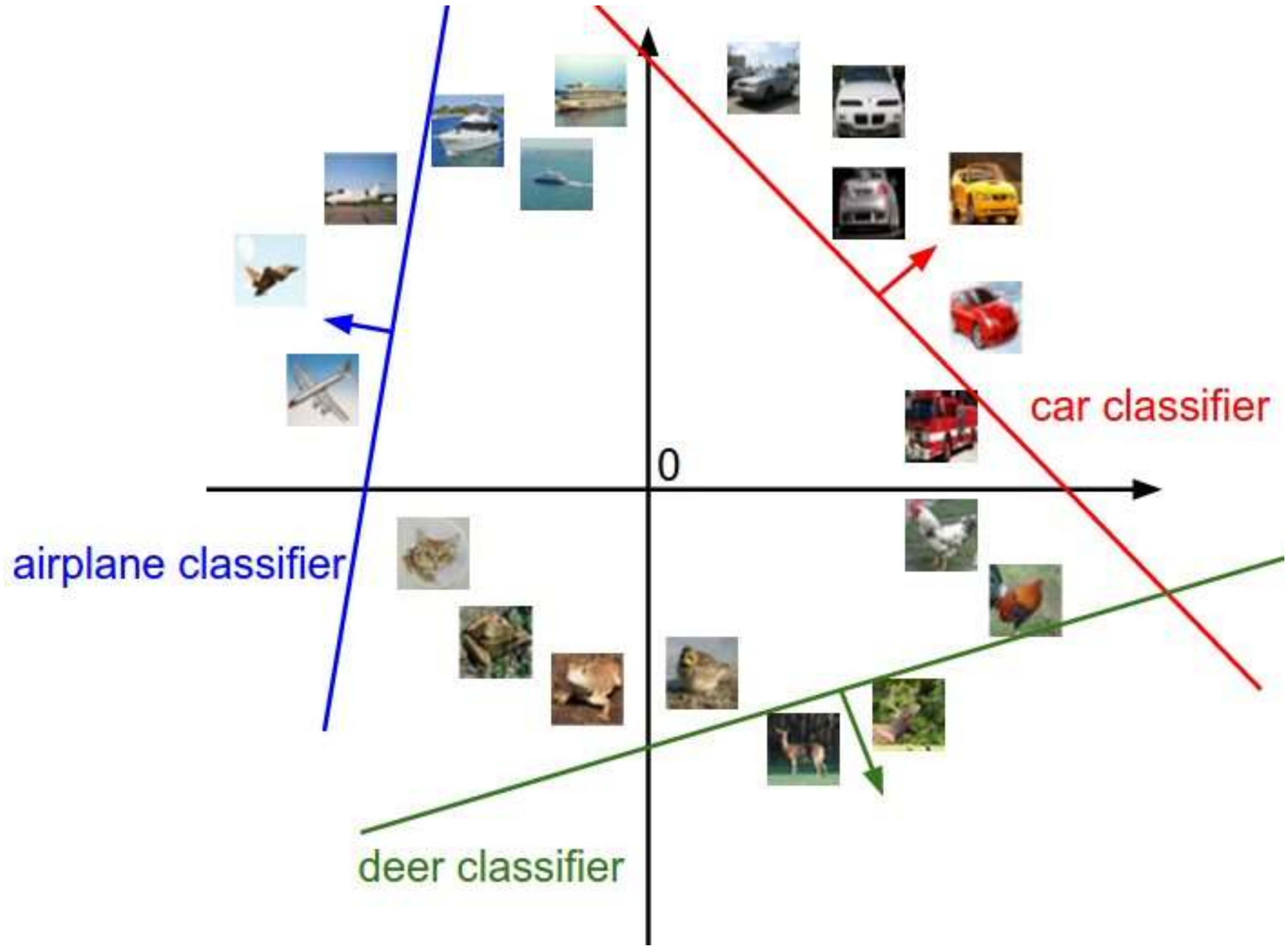
$$\mathbf{x}_i \; \textit{отрицательные} \; (y_i = -1): \qquad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Линейная классификация

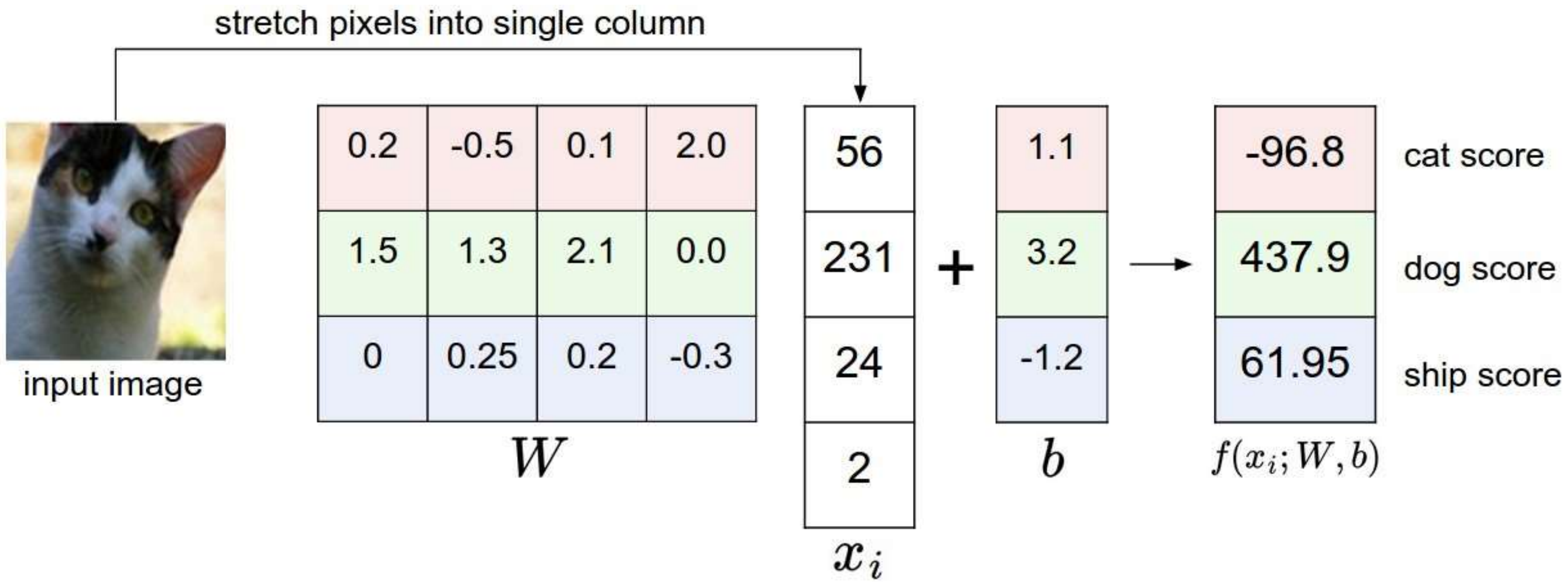
- Значение $f(x)$ от конкретного x называется *score*
- Часто обозначается как s
- Чем выше score, тем «увереннее» классификатор в своём предсказании
- В случае линейной классификации score обычно – расстояние до разделяющей гиперплоскости



Многоклассовая линейная классификация

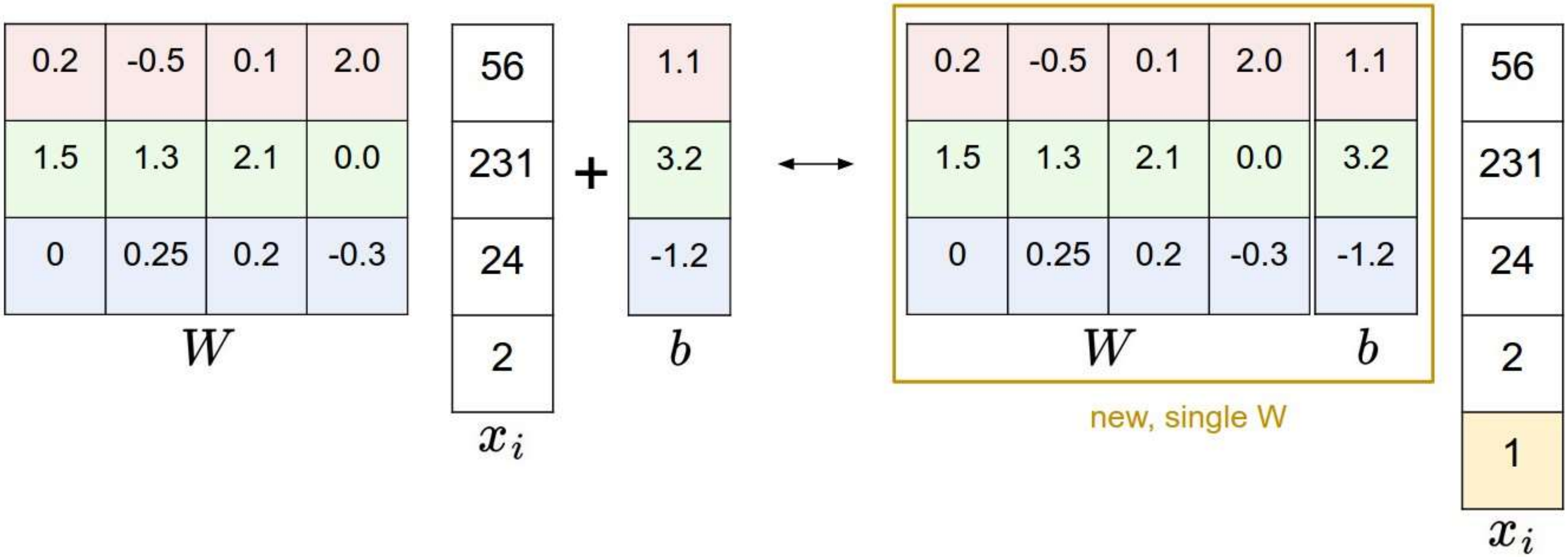


Пример



- В матрицу W мы объединяем все веса w для отдельных (бинарных) линейных классификаторов

Bias trick



- Можем упрятать параметр b в матрицу W за счет добавления 1 в конец вектор-признака x_i
- Получаем классификатор f(x;W)

Как будем искать?

Функция потерь

- Введем некоторую **функцию потерь** ***$L(f_w(x,w), y)$***, где (x,y) – обучающая выборка, w – параметры классификатора («веса»)
  - В случае классификации часто используют $L(f(\mathbf{x}), y) = I[y \neq f(\mathbf{x})]$,где $f(\mathbf{x})$ - предсказанный класс
  - Можем использовать и другие функции потерь
- Задача обучения состоит в том, чтобы найти набор параметров **w** классификатора **f** («весов»), при котором суммарные потери для *новых* данных будут минимальны
- «Метод классификации» = параметрическое семейство F + алгоритм оценки параметров классификатора по обучающей выборке {X,Y}

Общий риск

- Общий риск – математическое ожидание потерь:

$$R(x) = E\big(L(f(x,w),y)\big) = \int_{x,y} L(f(x,w),y)dP$$

- Наша задача – найти такие параметры w, при которых общий риск минимален
- Но общий риск рассчитать невозможно, поскольку распределение *P* для множества наших объектов (x,y) неизвестно

Эмпирический риск

- Дано $X^m = \{\mathbf{x_1}, ..., \mathbf{x_m}\}$ - обучающая выборка

- Эмпирический риск (ошибка обучения) – сумма потерь («суммарный лосс») на обучающей выборке:

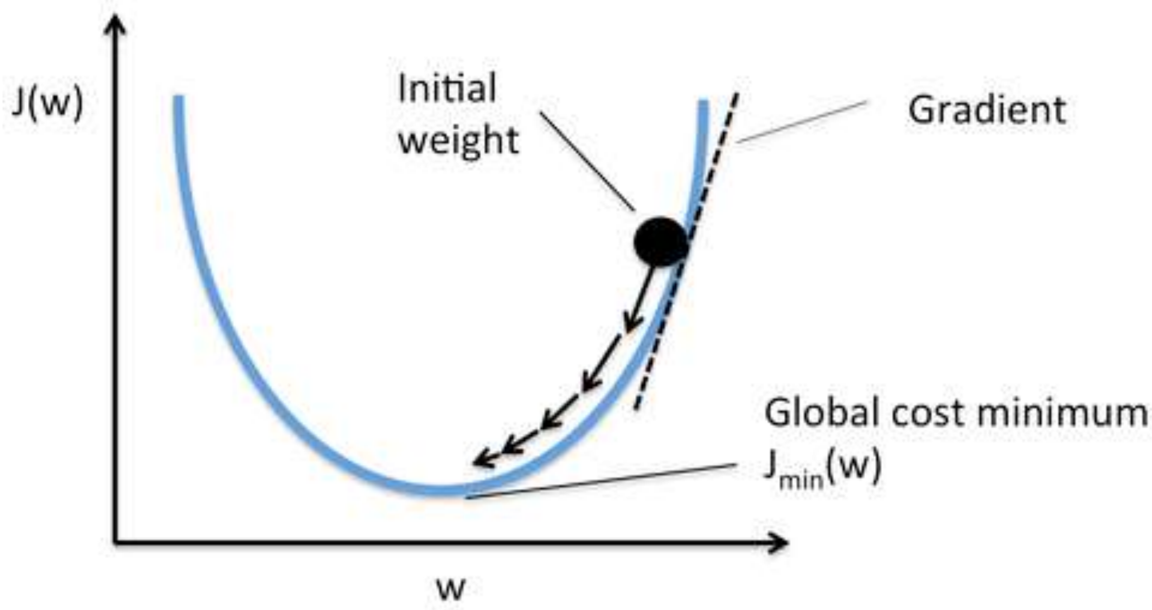
$$R_{emp}\left(f, X^m\right) = \frac{1}{m}\sum_{i=1}^{m} L(f(x_i), y_i)$$

- Метод минимизации эмпирического риска:

$$f = \arg\min_{f \in F} R_{emp}\left(f, X^m\right)$$

Смысл: подбираем параметры ***w*** классификатора ***f*** таким образом, чтобы эмпирический риск $\boldsymbol{R_{emp}}$ был минимален

Вариант: градиентный спуск



- Есть функция стоимости от параметров w, нужной найти параметры, при которых она достигает минимума
- Считаем градиент функции с точке начального приближения и сдвигаемся (w) в сторону уменьшения стоимости
- Повторяем до сходимости

Градиентный метод обучения

Минимизация аппроксимированного эмпирического риска:

$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\bigl(\langle w, x_i\rangle y_i\bigr) \to \min_w .$$

Численная минимизация методом *градиентного спуска*:

$w^{(0)} :=$ начальное приближение;

$$w^{(t+1)} := w^{(t)} - \eta \cdot \nabla Q(w^{(t)}), \qquad \nabla Q(w) = \left(\frac{\partial Q(w)}{\partial w_j}\right)_{j=0}^{n},$$

где η — *градиентный шаг*, называемый также *темпом обучения*.

$$w^{(t+1)} := w^{(t)} - \eta \sum_{i=1}^{\ell} \mathscr{L}'\bigl(\langle w^{(t)}, x_i\rangle y_i\bigr)\, x_i\, y_i.$$

Идея ускорения сходимости:

брать (x_i, y_i) по одному и сразу обновлять вектор весов.

SG: Стохастический градиентный спуск

Вход: выборка X^ℓ, темп обучения h, темп забывания λ;
Выход: вектор весов w;

инициализировать веса w_j, $j = 0, \dots, n$;
инициализировать оценку функционала: $\bar{Q} := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}_i(w)$;
повторять
    выбрать объект x_i из X^ℓ случайным образом;
    вычислить потерю: $\varepsilon_i := \mathscr{L}_i(w)$;
    сделать градиентный шаг: $w := w - h\nabla\mathscr{L}_i(w)$;
    оценить функционал: $\bar{Q} := (1 - \lambda)\bar{Q} + \lambda\varepsilon_i$;
пока значение $\bar{Q}$ и/или веса w не сойдутся;

Robbins, H., Monro S. A stochastic approximation method // Annals of Mathematical Statistics, 1951, 22 (3), p. 400–407.

Плюсы и минусы

Достоинства:

1. легко реализуется;
2. легко обобщается на любые $g(x, w)$, $\mathscr{L}(a, y)$;
3. возможно динамическое (потоковое) обучение;
4. на сверхбольших выборках можно получить неплохое решение, даже не обработав все (x_i, y_i);
5. подходит для задач с большими данными

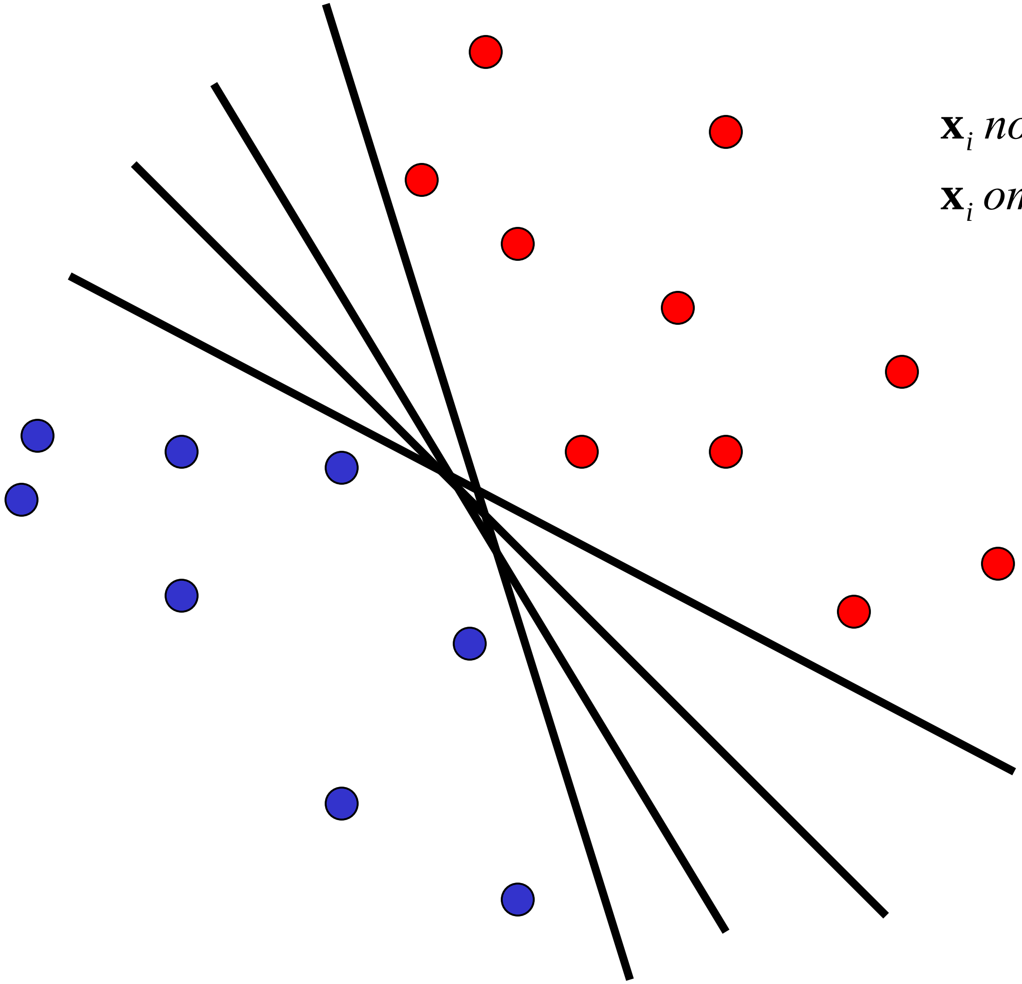
Недостатки:

1. возможна расходимость или медленная сходимость;
2. застревание в локальных минимумах;
3. подбор комплекса эвристик является искусством;
4. проблема переобучения;

Метод опорных векторов (SVM)

Линейный классификатор

- Найдем линейную функцию (гиперплоскость) и разделим положительные {y=+1} и отрицательные {y=-1} примеры



$\mathbf{x}_i$ *положительные*: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

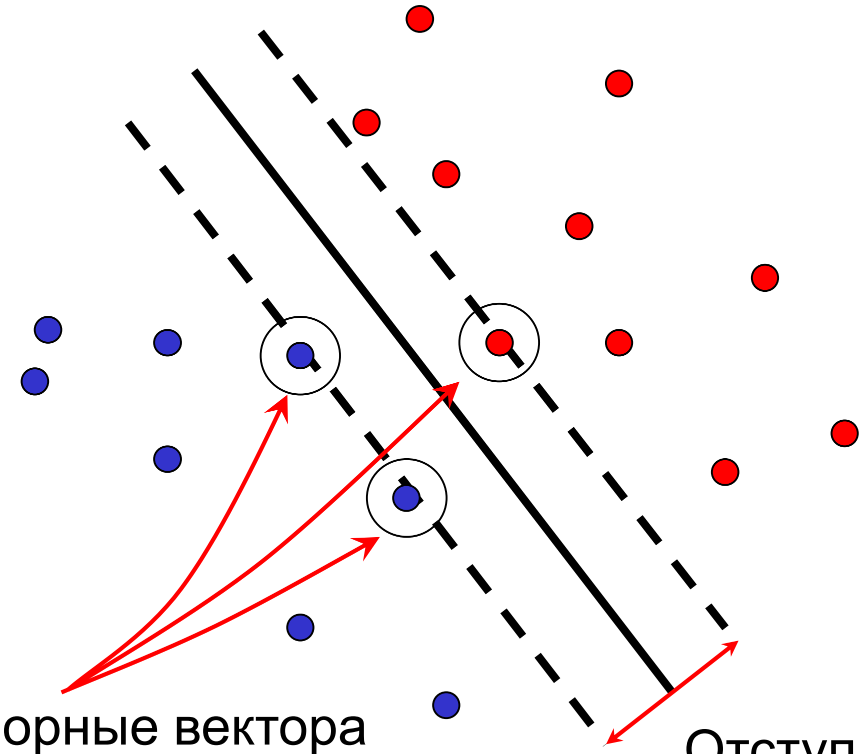
$\mathbf{x}_i$ *отрицательные*: $\quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$

Для всех рассмотренных плоскостей эмпирический риск одинаковый

Какая гиперплоскость наилучшая?

Метод опорных векторов

- Найдем гиперплоскость, максимизирующую *отступ* между положительными и отрицательными примерами
- Support Vector Machine (SVM)



$\mathbf{x}_i$ *положительные* $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ *отрицательные* $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

Для опорных векторов, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Расстояние от точки до гиперплоскости: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Поэтому отступ равен $2 / \|\mathbf{w}\|$

http://www.machinelearning.ru/wiki/index.php?title=SVM

Поиск гиперплоскости

1. Максимизируем отступ $2/\|\mathbf{w}\|$
2. Правильно классифицируем все данные:

$$\mathbf{x}_i \; \textit{позитивные} \; (y_i = 1): \qquad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \; \textit{негативные} \; (y_i = -1): \qquad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Квадратичная оптимизационная задача*:

- Минимизируем $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

  При условии $y_i(\mathbf{w}\cdot\mathbf{x}_i+b) \geq 1$

- Решается с помощью методом множителей Лагранжа

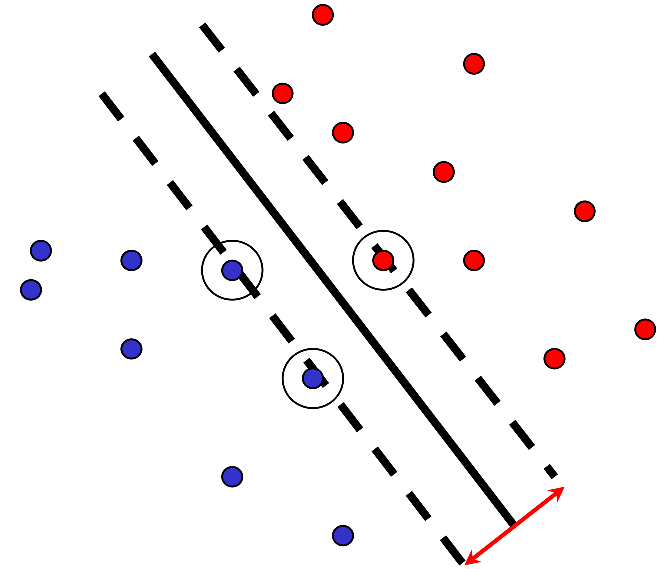
http://www.machinelearning.ru/wiki/index.php?title=SVM

Поиск гиперплоскости

- Решение: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Обученные веса

Опорные вектора



- Для большей части векторов вес = 0!
- Все вектора, для которых вес >0 называются опорными
- Определяется только опорными векторами

http://www.machinelearning.ru/wiki/index.php?title=SVM

Поиск гиперплоскости

- Решение: $$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ для любого опорного вектора

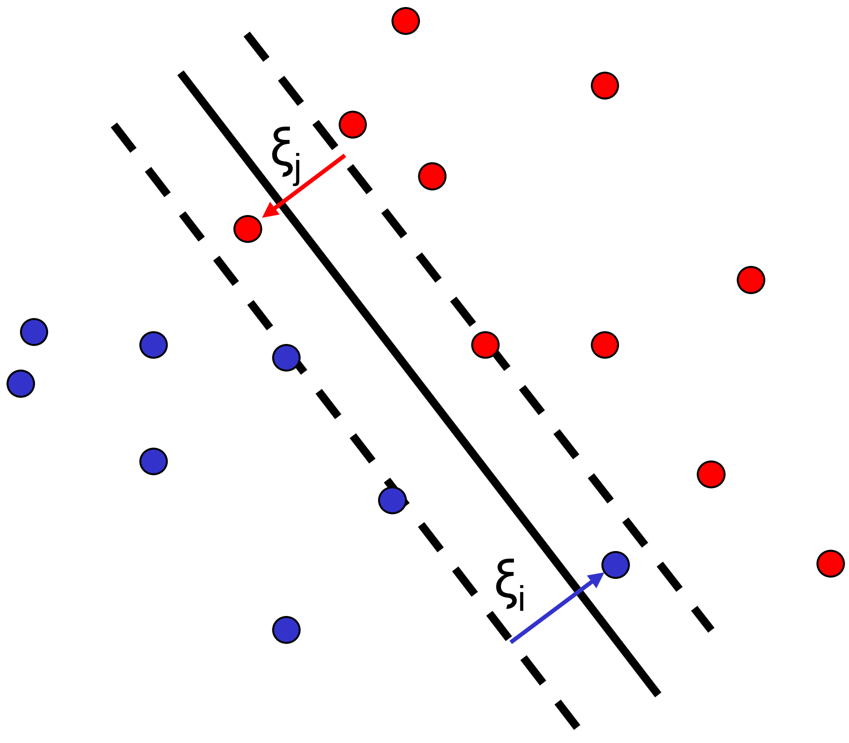
- Решающая функция:

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Решающая функция зависит от скалярных произведений (inner product) от тестового вектора $\boldsymbol{x}$ и опорных векторов $\boldsymbol{x}_i$
- Решение оптимизационной задачи также требует вычисления скалярных произведений $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ между всеми парами векторов из обучающей выборки

http://www.machinelearning.ru/wiki/index.php?title=SVM

Реальный случай



- Вводим дополнительные «slack» переменные ξ_i для каждой пары (x_i, y_i) так, чтобы:

$$y_i(wx_i + b) \geq 1 - \xi_i$$

- Если $0 \leq \xi_i \leq 1$ – то x_i классифицируется верно, но нарушается «зазор»
- Если $\xi_i > 1$ - ошибка

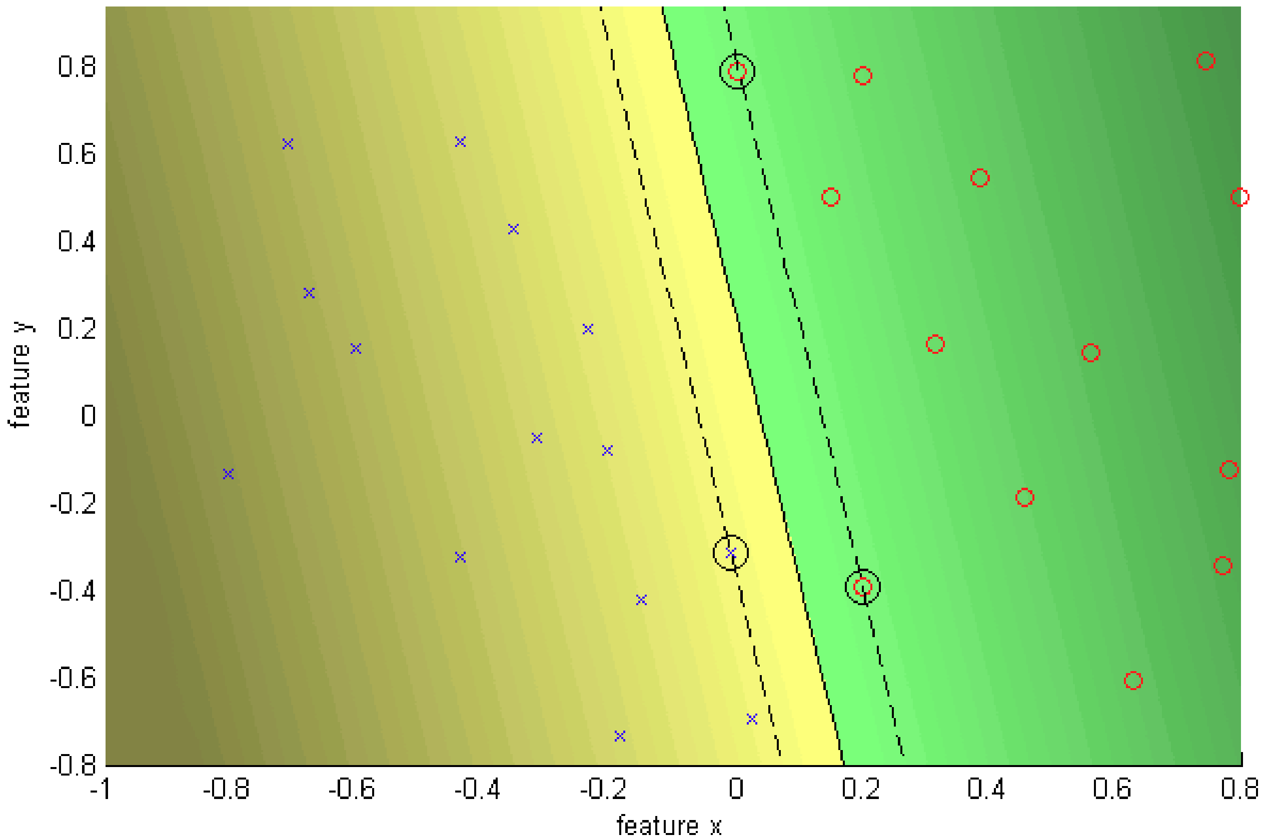
Минимизируем $\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$

При условии $y_i(wx_i + b) \geq 1 - \xi_i$

C – параметр регуляризации

В реальном случае идеально разделить данные невозможно (эмпирический риск всегда больше 0)

Пример C=1

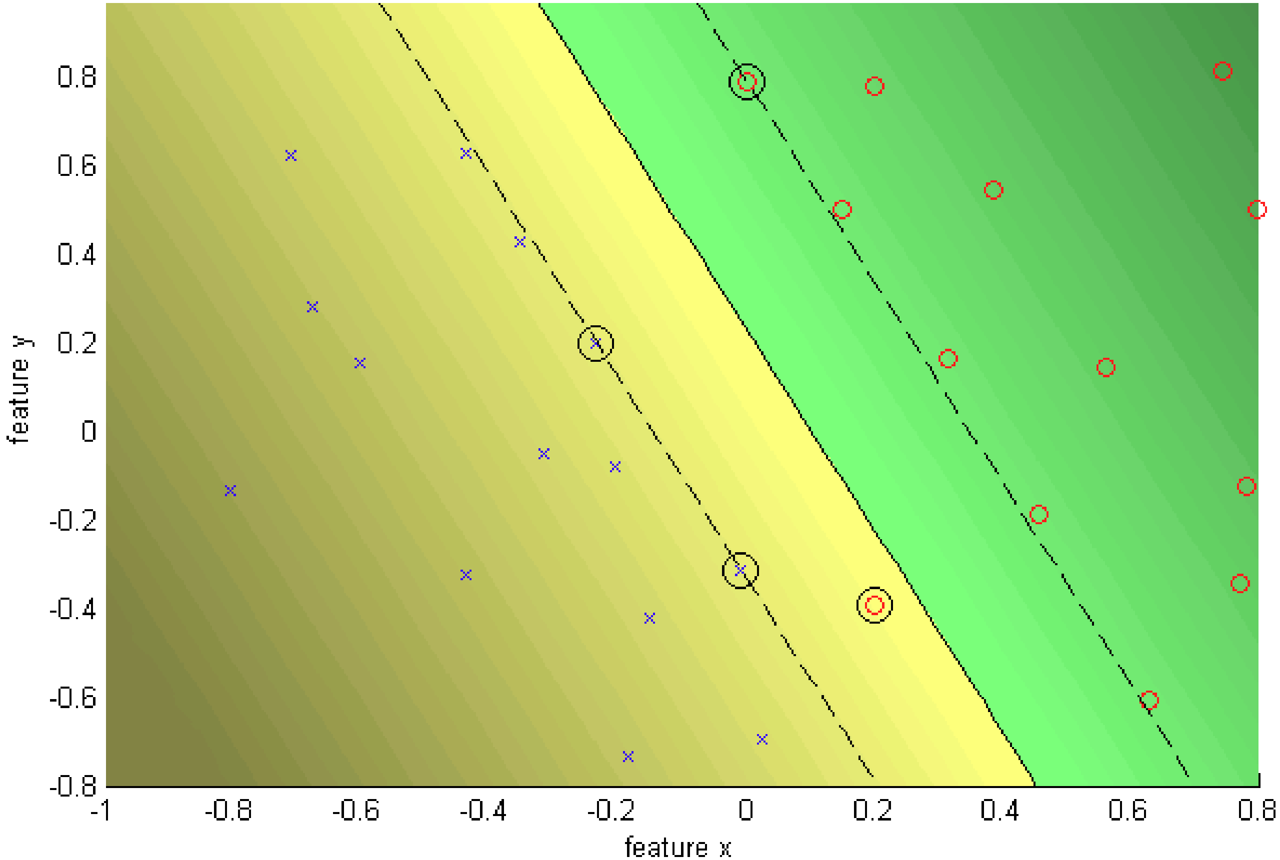


Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%

Source: http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Пример C=10



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%

Source: http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Функция потерь для SVM

- Мы хотим найти $\min(\|w\|^2 + C\sum_{i=1}^{n}\xi_i)$

  при условиях $y_i(wx_i + b) \geq 1 - \xi_i$

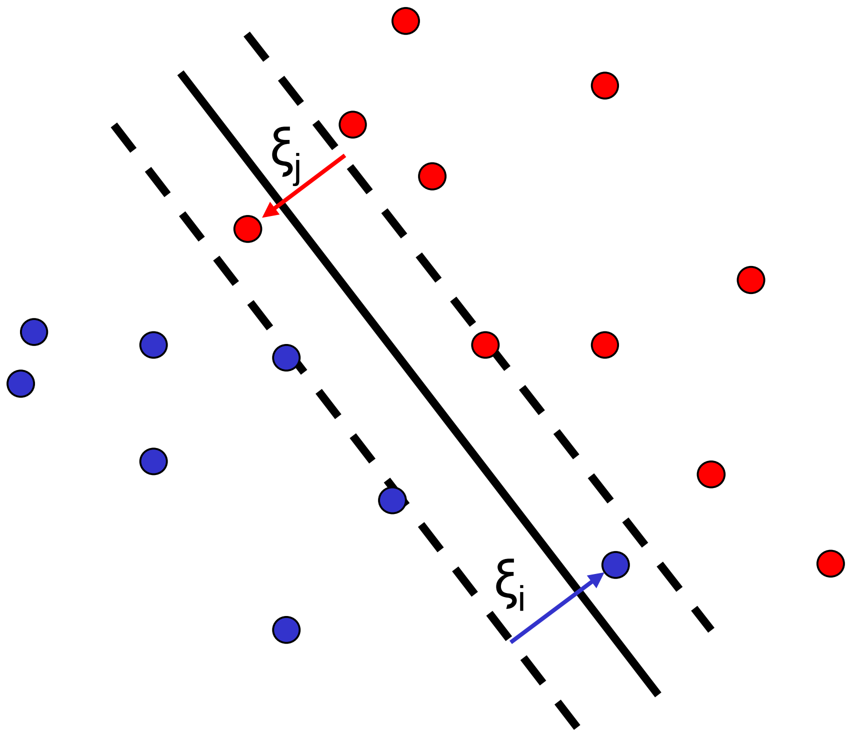
- Перепишем $\mathrm{y_i}(\mathrm{wx_i} + \mathrm{b}) \geq 1 - \xi_i$ как $\mathrm{y_i} f(\mathrm{x_i}) \geq 1 - \xi_i$
- Поскольку $\xi_i \geq 0$ то

  $$\xi_i = \max(0, 1 - \mathrm{y_i} f(\mathrm{x_i}))$$

- Тогда получаем задачу безусловной оптимизации

$$\min(\underbrace{\|w\|^2}_{\text{Regularization}} + \underbrace{C\sum_{i=1}^{n}\max(0, 1 - y_i f(x_i))}_{\text{Loss function}})$$

Функция потерь для SVM

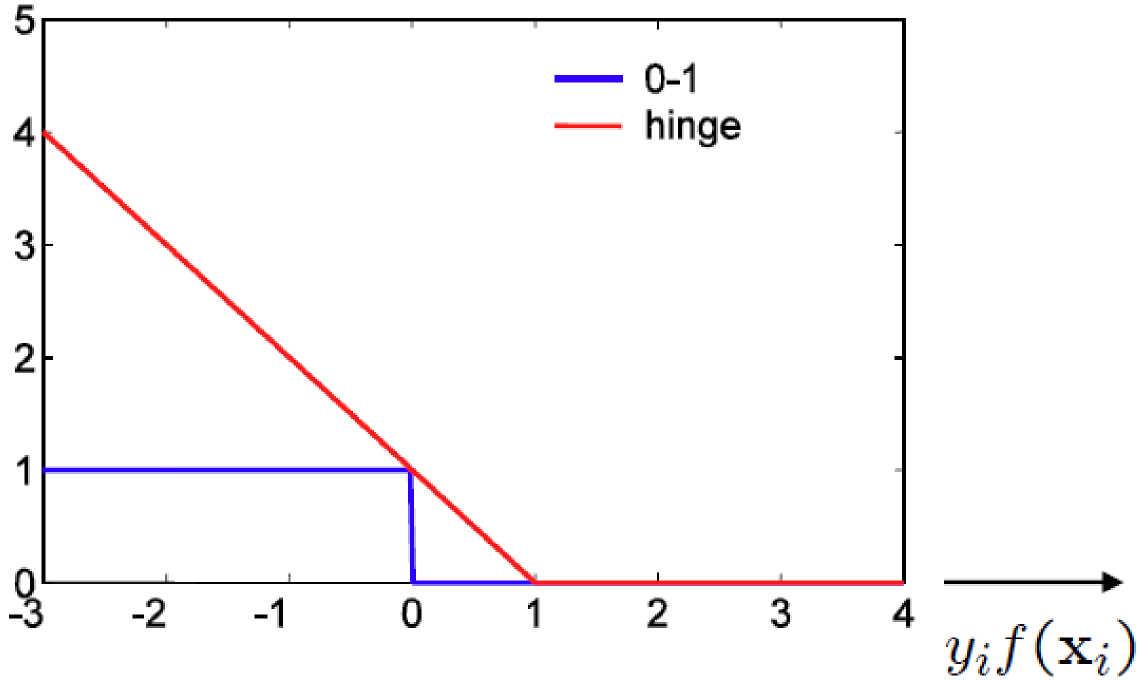


$$\min(\|w\|^2 + C\sum_{i=1}^{n}\max(0,1-y_i f(x_i))$$

Какой вклад точек в функцию потерь?

- yf(x)>1 – точка вне зазора, вклад в loss = 0
- yf(x)=1 – точка на границе зазора (опорный вектор), вклад в loss = 0
- yf(x)<1 – точка нарушает зазор, вклад в loss

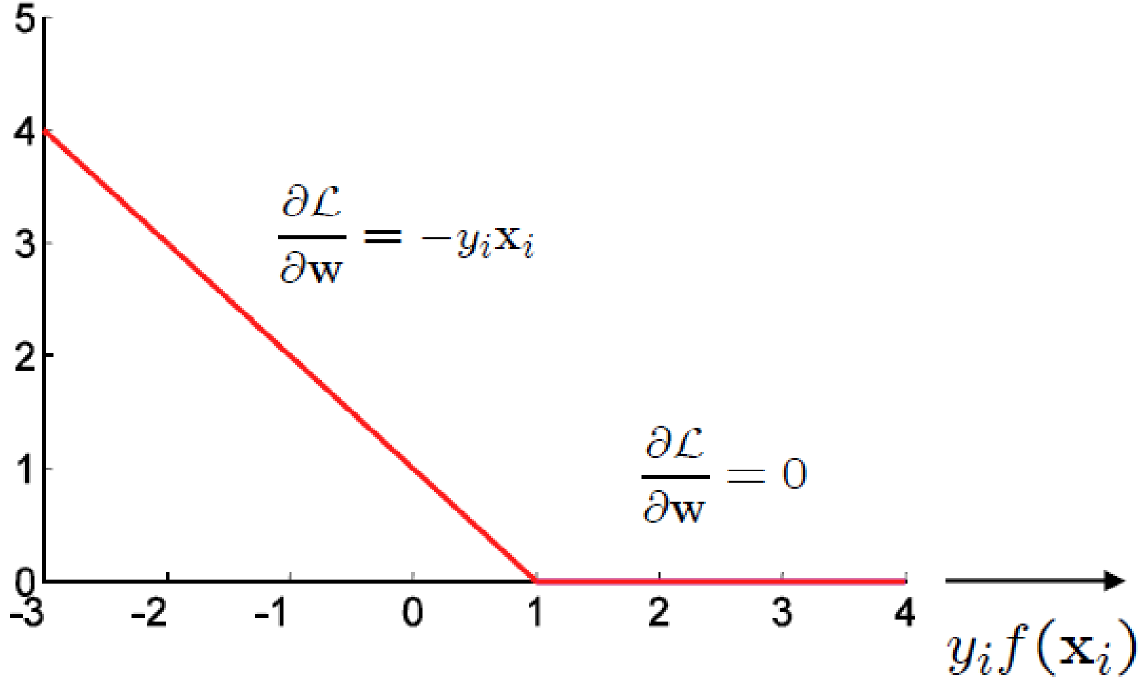
Функция потерь



- $\max(0, 1 - y_i f(x_i))$ - “hinge” loss (гребневая функция потерь)
- Аппроксимация 0-1 функции потерь

Субградиент

$$\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f(\mathbf{x}_i)) \qquad f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$



Многоклассовый SVM

- Идея – нужно обеспечить «зазор» Δ между верным и неверными ответами
  - Т.е. score для всех неверных классов был меньше score для верного класса на значение Δ
- Пусть:
  - (x_i, y_i)- пример из выборки
  - $s_j = f(x_i, W)_j$ - score класса j для примера x_i
- Тогда функция потерь:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

Многоклассовый SVM

- Аналогично бинарному линейному классификатору добавляем регуляризатор:

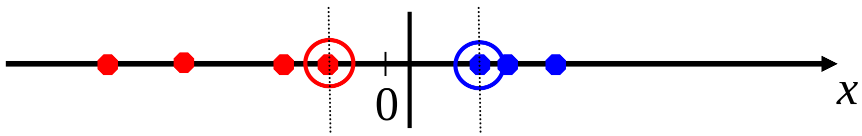
$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

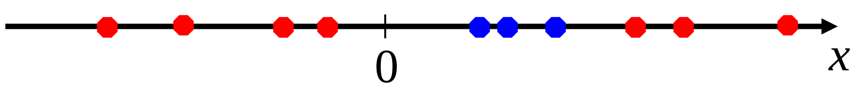
- Идея – обеспечить требуемый зазор за счёт как можно меньших весов W

Нелинейные SVM

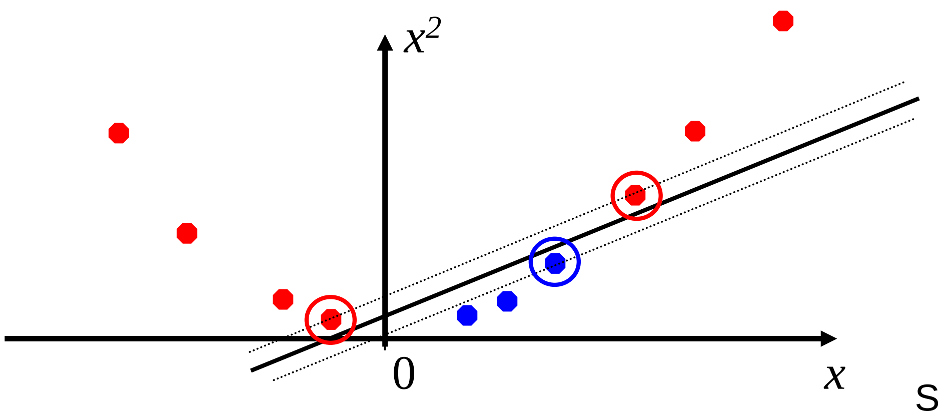
- На линейно разделимых данных SVM работает отлично:



- Но на более сложных данных не очень:



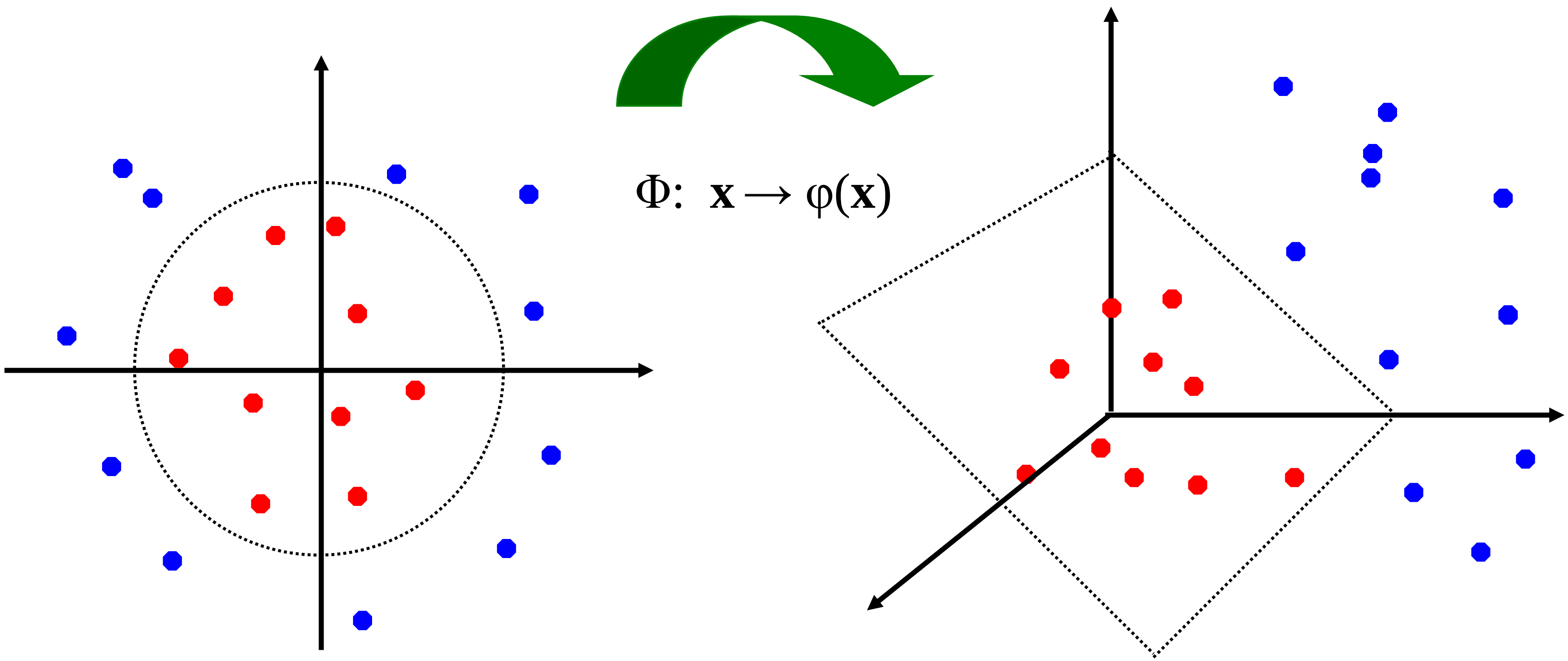
- Можно отобразить данные на пространство большей размерности и разделить их линейно там:



Slide credit: Andrew Moore

Нелинейные SVM

- **Идея:** отображение исходного пространства параметров на какое-то многомерное пространство признаков (feature space) где обучающая выборка линейно разделима:



Slide credit: Andrew Moore

Нелинейные SVM

- Вычисление скалярных произведений в многомерном пространстве вычислительно сложно
- *The kernel trick:* вместо прямого вычисления преобразования $\boldsymbol{\varphi}(\mathbf{x})$, мы определим ядровую функцию K:

$$K(\mathbf{x}_i\, ,\, \mathbf{x}_{jj}) = \boldsymbol{\varphi}(\mathbf{x}_i\,) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$$

- Чтобы все было корретно, ядро должно удовлетворять условию Мерсера (*Mercer's condition*)
  - Матрица $K(x_i,x_j)$ должна быть неотрицательно определенной
- С помощью ядра мы сможем построить нелинейную решающую функцию в исходном пространстве:

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

Пример ядра

- Полиномиальное: $K(x,y) = ((\mathbf{x}, y) + c)^d$

- Пусть d=2, x=(x_1,x_2):

$$K(x,y) = ((\mathbf{x}, y) + c)^2 = (x_1 y_1 + x_2 y_2 + c)^2 = \varphi(x) \cdot \varphi(y)$$

$$\varphi(x) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2, \sqrt{2c} x_1, \sqrt{2c} x_2, c)$$

- Т.е. из 2х мерного в 6и мерное пространство

SVM - резюме

- Один из базовых методов классификации
- Линейный метод достаточен во многих случаях, если у нас вектор-признак большой размерности
- Для нелинейных методов придумано много видов ядер
- Два способа нахождения параметров:
  - Точный минимум по всей выборке
  - Стохастическим градиентным спуском (чаще всего)

- Много доступных библиотек: http://www.kernel-machines.org/software

Softmax

- Другой вариант линейного многоклассового классификатора
- Мы заменяем гребневую функцию потерь на кросс-энтропию:

$$L_i = -\log\underbrace{\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)}_{\text{Функция softmax}}$$

- Чтобы вычислить её, выходы линейного классификатора мы подаем на нормализационное преобразование – в функцию softmax
- Score всех классов можно интерпретировать как «вероятности» классов (т.к. после softmax сумма = 1, и все от 0 до 1)

Смысл Softmax

- Пусть q – выходы softmax классификатора

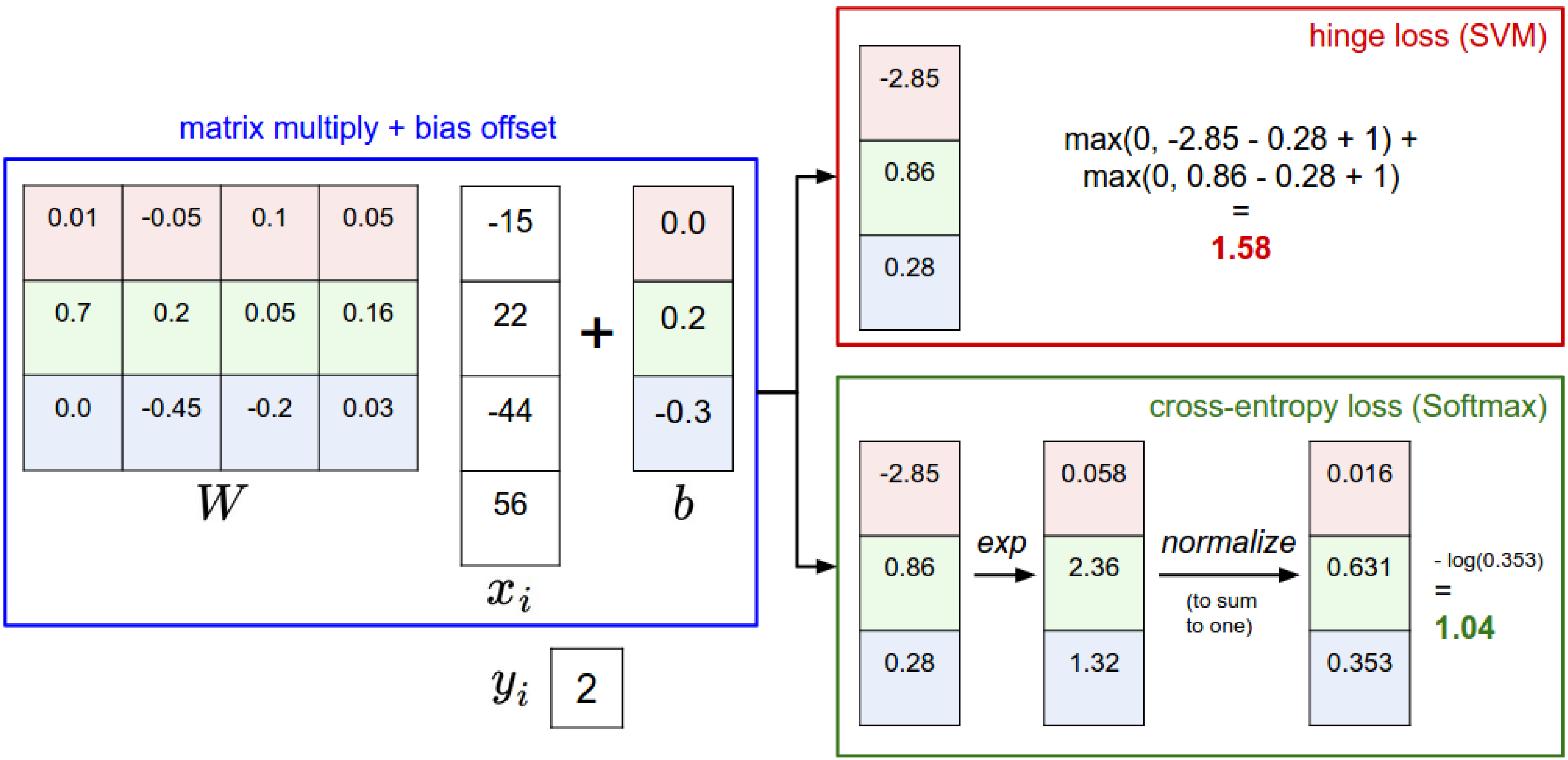
$$q_i = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

- Пусть p = [0,..,0,1,0,..0] – целевое распределение ответов, где 1 соответствует истинному классу
- Тогда наша функция потерь соответствует оценке *кросс-энтропии* между целевым и выданным распределениями

$$H(p,q) = -\sum_x p(x) \log q(x)$$

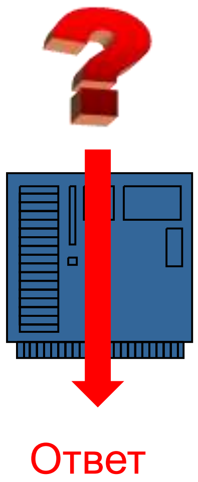
- Мы хотим, чтобы основной «вес» ответа приходился на верный класс

SVM vs Softmax



Предсказательная способность

- Научились обучать линейные классификаторы с помощью стохастического градиентного спуска на обучающей выборке
- Нам нужно, чтобы классификатор хорошо работал на данных, которые не использовались при обучении
- «Предсказательная способность» - качество классификации вне обучающей выборки
- Как оценить *предсказательную способность*?
- Обучать на одной части известных данных, проверять на другой
  - «Удерживание»
  - «Скользящий контроль» (Cross-validation)

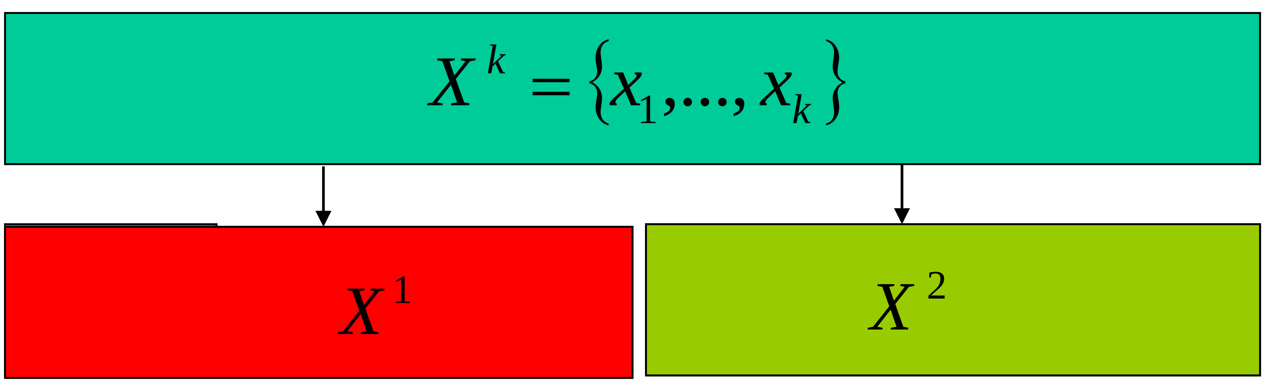


Удерживание

Будем разбивать имеющуюся обучающую выборку на части

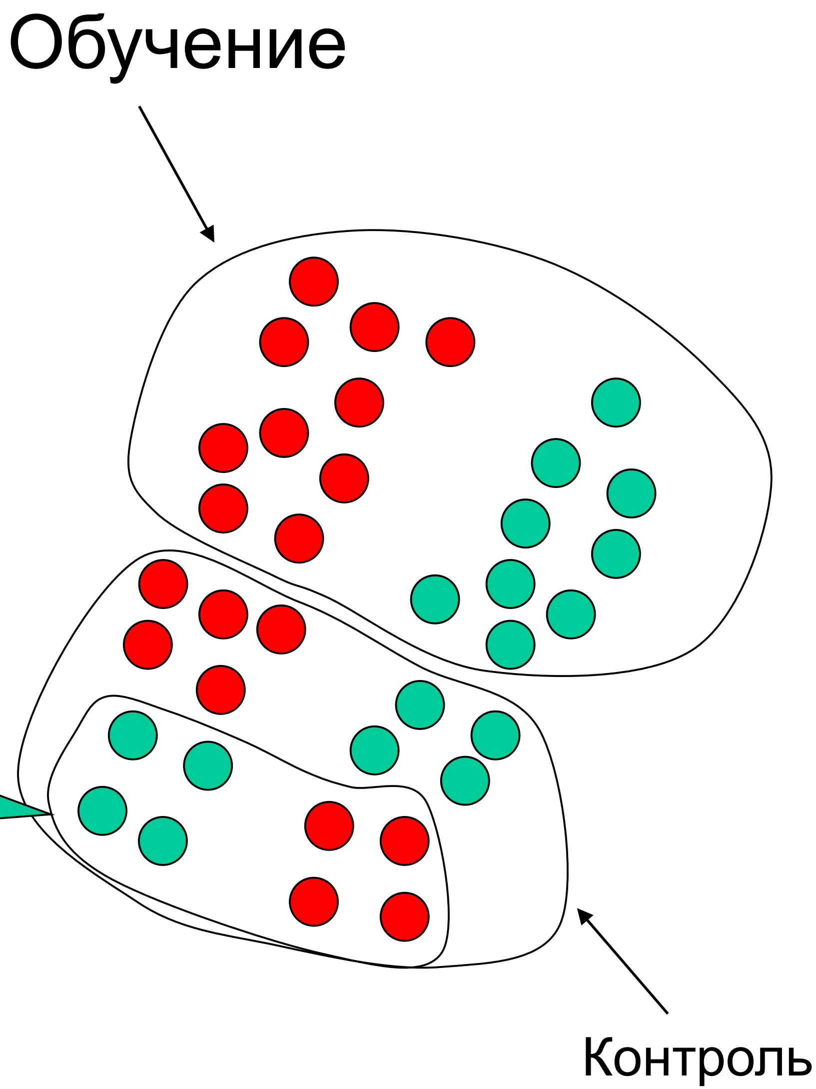
- На одной части будем обучать классификатор (минимизировать эмпирический риск)
- На другой будем измерять ошибки обученного классификатора (оценивать предстказательную способность)

$$R(f,X) \sim P\left(f(x) \neq y \mid X^c\right) = \frac{1}{c}\sum_{j=1}^{c}\left[f(x_j) \neq y_j\right]$$



Свойства «удерживания»

- Быстро и просто рассчитывается
- Некоторые «сложные» прецеденты могут полностью попасть в только одну из выборок и тогда оценка ошибки будет смещенной



Ошибка произойдет не по вине классификатора, а из-за разбиения!

Скользящий контроль

- Разделим выборку на d непересекающихся частей и будем поочередно использовать одно из них для контроля а остальные для тренировки

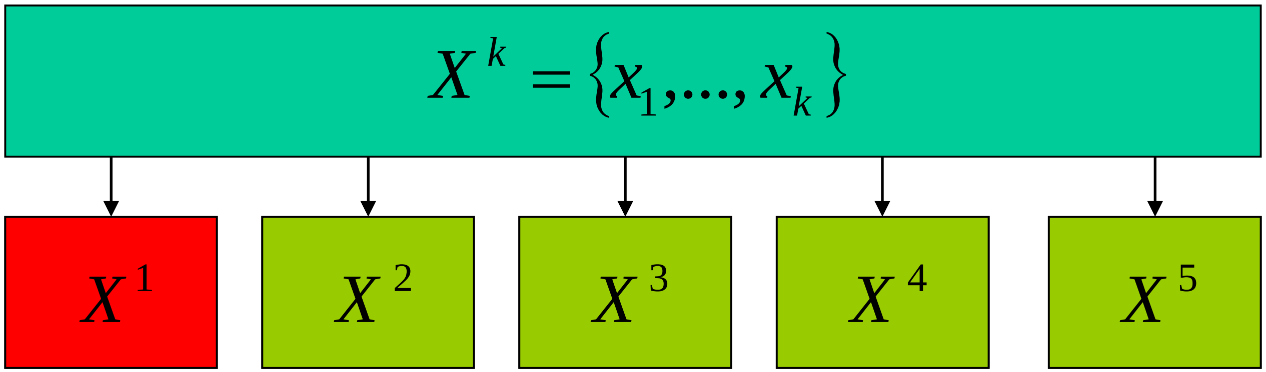
  - Разбиваем: $\{X^i\}_1^d : X^i \cap X^j = 0, i \neq j$

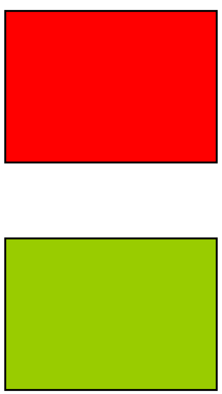
    $$\bigcup_{i=1}^{d} X^i = X^k$$

  - Приближаем риск:

$$P(f(X^k) = y^*) \approx \frac{1}{d} \sum_{i=1}^{d} P(f(X^i) \neq y^* \mid \bigcup_{i \neq j} X^i)$$

Иллюстрация





Контроль

Обучение

Результат считается как средняя ошибка по всем итерациям

Свойства

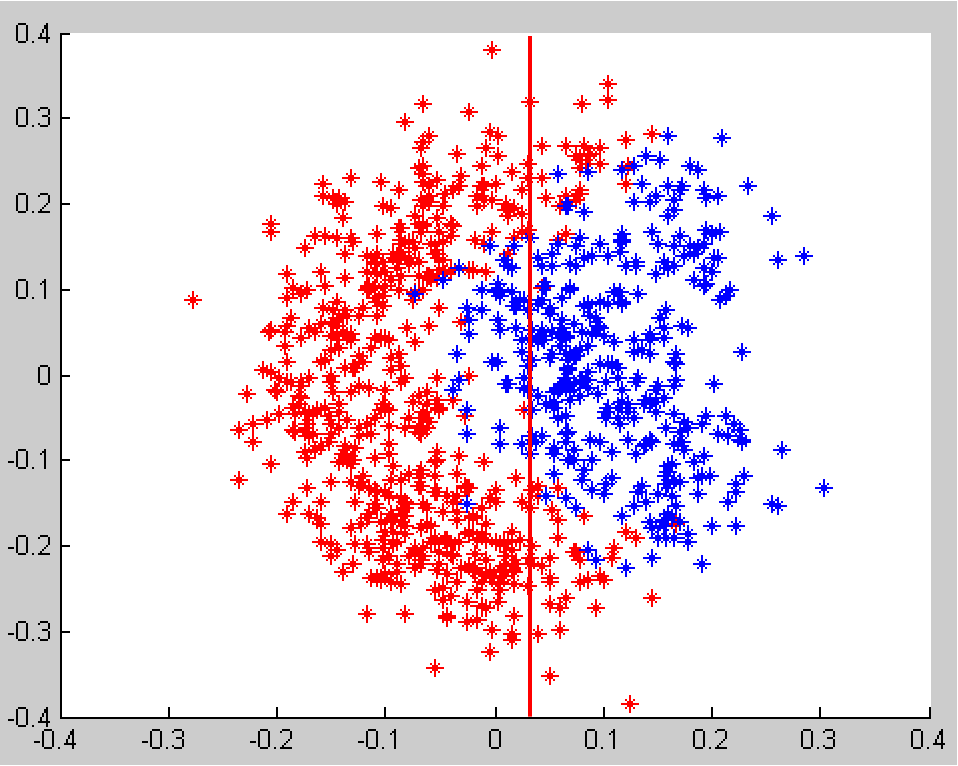
- В пределе равен общему риску
- Каждый прецедент будет один раз присутствовать в контрольной выборке
- Обучающие выборки будут сильно перекрываться (чем больше сегментов, тем больше перекрытие)
  - Если одна группа «сложных прецедентов» попала полностью в один сегмент, то оценка будет смещенной
- Предельный вариант – “leave one out”
  - Обучаемся на всех элементах, кроме одного
  - Проверяем на одном
  - Повторяем для всех элементов

Пример

- Данные – точки на плоскости
- «Классификатор» – порог по оси X

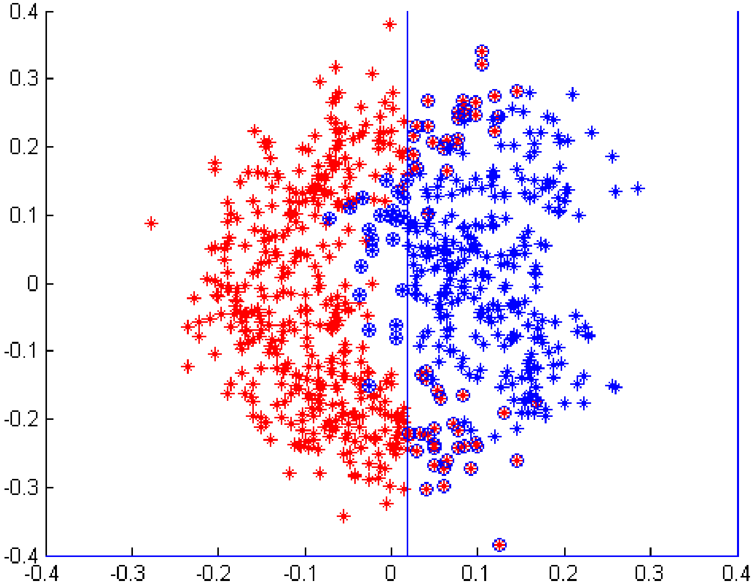
$$a(x^1, x^2) = \begin{cases} +1, x_1 > \Theta \\ -1, x_1 \le \Theta \end{cases}$$

- Будем обучать его, пользуясь разными подходами и измерять ошибки
  - Удерживание
  - Скользящий контроль



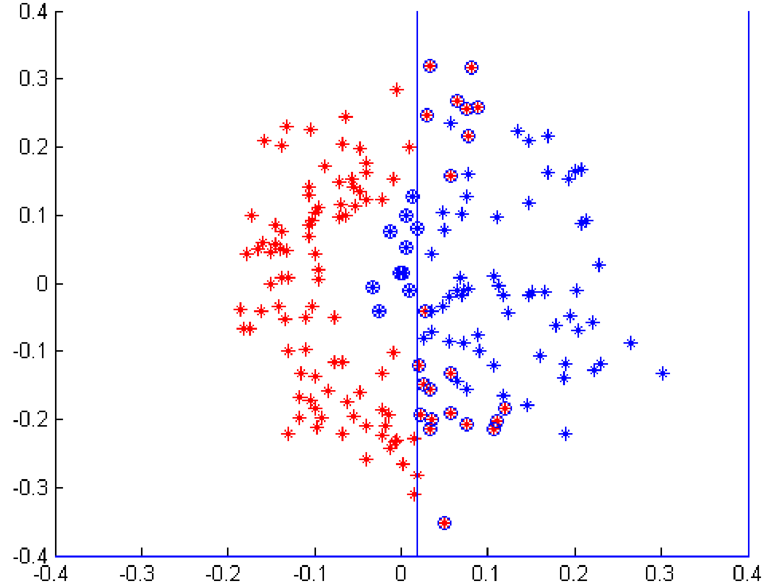
Удерживание

Ошибка: 0.1133



Тренировочная выборка

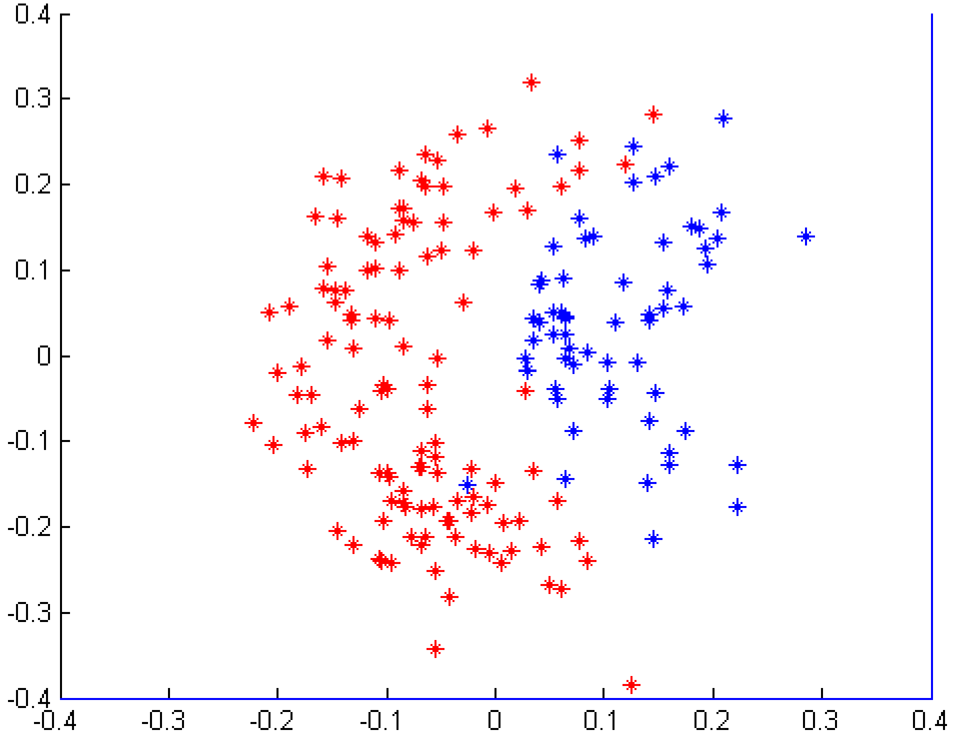
Ошибка: 0.1433



Контрольная выборка

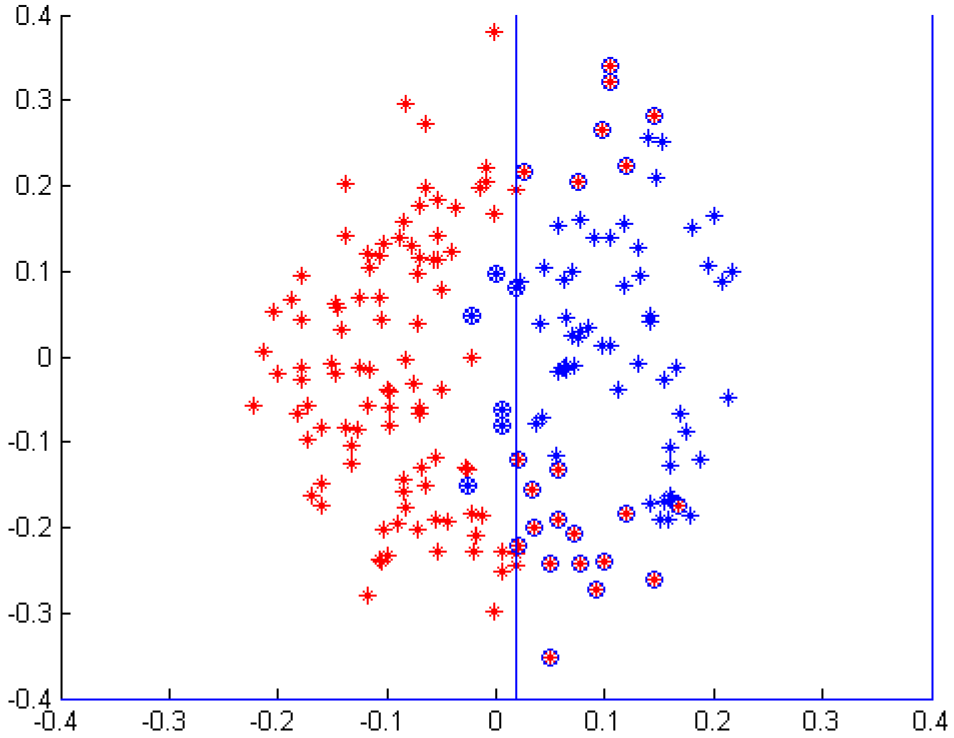
Разобъём данные на 2 части, обучим на одной, проверим на другой

Скользящий контроль: разбиение



Разобъём данные на 5 частей

Скользящий контроль: измерение



Обучим и измерим ошибку 5 раз

Скользящий контроль

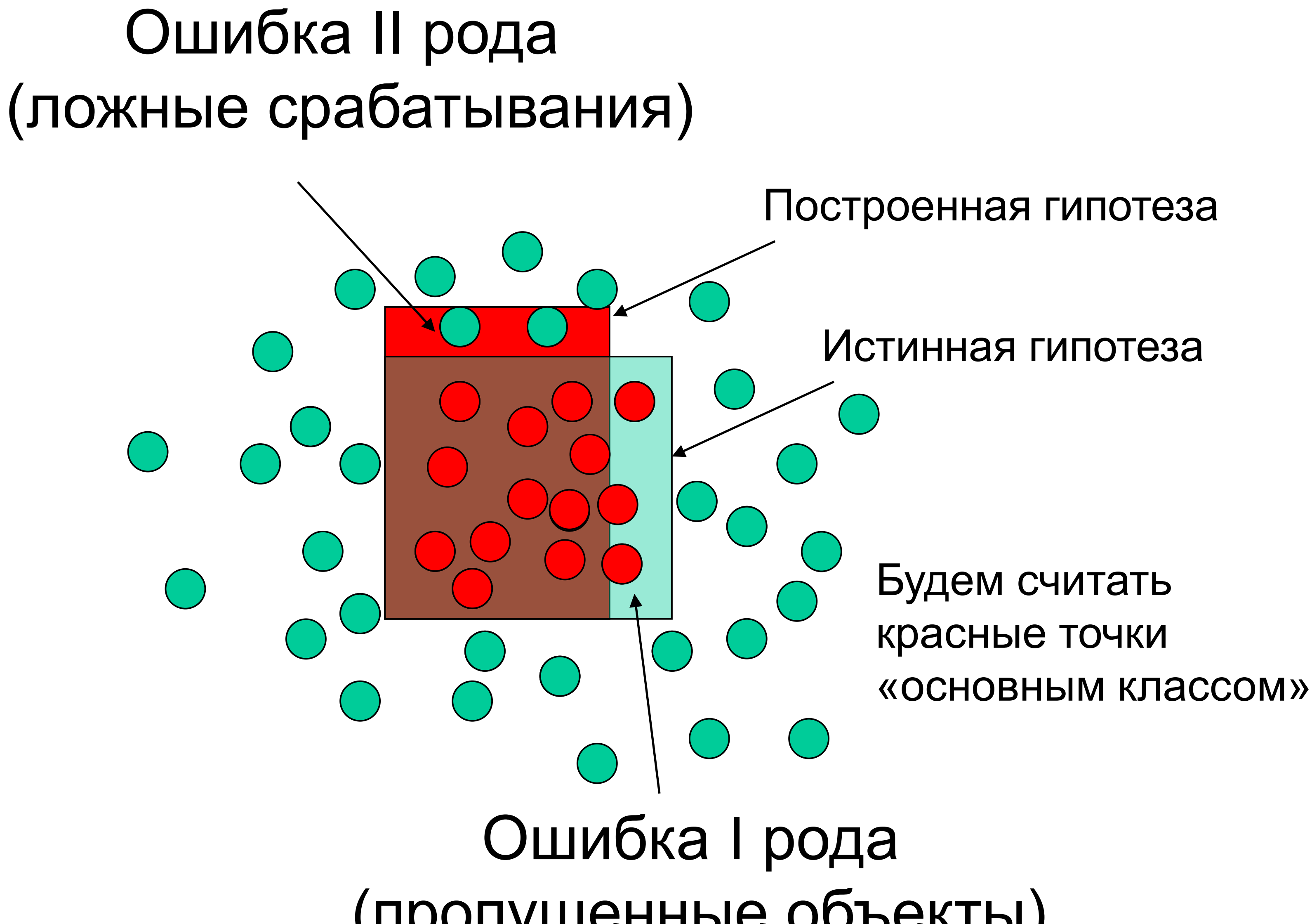
- Тренировочная ошибка:
  - $\{0.1236 \quad 0.1208 \quad 0.1250 \quad 0.1097 \quad 0.1306\}$
  - Среднее = 0.1219

- Ошибка на контроле
  - $\{0.1500 \quad 0.1333 \quad 0.1222 \quad 0.1778 \quad 0.1000\}$
  - Среднее = 0.1367

Ошибки I и II рода

- В наших задачах обычно классы объектов неравнозначны
  - Лицо человека – «основной» класс (положительные примеры)
  - Фон – «вторичный» класс (отрицательные примеры)
- **Ошибка первого рода** равна вероятности принять основной класс за вторичный
  - Вероятность «промаха», когда искомый объект будет пропущен (пропустить лицо)
- **Ошибка второго рода** равна вероятности принять вторичный класс за основной
  - Вероятность «ложной тревоги», когда за искомый объект будет принят «фон»

Ошибки I и II рода



Чувствительность и избирательность

- **Чувствительность** – вероятность дать правильный ответ на пример основного класса

$$sensitivity = P(f(x) = y \mid y = +1)$$

  - Также *уровень обнаружения (detection rate)*

- **Избирательность** – вероятность дать правильный ответ на пример вторичного класса

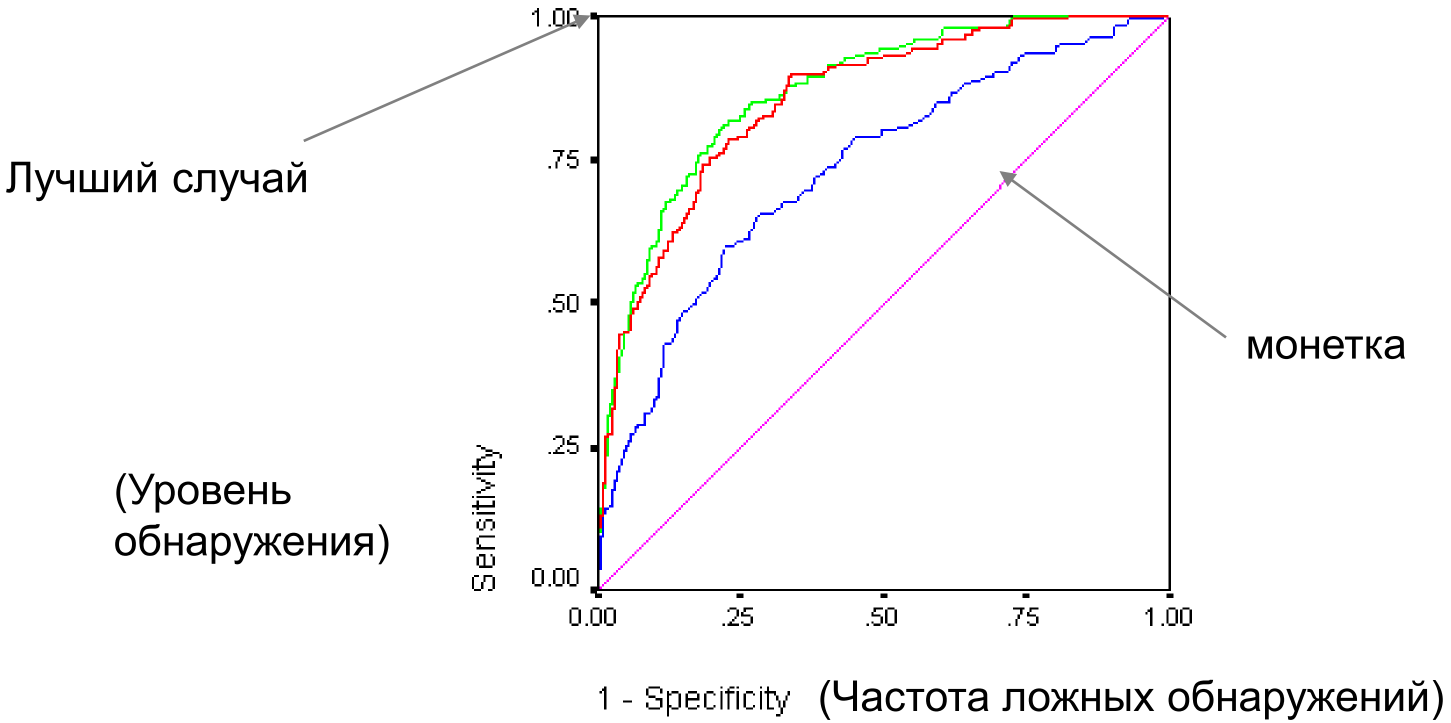
$$specificity = P(f(x) = y \mid y = -1)$$

  - Обычно считают частоту ложных обнаружений

    False positive rate = 1 - specificity

ROC кривая

- ROC – Receiver Operating Characteristic curve
  - Кривая, отражающая зависимость чувствительности и ошибки второго рода

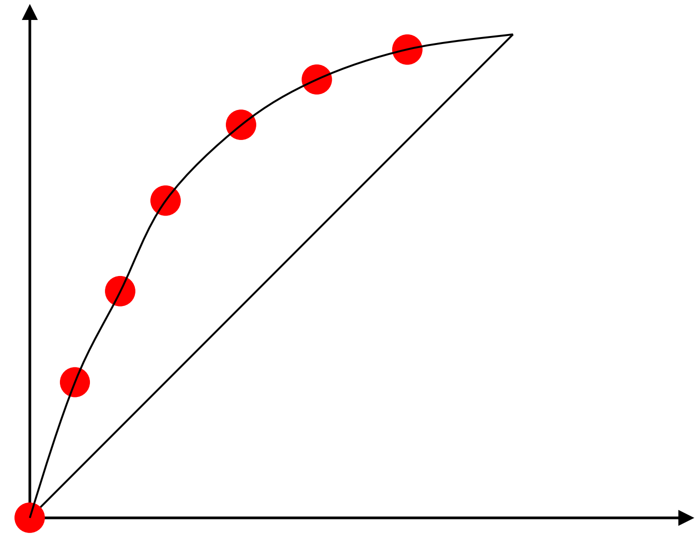


Как можно её использовать?

ROC кривая - Построение

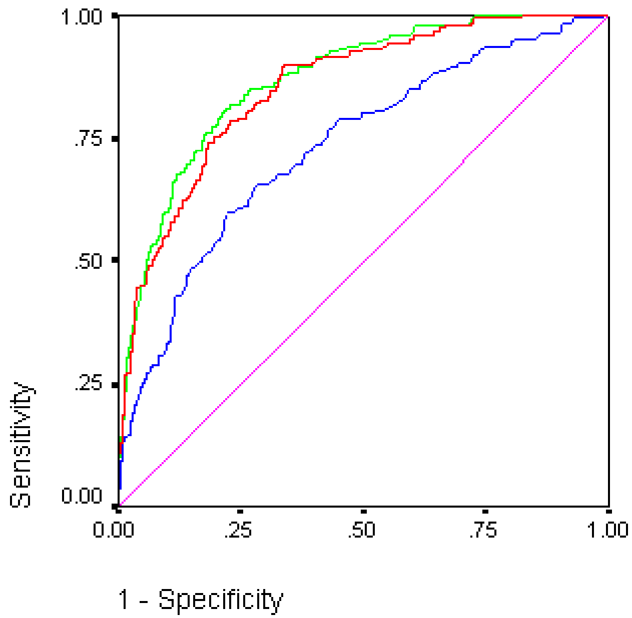
- Для различных значений параметра строится таблица ошибок
  - Сам параметр в таблице не участвует!
  - Классификатор строится и оценивается на разных выборках!
- По таблице строиться набор точек в плоскости sensitivity x (false positive)
  - Каждая строка таблицы - точка
- По точкам строиться кривая

Sensitivity	False Positive
0.0	0.0
0.25	0.5
0.5	0.8
…	…
1.0	1.0



Анализ ROC кривой

- Площадь под графиком – AUC
  - Дает некоторый объективный показатель качества классификатора
  - Позволяет сравнивать разные кривые
- Соблюдение требуемого значения ошибок I и II рода
  - Зачастую, для конкретной задачи существуют рамки на ошибку определенного рода. С помощью ROC можно анализировать возможность текущего решения соответствовать требованию



$f(xi,W,b)=Wxi+b$

Резюме лекции

- Мы познакомились с рядом ключевых понятий машинного обучения
- Посмотрели классификаторы:
  - Пороговый классификатор
  - Линейный классификатор
  - Многоклассовый линейный классификатор
  - SVM
  - Softmax