Линейные модели классификации

на основе CSC2515 University of Toronto

Владимир Анатольевич Судаков

2025

Что такое «линейная» классификация?

- Классификация по своей сути нелинейна
  - Он помещает неидентичные вещи в один и тот же класс, разница во входном векторе иногда приводит к нулевому изменению ответа (к чему это приводит?)
- «Линейная классификация» означает, что часть, которая адаптируется, является линейной.
  - За адаптивной частью следует фиксированная нелинейность.
  - Ему также может предшествовать фиксированная нелинейность (например, нелинейные базисные функции).

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0, \qquad Decision = f(y(\mathbf{x}))$$

адаптивная линейная функция

фиксированная нелинейная функция

Представление целевых значений для классификации

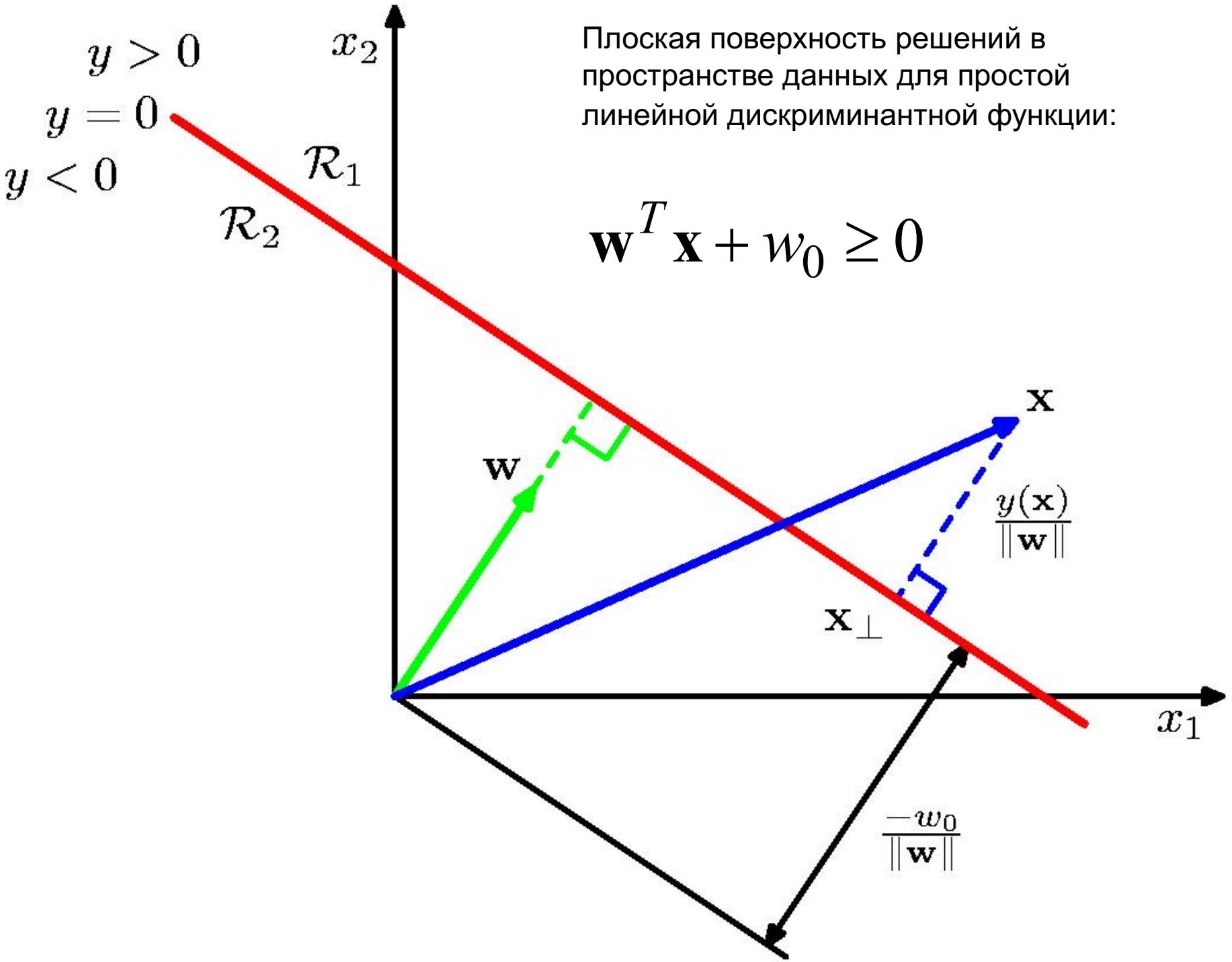
- Если есть только два класса, мы обычно используем один действительный выходной сигнал, который имеет целевые значения 1 для «положительного» класса и 0 (или иногда -1) для другого класса.
  - Для вероятностных меток классов целевым значением может быть вероятность положительного класса, а выходные данные модели также могут представлять собой вероятность, которую модель присваивает положительному классу.
- Если имеется N классов, мы часто используем вектор из N целевых значений, содержащий одну 1 для правильного класса и нули в остальных местах.
  - Для вероятностных меток мы можем затем использовать вектор вероятностей классов в качестве целевого вектора.

Три подхода к классификации

- Используйте дискриминантные функции напрямую, без вероятностей:
  - Преобразовать входной вектор в одно или несколько действительных значений, чтобы можно было применить простую операцию (например, определение порога) для получения класса.
    - Действительные значения следует выбирать так, чтобы максимально использовать полезную информацию о метке класса, содержащуюся в действительном значении.
- Вывести условные вероятности классов: $p(class = C_k \mid \mathbf{x})$
  - Вычислите условную вероятность каждого класса.
    - Затем примите решение, которое минимизирует некоторую функцию потерь.
- Сравните вероятность ввода в отдельных, специфичных для класса, генеративных моделях.
  - Например, подгоните многомерную гауссову функцию к входным векторам каждого класса и посмотрите, какая гауссова функция сделает вектор тестовых данных наиболее вероятным. (Это лучший вариант?)

Плоская поверхность решений в пространстве данных для простой линейной дискриминантной функции:

$$\mathbf{w}^T\mathbf{x} + w_0 \geq 0$$



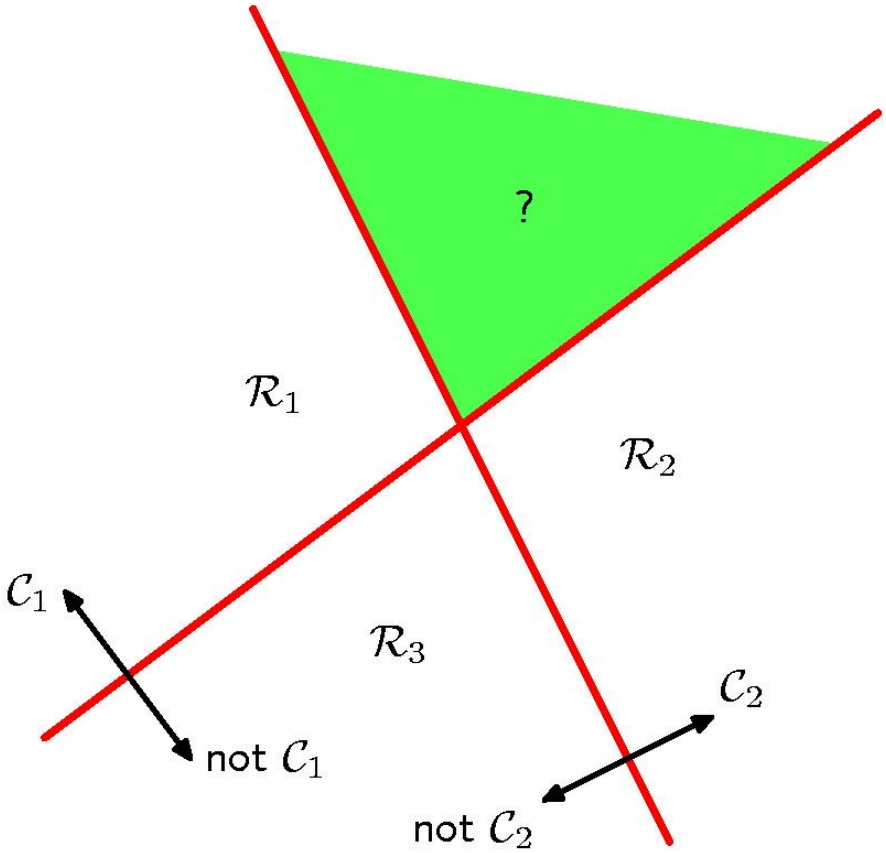
Напоминание: три разных пространства, которые легко спутать

- Пространство весов
  - Каждая ось соответствует весу
  - Точка — это вектор весов
  - Размерность = #веса +1 дополнительное измерение для потери
- Пространство данных
  - Каждая ось соответствует входному значению
  - Точка — это вектор данных. Поверхность решения — это плоскость.
  - Размерность = размерность вектора данных
- «Пространство случая»
  - Каждая ось соответствует учебному случаю
  - Точка присваивает скалярное значение каждому обучающему случаю.
    - Таким образом, он может представлять одномерные цели или может представлять значение одного входного компонента по всем обучающим данным.
  - Размерность = #кейсы обучения

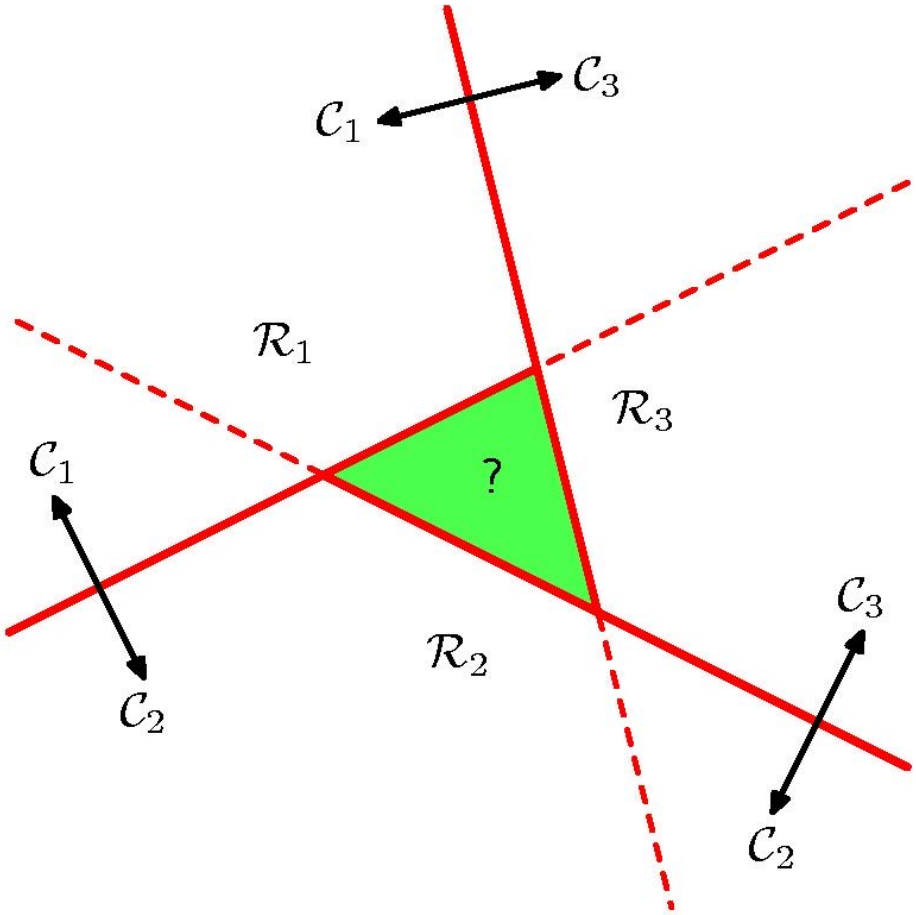
Дискриминантные функции для N>2 классов

- Одной из возможностей является использование N двусторонних дискриминантных функций.
  - Каждая функция отличает один класс от остальных.
- Другая возможность — использовать N(N-1)/2 двусторонних дискриминантных функций.
  - Каждая функция различает два конкретных класса.
- Оба эти метода имеют проблемы

Проблемы с многоклассовыми дискриминантными функциями



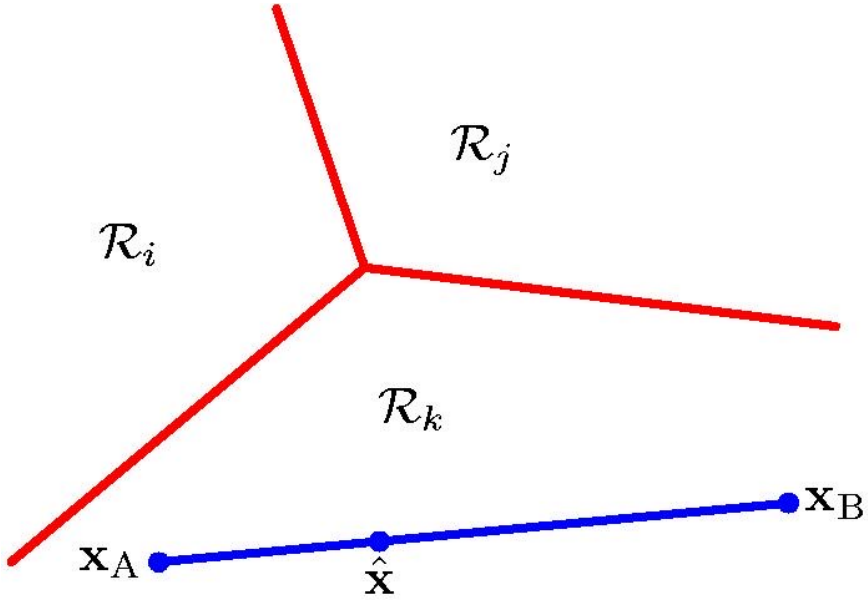
Более одного хорошего ответа



Двусторонние предпочтения не обязательно должны быть транзитивными!

Простое решение

- Используйте N $y_i, y_j, y_k \cdots$ дискриминантных функций и выберите максимальную.
  - Это гарантированно даст последовательные и выпуклые области решений, если *y* линейна.



$$y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A) \;\; and \;\; y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$$

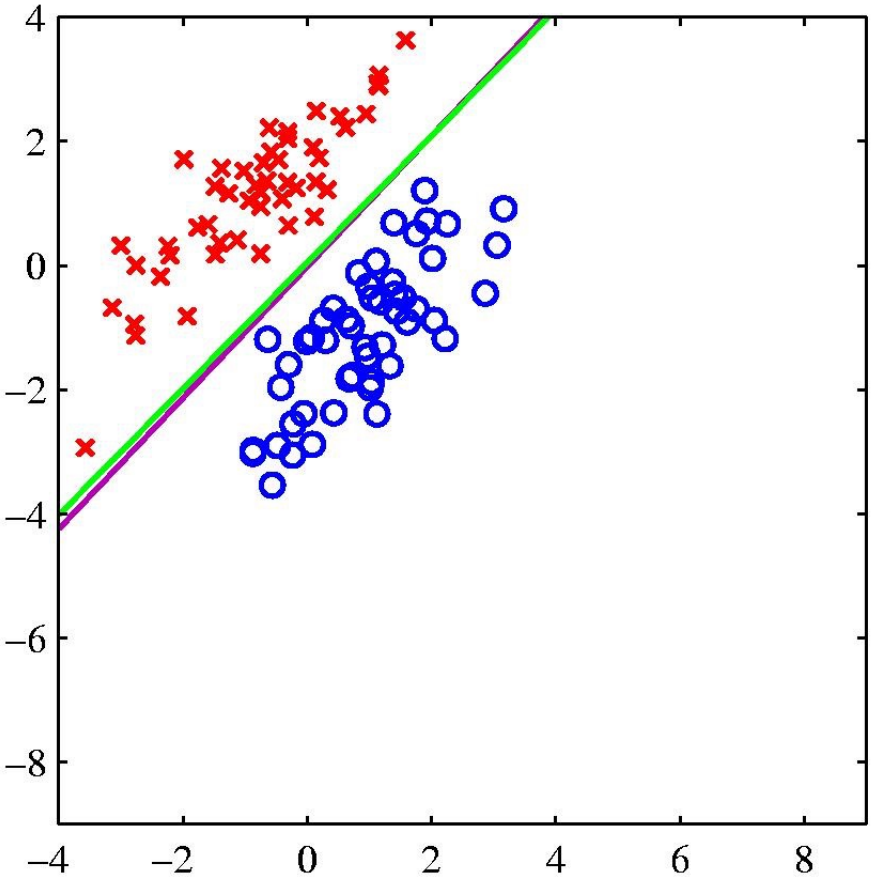
$$implies \; (for \;\; positive \; \alpha) \;\; that$$

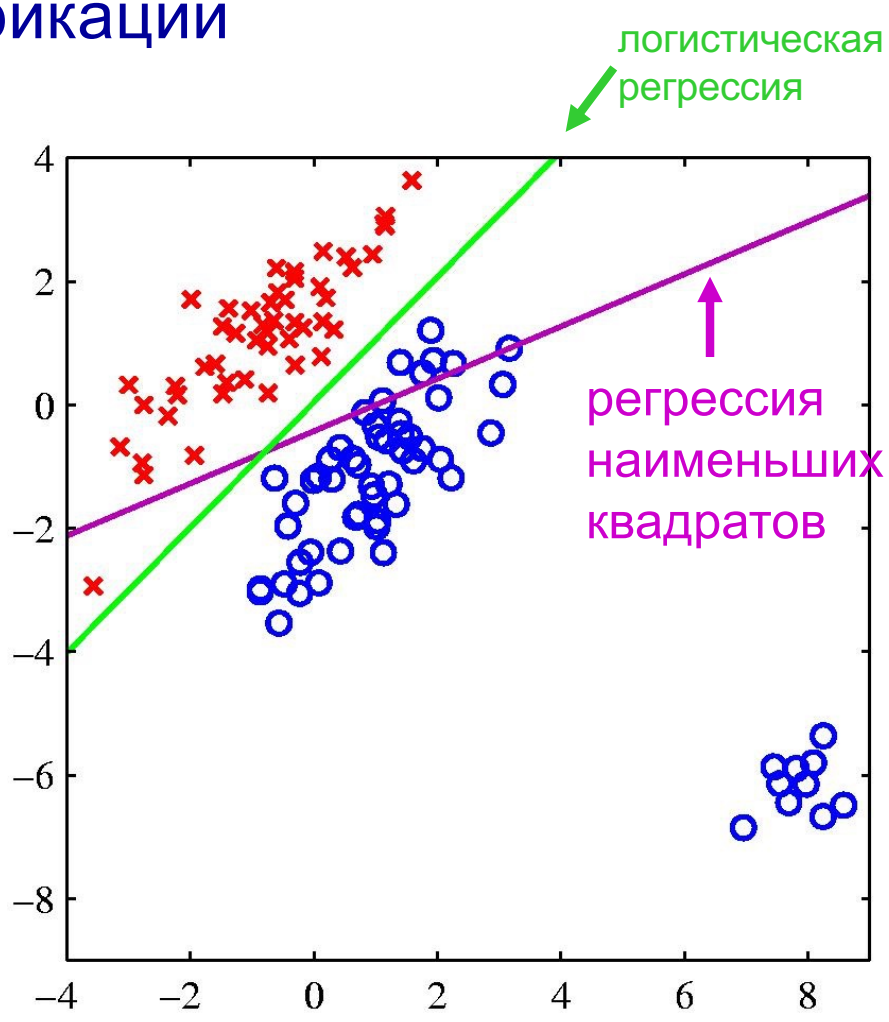
$$y_k\left(\alpha\,\mathbf{x}_A + (1-\alpha)\,\mathbf{x}_B\right) > y_j\left(\alpha\,\mathbf{x}_A + (1-\alpha)\,\mathbf{x}_B\right)$$

Использование метода наименьших квадратов для классификации

- Это неправильный подход и не работает так хорошо, как лучшие методы, но это просто:
  - Он сводит классификацию к регрессии наименьших квадратов.
  - Мы уже умеем строить регрессию. Остаётся только найти оптимальные веса, используя матричную алгебру (см. ML 2).
- Используем цели, которые равны условной вероятности класса с учетом входных данных.
  - Когда имеется более двух классов, мы рассматриваем каждый класс как отдельную задачу (мы не можем избежать этого, если используем функцию принятия решения “max”).

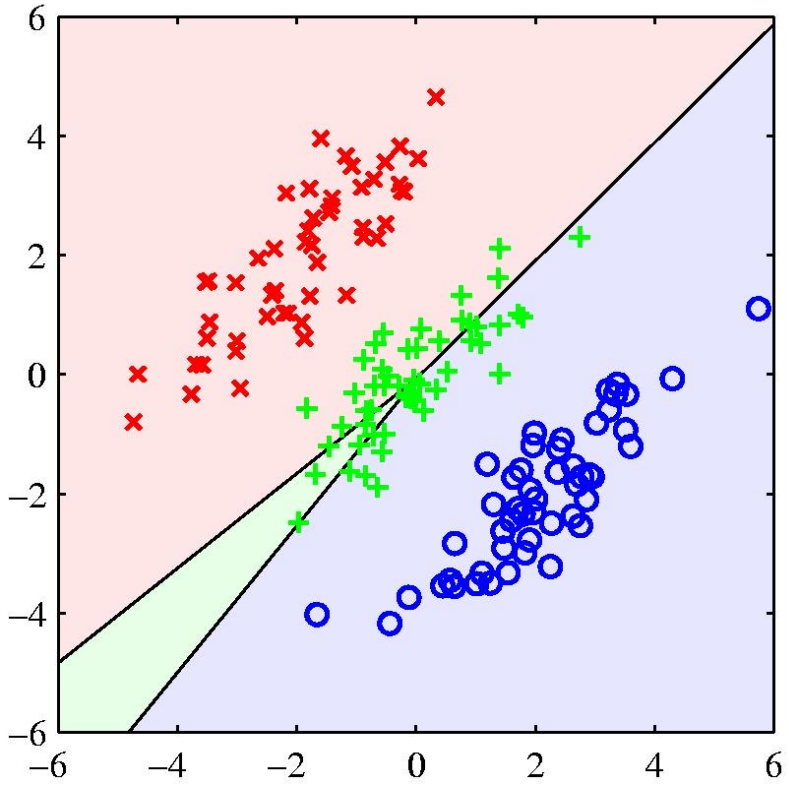
Проблемы с использованием наименьших квадратов для классификации

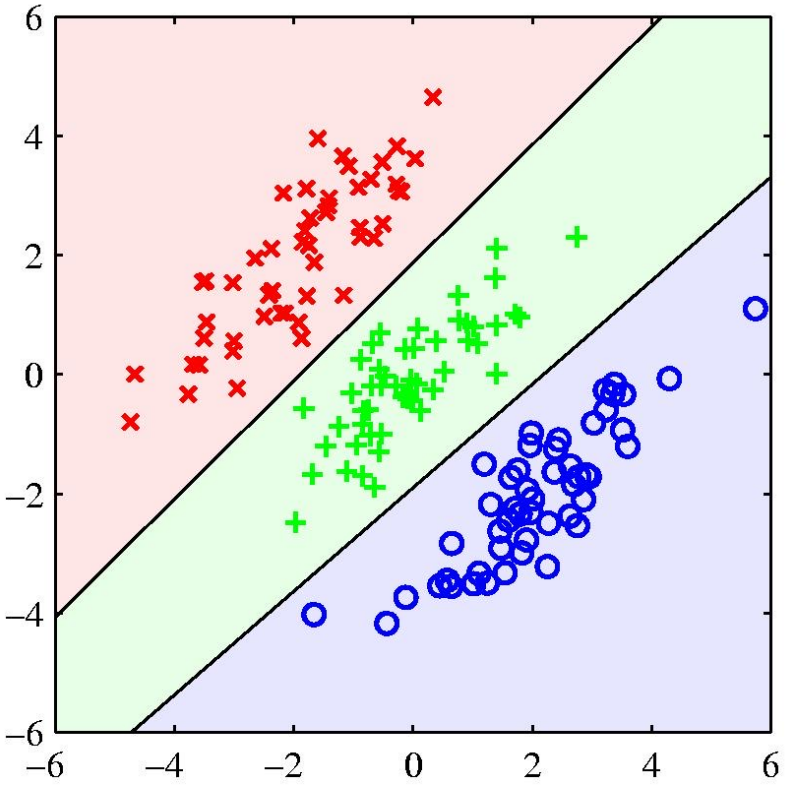




Если правильный ответ — 1, а модель говорит 1,5, она проигрывает, поэтому она меняет границу, чтобы избежать «слишком правильной» оценки.

Другой пример, когда регрессия наименьших квадратов дает плохие поверхности решений

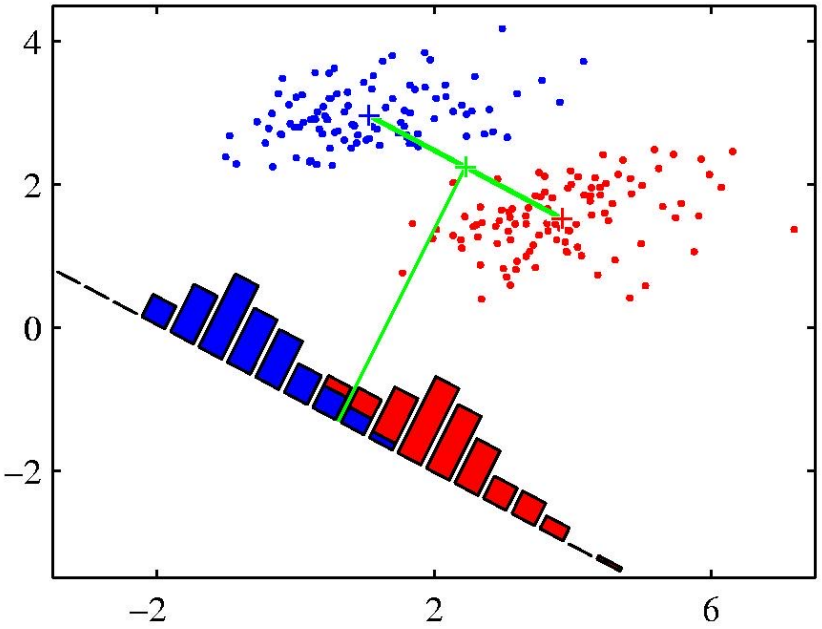


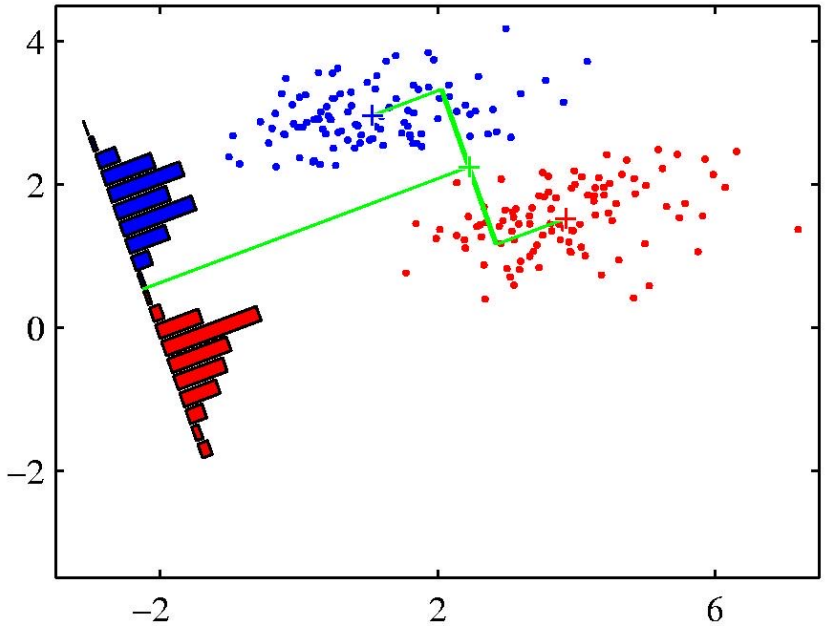


Линейный дискриминант Фишера

- Простая линейная дискриминантная функция представляет собой проекцию данных на одномерный уровень.
  - Итак, выберите проекцию, которая обеспечивает наилучшее разделение классов. Что мы подразумеваем под «наилучшим разделением»?
- Очевидным направлением выбора является направление линии, соединяющей средние значения класса.
  - Однако если основное направление дисперсии в каждом классе не ортогонально этой линии, то это не даст хорошего разделения (см. следующий рисунок).
- Метод Фишера выбирает направление, которое максимизирует отношение межклассовой дисперсии к внутриклассовой дисперсии.
  - Это направление, в котором спроецированные точки содержат наибольшую информацию о принадлежности к классу (в соответствии с гауссовыми предположениями).

Рисунок, демонстрирующий преимущество линейного дискриминанта Фишера





При проецировании на линию, соединяющую классы, классы не очень хорошо разделены.

Фишер выбирает направление, которое делает прогнозируемые классы гораздо плотнее, даже если их прогнозируемые средние значения не так далеко друг от друга.

Математика линейных дискриминантов Фишера

- Какое линейное преобразование лучше всего подходит для дискриминации?
- Проекция на вектор, разделяющий класс, означает, что это кажется разумным:

$$y = \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$$

- Но нам также нужна небольшая дисперсия внутри каждого класса:

$$s_1^2 = \sum_{n \varepsilon C_1} (y_n - m_1)$$

$$s_2^2 = \sum_{n \varepsilon C_2} (y_n - m_2)$$

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- Целевая функция Фишера:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

← между

← в пределах

Больше математики линейных дискриминантов Фишера

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

$$Optimal \;\; solution: \;\; \mathbf{w} \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

Персептрон

- «Персептроны» представляют собой целое семейство обучающихся машин, но стандартный тип состоит из слоя фиксированных нелинейных базисных функций, за которым следует простая линейная дискриминантная функция.
  - Они были введены в конце 1950-х годов и имели простую процедуру онлайн-обучения.
  - Об их способностях были сделаны громкие заявления. Это вызвало множество споров.
  - Исследователи символического ИИ подчеркивали его ограничения (в рамках идеологической кампании против действительных чисел, вероятностей и обучения)
- Машины опорных векторов (Support Vector Machines)— это просто персептроны с умным способом выбора неадаптивных, нелинейных базисных функций и лучшей процедурой обучения.
  - Они имеют те же ограничения, что и персептроны, в отношении того, какие типы функций они могут изучить.
    - Но люди, похоже, об этом забыли.

Персептрон

представляет собой модель бинарной классификации, в которой входной вектор $\mathbf{x}$ сначала преобразуется с помощью нелинейного преобразования для получения вектора признаков $\boldsymbol{\phi}(\mathbf{x})$, а затем используется для построения обобщенной линейной модели вида

$$y(\mathbf{x}) = f(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}))$$

Предполагаем, что вектор $\boldsymbol{\phi}(\mathbf{x})$ включает компонент смещения ϕ_0. Нелинейная функция активации $f(\cdot)$ задаётся ступенчатой функцией вида

$$f(\alpha) = \begin{cases} +1, & \alpha \geq 0 \\ -1, & \alpha < 0 \end{cases}$$

Это связано с тем, что для персептрона удобнее использовать целевые значения

$t = +1$ для класса $\mathcal{C}_1$

$t = -1$ для класса $\mathcal{C}_2$, вместо $t \in \{0,1\}$.

Функция ошибки

рассматриваем функцию ошибки, называемую *критерием персептрона* .

Ищется весовой вектор $\mathbf{w}$, такой, что входные данные $\mathbf{x}_n$, принадлежащие классу $\mathcal{C}_1$, имеют $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) > 0$, тогда как входные данные , принадлежащие классу $\mathcal{C}_2$, имеют $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) < 0$.

Учитывая схему кодирования $t \in \{-1, +1\}$, следует, что все входные данные должны удовлетворять:

$$\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)t_n > 0$$

Таким образом, критерий персептрона пытается минимизировать величину $-\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)t_n$, для всех неправильно классифицированных входных данных.

$$E_P(\mathbf{w}) = -\sum_{n\in\mathcal{M}} \mathbf{w}^T\,\boldsymbol{\phi}(\mathbf{x}_n)t_n$$

где $\mathcal{M}$ обозначает набор неправильно классифицированных шаблонов.

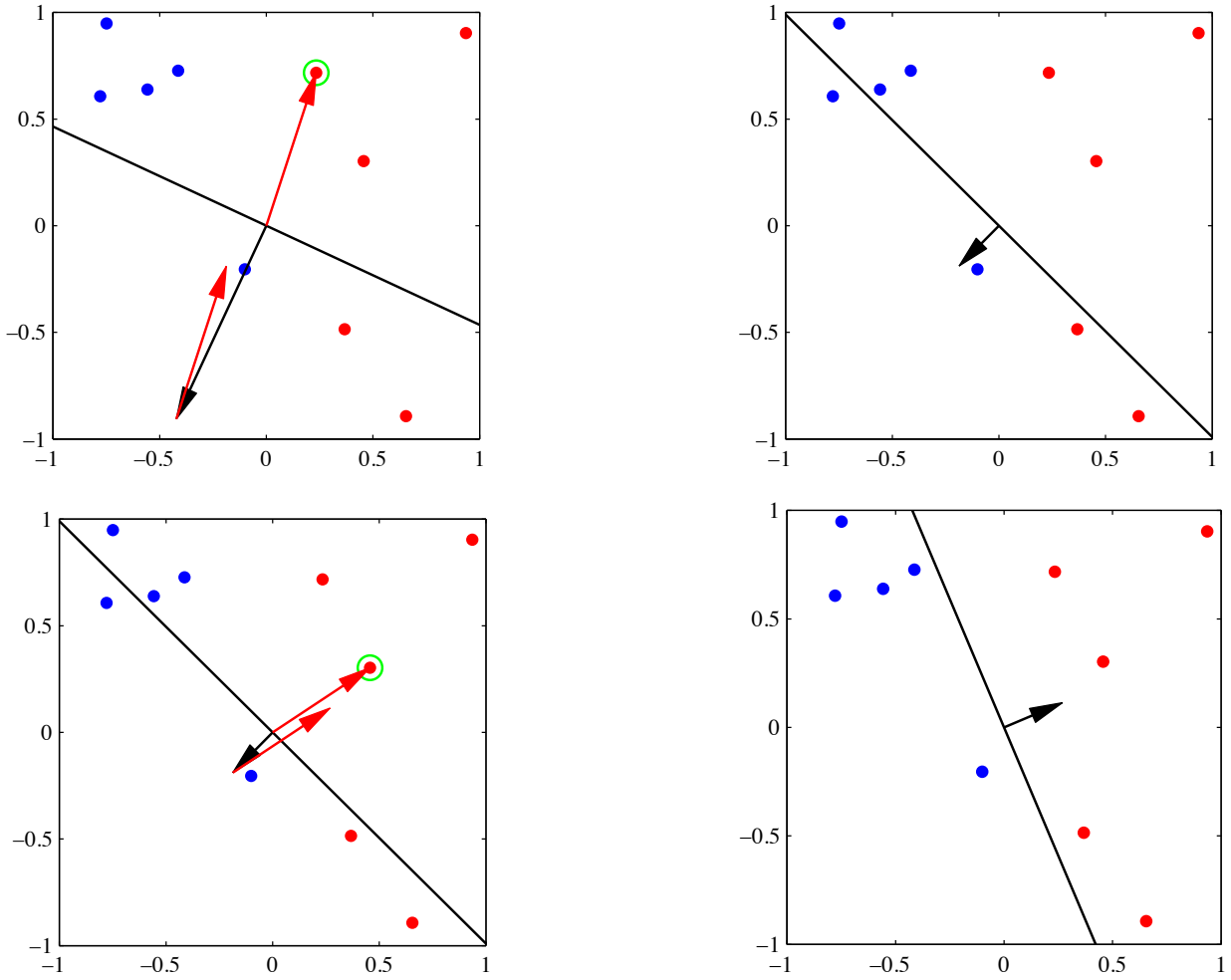
Процедура сходимости персептрона

- Добавьте к каждому вектору признаков дополнительный компонент со значением 1. Вес «смещения» этого компонента равен минус пороговому значению. Теперь о пороговом значении можно забыть.
- Выбирайте учебные случаи, используя любую политику, которая гарантирует, что каждый учебный случай будет выбран.
  - Если вывод правильный, оставьте его веса в покое.
  - Если выход равен 0, а должен быть 1, добавьте вектор признаков к вектору весов.
  - Если выход равен 1, а должен быть 0, вычтите вектор признаков из вектора веса.
- Это гарантирует нахождение набора весов, который даст правильный ответ на всем обучающем наборе, если такой набор существует.
- Нет необходимости выбирать скорость обучения.

Естественный способ попытаться доказать сходимость

- Очевидный подход — записать функцию ошибки и попытаться показать, что каждый шаг процедуры обучения уменьшает ошибку.
  - Для стохастического онлайн-обучения мы хотели бы показать, что каждый шаг уменьшает ожидаемую ошибку, где ожидание касается выбора обучающих случаев.
  - Это не может быть квадратичная ошибка, поскольку размер обновления не зависит от размера ошибки.
- В учебнике в качестве меры погрешности пытаются использовать сумму расстояний на неправильной стороне поверхности принятия решений.
  - В результате был сделан вывод о том, что процедура сходимости персептрона не гарантирует снижения общей ошибки *на каждом шаге*.
    - Это справедливо для данной функции ошибки, даже если существует набор весов, дающий правильный ответ для каждого случая обучения.

Вес и пространство данных



Сходимость алгоритма обучения персептрона, на которой показаны точки из двух классов {красного и синего) в двухмерном пространстве признаков $(\phi 1, \phi 2)$. Слева вверху показан начальный вектор параметров представленный в виде черной стрелки вместе с соответствующей границей решения (черная линия), где стрелка указывает на область принятия решения, которая классифицируется как принадлежащая к красному классу. Точка, обведенная зеленым кружком, классифицирована ошибочно, поэтому ее
вектор-функция добавляется к текущему вектору весов, что дает новую границу решения, показанную справа вверху. Слева внизу показана следующая ошибочная точка, обозначенная зеленым кружком, которую следует учесть, а ее вектор-функция снова добавляется к весовому вектору, давая границу решения, показанную справа внизу, где все точки классифицированы правильно

Лучший способ доказать сходимость
(используя выпуклость решений в весовом пространстве)

- Очевидный тип функции ошибок измеряет расхождение между целевыми значениями и выходными данными модели.
- Другой тип функции стоимости заключается в использовании квадрата расстояния между текущими весами и допустимым набором весов.
  - Используя эту функцию стоимости, мы можем показать, что каждый шаг процедуры уменьшает ошибку.
    - При условии, что существует набор допустимых весов.
- Используя этот тип функции стоимости, процедуру можно легко обобщить на более чем два класса, используя правило принятия решений MAX.

Почему процедура обучения работает

- Рассмотрим квадрат расстояния между любым удовлетворительным весовым вектором и текущим весовым вектором.
  - Каждый раз, когда персептрон совершает ошибку, алгоритм обучения уменьшает квадрат расстояния между текущим вектором веса и любым удовлетворительным вектором веса (если только он не пересекает плоскость принятия решений).
- Поэтому рассмотрим «вполне удовлетворительные» векторы веса, которые лежат в допустимой области с запасом, по крайней мере таким же большим, как и наибольшее обновление.
  - Каждый раз, когда персептрон совершает ошибку, квадрат расстояния до всех этих весовых векторов всегда уменьшается по крайней мере на квадрат длины наименьшего вектора обновления.

Чему персептроны не могут научиться

- Адаптивная часть персептрона не может даже определить, имеют ли два однобитовых признака одинаковое значение!

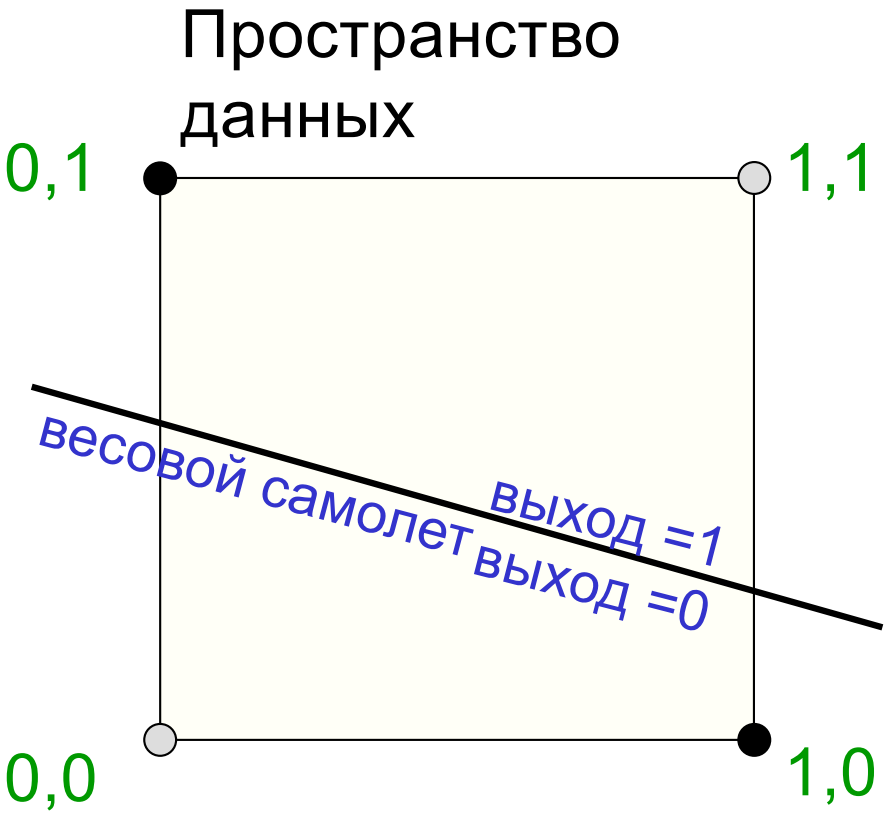
  Одинаковые : (1,1) → 1; (0,0) → 1

  Разные : (1,0) → 0; (0,1) → 0

- Четыре пары «признак-выход» дают четыре неравенства, которые невозможно удовлетворить:

$$w_1 + w_2 \geq \theta, \quad 0 \geq \theta$$

$$w_1 < \theta, \qquad w_2 < \theta$$



Положительные и отрицательные случаи не могут быть разделены плоскостью

Что умеют персептроны?

- Они могут решать задачи только в том случае, если закодированные вручную признаки преобразуют исходную задачу в линейно разделимую. Насколько это сложно?
- Задача проверки четности N-бит:
  - Требуется N признаков вида: Включены ли по крайней мере m бит?
  - Каждая функция должна учитывать все компоненты входных данных.
- Задача на двумерную связность
  - требуется экспоненциальное количество функций!

Задача проверки четности N-бит

- Существует простое решение, требующее N скрытых единиц.
  - Каждый скрытый блок вычисляет, включено ли более M входов.
  - Это линейно разделимая задача.
- Существует множество вариантов этого решения.
  - Этому можно научиться.
  - Он хорошо обобщает, если: $2^N >> N^2$