УПРАВЛЕНИЕ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ
Машинное обучение II

А.В. Макаренко
avm@rdcn.ru

Научно-исследовательская группа «Конструктивная Кибернетика»
Москва, Россия, www.rdcn.ru

Институт проблем управления РАН
Москва, Россия

Учебный курс – Лекция 3

05 марта 2020 г.

ИПУ РАН, Москва, Россия

Outline

Outline section

Основные обозначения

Даны мультимножества:

X – описаний объектов (характеристики, признаки, features);

R – решений алгоритма (ответы, метки, patterns, labels).

Существует, но неизвестна, целевая функция (target function): $\mathrm{G}' : \mathrm{X} \to \mathrm{R}$.

Логическая пара: $\grave{\mathrm{d}}_n = (\mathbf{x}_n, \mathbf{r}_n)$ – составляет n-й прецедент.

Необходимо найти алгоритм (решающую функцию, decision function):
$\mathrm{G} : \mathrm{X} \to \mathrm{R}$, которая восстанавливает оценку G'.

Подмножества прецедентов, выборки:

$\mathrm{D}^{\mathrm{Tr}} = \left\{(\mathbf{x}_n, \mathbf{r}_n)\right\}_{n=1}^{N_{\mathrm{Tr}}}$ – обучающая (train set),

$\mathrm{D}^{\mathrm{Ts}} = \left\{(\mathbf{x}_n, \mathbf{r}_n)\right\}_{n=1}^{N_{\mathrm{Ts}}}$ – тестовая (test set).

Исключение протечек данных (leaked data):

$\mathrm{X}^{\mathrm{Tr}} \cap \mathrm{X}^{\mathrm{Ts}} = \emptyset$ – необходимое условие.

Множество W – допустимые параметры алгоритма G.

Функционал Q – оценивание качества функционирования алгоритма G.

Признаки

Введём отображение: $\mathrm{U}' : \mathrm{X} \to \mathrm{V}$, где $\mathrm{V}^{(m)}$ – множество допустимых значений m-го признака, $m = \overline{1, M}$.

В зависимости от структуры множества $\mathrm{V}^{(m)}$, признаки делятся на следующие типы:

- $\mathrm{V}^{(m)} = \{0, 1\}$ – бинарный.
- $\mathrm{V}^{(m)} \subset \mathbb{Z}$, $|\mathrm{V}| < \infty$ – именованный (номинальный).
- $\mathrm{V}^{(m)} \subset \mathbb{Z}$, $|\mathrm{V}| < \infty$, $v_i \prec v_{i+1}$ – порядковый.
- $\mathrm{V}^{(m)} \subset (\mathbb{Z}, \mathbb{R}, \mathbb{C})$, $v_i < v_j$, $\mathrm{d}(v_i, v_j)$ – количественный.

Вектор $\mathbf{x}_j = \left[x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(M)}\right]$ называют признаковым описанием j-го наблюдаемого объекта.

Если тип всех признаков $m = \overline{1, M}$ одинаков, то признаковое описание называют однородным, иначе гетерогенным.

Наиболее распространённым способом представления множества X в прикладных задачах является матрица объектов–признаков:

$$\begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \ldots & x_1^{(M)} \\ x_2^{(1)} & x_2^{(2)} & \ldots & x_2^{(M)} \\ \ldots & \ldots & \ldots & \ldots \\ x_N^{(1)} & x_N^{(2)} & \ldots & x_N^{(M)} \end{bmatrix}.$$

Задача Классификации

Множество допустимых значений R – выражается в шкале Наименований (или Порядка):

$$\mathrm{U}' : \mathrm{R} \to \mathrm{V}, \quad \mathrm{V}^{(m)} \subset \mathbb{Z}, \quad |\mathrm{V}| < \infty, \quad \left(v_i^{(m)} \prec v_{i+1}^{(m)}\right).$$

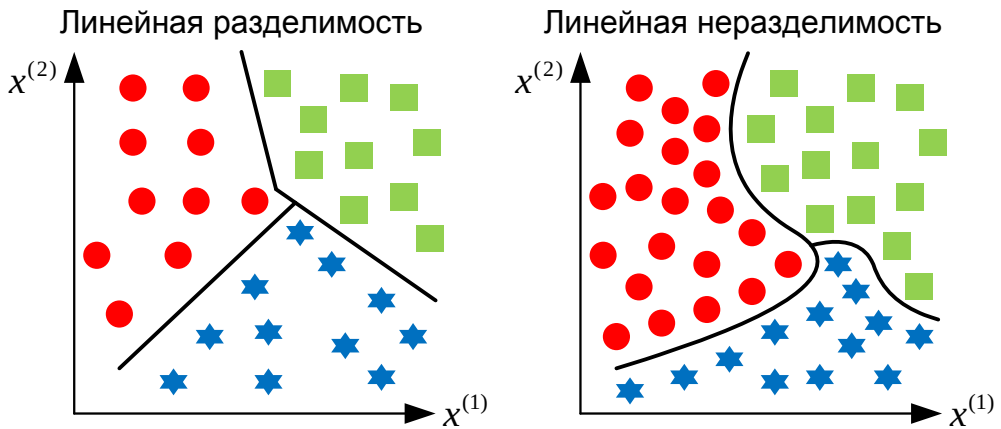
Дано обучающее множество

$$\mathrm{D}^{\mathrm{Tr}} \neq \emptyset.$$

Структура V – известна, её тип определяет тип классификатора:

- $\mathrm{V}^{(m)} = \{0, 1\}$ – бинарный.
- $\mathrm{V}^{(m)} = \{1, \ldots, M\}$ – многоклассовый на M непересекающихся классов.
- $\mathrm{V}^{(m)} = \{0, 1\}^M$ – многоклассовый на M пересекающихся классов.

Два фундаментальных типа классифицируемости X:



Задача Кластеризации

Множество допустимых значений R – выражается в шкале Наименований:

$$\mathrm{U}' : \mathrm{R} \to \mathrm{V}, \quad \mathrm{V} \subset \mathbb{N}, \quad |\mathrm{V}| < \infty.$$

Под $v_i \in \mathrm{V}$ понимается номер кластера.

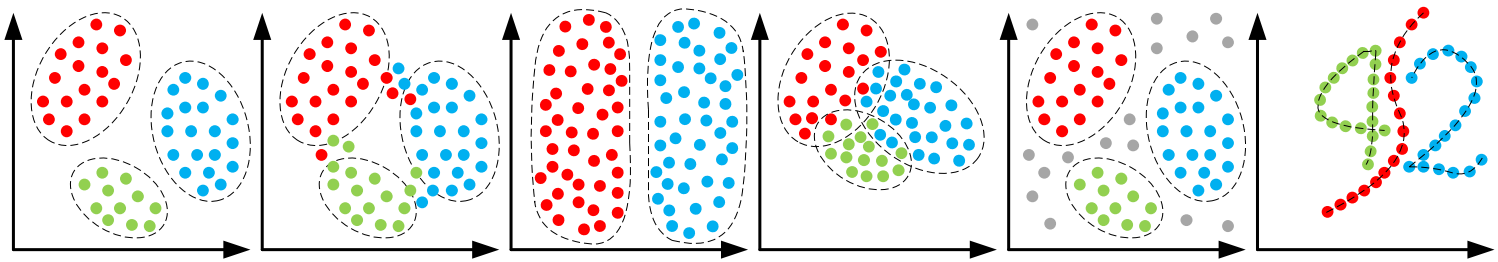
Обучающее множество – отсутствует:

$$\mathrm{D}^{\mathrm{Tr}} \equiv \emptyset.$$

Структура V – в общем случае неизвестна. Априорные предположения:

- Возможно указать число кластеров $\hat{M}$, $m = \overline{1, \hat{M}}$.
- Возможно указать координаты центров кластеров $\hat{\mathbf{x}}_{\mathrm{c}}^{(m)}$.

Типы кластерных многообразий:



Задача Регрессии

Множество допустимых значений R – выражается в шкале Интервалов, Отношений или Абсолютной:

$$\mathrm{U}' : \mathrm{R} \to \mathrm{V}, \quad \mathrm{V}^{(m)} \subset (\mathbb{R}, \mathbb{C})^N, \quad v_i < v_j, \quad \mathrm{d}(v_i, v_j).$$

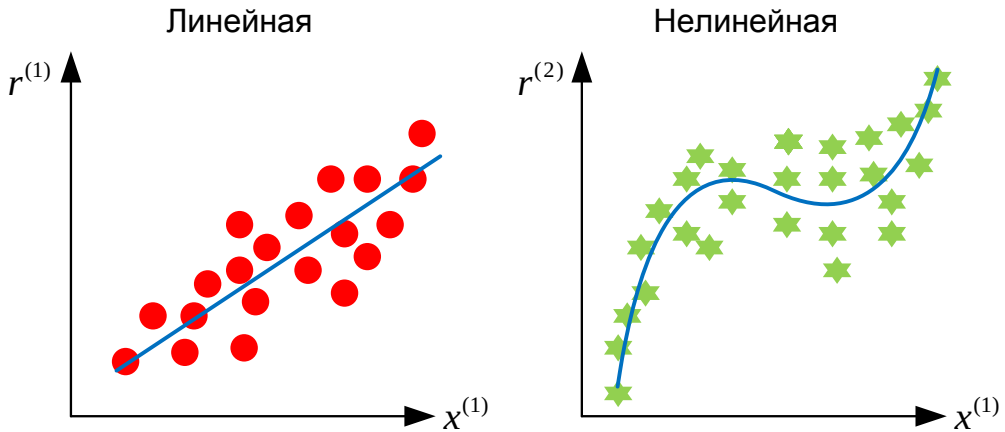
Дано обучающее множество

$$\mathrm{D}^{\mathrm{Tr}} \neq \emptyset.$$

Структура V – известна (это, как правило, результаты случайных измерений).

$\mathrm{X} \xrightarrow{\mathrm{G}} \mathrm{R}$ – регрессионная модель (функция регрессии).

Два фундаментальных типа регрессионных моделей:



Задача Редукции Размерности

$\mathbf{x}_{\text{src}}$, $\mathbf{x}_{\text{red}}$ – исходное и редуцированное признаковое описание объектов.

Обучающее множество – отсутствует:

$$\mathrm{D}^{\mathrm{Tr}} \equiv \emptyset.$$

Множество R – не требуется.

Суть задачи $\mathrm{X}_{\text{src}} \xrightarrow{\mathrm{G}} \mathrm{X}_{\text{red}}$ (manifold learning):

$$\dim \mathbf{x}_{\text{src}} > \dim \mathbf{x}_{\text{red}},$$

при том, что:

$$\mathrm{Q}^* \Big[\mathrm{R}_{\text{red}} | \mathrm{X}_{\text{red}} \Big] \geqslant \mathrm{Q}^* \Big[\mathrm{R}_{\text{src}} | \mathrm{X}_{\text{src}} \Big] - \epsilon, \quad \epsilon \to +0,$$

где Q^* – качество решения сопряжённой задачи (классификация, кластеризация, регрессия, и т.п.).

Назначение редуцирующей модели G:

- Снижение вычислительной нагрузки при функционировании моделей машинного обучения.
- Повышение качества функционирования моделей.
- Когнитивная визуализация, исследование локальной и глобальной структур данных.

Общие положения

Рассмотрим алгоритм $\mathrm{G} : \mathrm{X} \times \mathrm{W} \to \mathrm{R}$, где W – множество допустимых значений вектора параметра $\mathbf{w}$. Множество W также называют пространством параметров или пространством поиска.

Тогда моделью алгоритмов будет параметрическое семейство функций:

$$\mathrm{A} = \{\mathrm{G}(\mathbf{x}, \mathbf{w}) | \mathbf{w} \in \mathrm{W}\}.$$

Метод обучения (learning algorithm) – это отображение вида:

$$\mu : \left(\mathrm{X} \times \mathrm{R}\right)^{N_{\mathrm{ds}}} \to \mathrm{A},$$

которое произвольной конечной выборке $\mathrm{D}^{\mathrm{ds}} = \left\{(\mathbf{x}_n, \mathbf{r}_n)\right\}_{n=1}^{N_{\mathrm{ds}}}$ ставит в соответствие некоторый алгоритм $a \in \mathrm{A}$, фактически находит оптимальное значение вектора параметров $\tilde{\mathbf{w}}$: $a \equiv \mathrm{G}(\mathbf{x}, \tilde{\mathbf{w}})$.

В машинном обучении по прецедентам выделяют два основных этапа:

- Собственно обучение (training) – построение алгоритма $\mathrm{G}(\mathbf{x}, \tilde{\mathbf{w}})$ по выборке D^{ds}.
- Применение (testing) – эксплуатация алгоритма $\mathrm{G}(\mathbf{x}, \tilde{\mathbf{w}})$.

Обучение с учителем

Основные области применения:

- Классификация.
- Регрессия.

Основной момент: доступность обучающего множества $\mathrm{D}^{\mathrm{Tr}} \neq \emptyset$.

Обучение без учителя

Основные области применения:

- Кластеризация.
- Manifold Learning.

Основной момент: отсутствие обучающего множества $\mathrm{D}^{\mathrm{Tr}} \equiv \emptyset$.

Обучение с подкреплением

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ (reinforcement learning) – один из способов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или её модель.

Обучение с подкреплением

Q-ОБУЧЕНИЕ (Q-learning) – метод, применяемый в искусственном интеллекте при агентном подходе. Относится к экспериментам вида обучение с подкреплением. На основе получаемого от среды вознаграждения агент формирует функцию полезности Q, что впоследствии даёт ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Одно из преимуществ Q-обучения в том, что оно в состоянии сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде марковского процесса принятия решений.

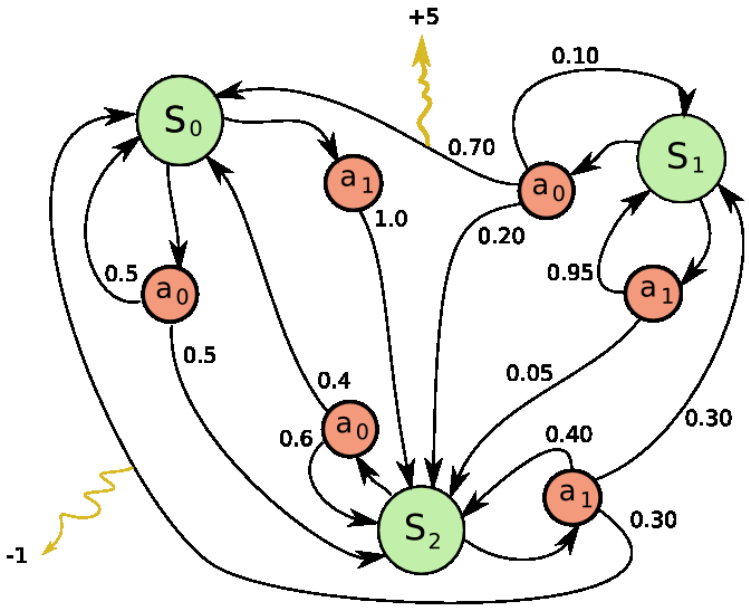
Обучение с подкреплением

Q-ОБУЧЕНИЕ (Q-learning) – метод, применяемый в искусственном интеллекте при агентном подходе. Относится к экспериментам вида обучение с подкреплением. На основе получаемого от среды вознаграждения агент формирует функцию полезности Q, что впоследствии даёт ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Одно из преимуществ Q-обучения в том, что оно в состоянии сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде марковского процесса принятия решений.

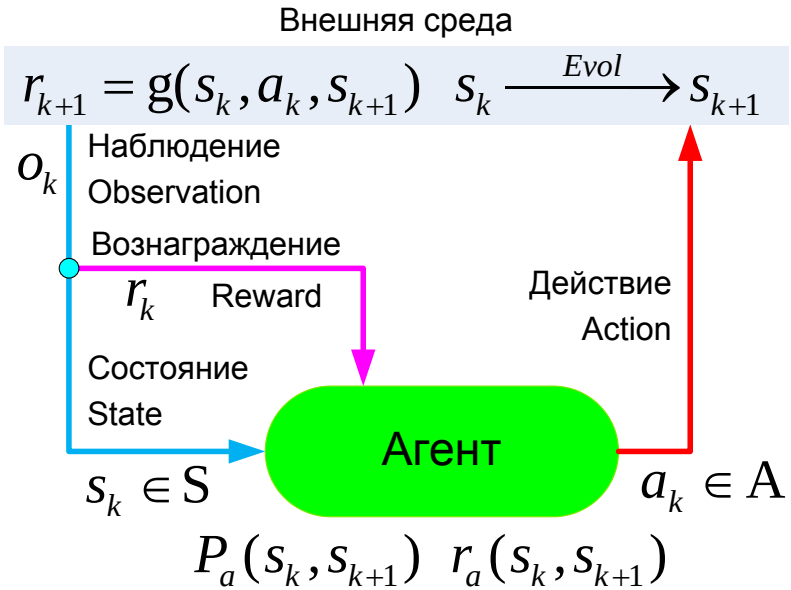
МАРКОВСКИЙ ПРОЦЕСС ПРИНЯТИЯ РЕШЕНИЙ (Markov decision process, MDP) – спецификация задачи последовательного принятия решений для полностью наблюдаемой среды с марковской моделью перехода и дополнительными вознаграждениями. Служит математической основой для того, чтобы смоделировать принятие решения в ситуациях, где результаты частично случайны и частично под контролем лица, принимающего решения.

Обучение с подкреплением

Q-ОБУЧЕНИЕ (Q-learning) – метод, применяемый в искусственном интеллекте при агентном подходе. Относится к экспериментам вида обучение с подкреплением. На основе получаемого от среды вознаграждения агент формирует функцию полезности Q, что впоследствии даёт ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Одно из преимуществ Q-обучения в том, что оно в состоянии сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде марковского процесса принятия решений.



Автор: MistWiz



Функционал качества

ФУНКЦИЯ ПОТЕРЬ (loss function) – это неотрицательная функция $L(\tilde{\mathbf{r}}, \mathbf{r})$, характеризующая величину ошибки алгоритма $\tilde{\mathbf{r}}_n = a(\mathbf{x}_n)$ на n-м прецеденте $\grave{\mathrm{d}}_n$. Если $L(\tilde{\mathbf{r}}, \mathbf{r}) = 0$, то ответ $a(\mathbf{x}_n)$ называется корректным.

Напоминание: $a \equiv \mathrm{G}(\mathbf{x}, \tilde{\mathbf{w}})$, $\grave{\mathrm{d}}_n = (\mathbf{x}_n, \mathbf{r}_n)$.

Функционал качества

ФУНКЦИЯ ПОТЕРЬ (loss function) – это неотрицательная функция $L(\tilde{\mathbf{r}}, \mathbf{r})$, характеризующая величину ошибки алгоритма $\tilde{\mathbf{r}}_n = a(\mathbf{x}_n)$ на n-м прецеденте $\grave{\mathrm{d}}_n$. Если $L(\tilde{\mathbf{r}}, \mathbf{r}) = 0$, то ответ $a(\mathbf{x}_n)$ называется корректным.

Напоминание: $a \equiv \mathrm{G}(\mathbf{x}, \tilde{\mathbf{w}})$, $\grave{\mathrm{d}}_n = (\mathbf{x}_n, \mathbf{r}_n)$.

Функционал качества алгоритма a на произвольной конечной выборке D^{ds}

$$\mathrm{Q}\big(a, \mathrm{D}^{\mathrm{ds}}\big) = \frac{1}{N^{ds}} \sum_{n=1}^{N^{ds}} L\big(a(\mathbf{x}_n), \mathbf{r}_n\big),$$

называют также функционалом средних потерь или эмпирическим риском, так как он вычисляется по эмпирическим данным D^{ds}.

Функционал качества

ФУНКЦИЯ ПОТЕРЬ (loss function) – это неотрицательная функция $L(\tilde{\mathbf{r}}, \mathbf{r})$, характеризующая величину ошибки алгоритма $\tilde{\mathbf{r}}_n = a(\mathbf{x}_n)$ на n-м прецеденте $\grave{\mathrm{d}}_n$. Если $L(\tilde{\mathbf{r}}, \mathbf{r}) = 0$, то ответ $a(\mathbf{x}_n)$ называется корректным.

Напоминание: $a \equiv \mathrm{G}(\mathbf{x}, \tilde{\mathbf{w}})$, $\grave{\mathrm{d}}_n = (\mathbf{x}_n, \mathbf{r}_n)$.

Функционал качества алгоритма a на произвольной конечной выборке D^{ds}

$$\mathrm{Q}\big(a, \mathrm{D}^{\mathrm{ds}}\big) = \frac{1}{N^{ds}} \sum_{n=1}^{N^{ds}} L\big(a(\mathbf{x}_n), \mathbf{r}_n\big),$$

называют также функционалом средних потерь или эмпирическим риском, так как он вычисляется по эмпирическим данным D^{ds}.

МЕТОД МИНИМИЗАЦИИ ЭМПИРИЧЕСКОГО РИСКА (empirical risk minimization, ERM) – один из наиболее распространённых подходов к обучению алгоритмов по прецедентам. Он заключается в том, чтобы в заданной модели алгоритмов $\mathrm{A} = \big\{\mathrm{G}(\mathbf{x}, \mathbf{w}) | \mathbf{w} \in \mathrm{W}\big\}$ найти алгоритм a, минимизирующий среднюю ошибку на обучающей выборке:

$$a = \arg\min_{a \in \mathrm{A}} \mathrm{Q}\big(a, \mathrm{D}^{\mathrm{Tr}}\big)$$

Следует отметить, что ERM сводит задачу обучения к оптимизации и задача может быть решена численными методами оптимизации.

Переобучение алгоритма

Чем плохо $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Tr}}\big) = 0$?

Переобучение алгоритма

Чем плохо $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Tr}}\big) = 0$?

Как минимум тем, что ничего не известно о величине $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Ts}}\big)$.

Переобучение алгоритма

Чем плохо $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Tr}}\big) = 0$?

Как минимум тем, что ничего не известно о величине $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Ts}}\big)$.

Величина $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Tr}}\big) = 0$ и даже $\mathrm{Q}\big(a,\, \mathrm{D}^{\mathrm{Ts}}\big) = 0$ – в общем случае не гарантируют «требуемого решения задачи».

Переобучение алгоритма

ПЕРЕОБУЧЕНИЕ (переподгонка, overfitting) – явление в машинном обучении и статистике, при котором построенная модель a хорошо объясняет примеры из обучающей выборки D^{Tr}, но относительно плохо работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки D^{Ts}). Как правило, переобучение возникает при использовании избыточно сложных моделей.

Переобучение алгоритма

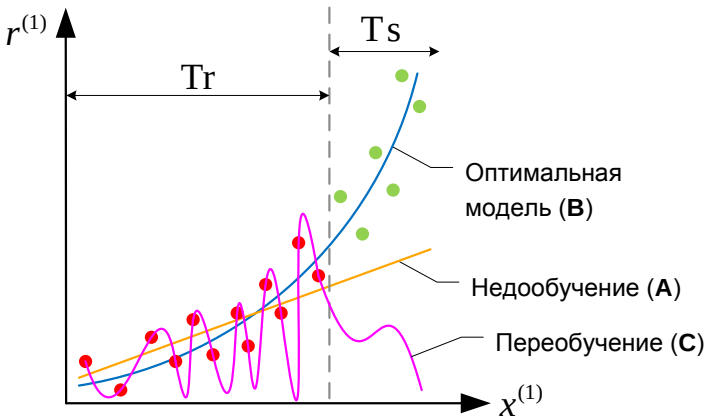
ПЕРЕОБУЧЕНИЕ (переподгонка, overfitting) – явление в машинном обучении и статистике, при котором построенная модель a хорошо объясняет примеры из обучающей выборки D^{Tr}, но относительно плохо работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки D^{Ts}). Как правило, переобучение возникает при использовании избыточно сложных моделей.

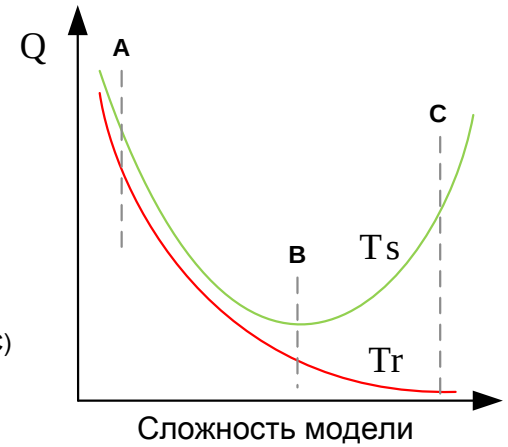
НЕДООБУЧЕНИЕ – явление в машинном обучении по прецедентам, когда алгоритм обучения μ не обеспечивает достаточно малой величины средней ошибки Q на обучающей выборке D^{Tr}. Как правило, недообучение возникает при использовании недостаточно сложных моделей.

Переобучение алгоритма

ПЕРЕОБУЧЕНИЕ (переподгонка, overfitting) – явление в машинном обучении и статистике, при котором построенная модель a хорошо объясняет примеры из обучающей выборки D^{Tr}, но относительно плохо работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки D^{Ts}). Как правило, переобучение возникает при использовании избыточно сложных моделей.

НЕДООБУЧЕНИЕ – явление в машинном обучении по прецедентам, когда алгоритм обучения μ не обеспечивает достаточно малой величины средней ошибки Q на обучающей выборке D^{Tr}. Как правило, недообучение возникает при использовании недостаточно сложных моделей.





Успешное обучение требует не только умения запоминать (memorization), но и способности обобщать (generalization).

Регуляризация

Один из эффективных способов борьбы с переобучением модели – это регуляризация.

Регуляризация

РЕГУЛЯРИЗАЦИЯ – (в машинном обучении, статистике, теории обратных задач) – метод добавления некоторой дополнительной информации к условию с целью решить некорректно поставленную задачу или предотвратить переобучение. Эта информация часто имеет вид штрафа за сложность модели. Например, это могут быть ограничения гладкости результирующей функции или ограничения по норме векторного пространства.

Регуляризация

РЕГУЛЯРИЗАЦИЯ – (в машинном обучении, статистике, теории обратных задач) – метод добавления некоторой дополнительной информации к условию с целью решить некорректно поставленную задачу или предотвратить переобучение. Эта информация часто имеет вид штрафа за сложность модели. Например, это могут быть ограничения гладкости результирующей функции или ограничения по норме векторного пространства.

Наиболее употребимые виды регуляризации ($\lambda \geqslant 0$ – параметр регуляризации):

L_1 – lasso regression (разряженные модели, отбор признаков):

$$L_1 = \lambda \sum_{i=1}^{D} |w_i|.$$

L_2 – ridge regression (Регуляризация Тихонова):

$$L_2 = \lambda \sum_{i=1}^{D} w_i^2.$$

Elastic Net – эластичная сеть:

$$L_{\text{x}}|\beta = \beta\, L_1 + (1-\beta)\, L_2.$$

$$L'(\tilde{\mathbf{r}},\, \mathbf{r}) = L(\tilde{\mathbf{r}},\, \mathbf{r}) + L_{\circ}$$

Варианты проведения анализа данных

Полу-ручная обработка экспертом малых выборок данных:

Ограничения:

- Низкая производительность труда и повторяемость результатов.
- Высокий уровень субъективизма в результатах (низкие уровни стат. значимости).
- Низкая обнаружительная способность – пропуск значимых эффектов.

Варианты проведения анализа данных

Полу-ручная обработка экспертом малых выборок данных:

Ограничения:

- Низкая производительность труда и повторяемость результатов.
- Высокий уровень субъективизма в результатах (низкие уровни стат. значимости).
- Низкая обнаружительная способность – пропуск значимых эффектов.

Автоматизированная обработка средних и больших выборок данных:

Ограничения:

- Квалификация специалиста в области параллельного программирования.
- Существенные затраты на сбор релевантных данных.
- Существенные затраты на разработку аналитических инструментов.

Варианты проведения анализа данных

Полу-ручная обработка экспертом малых выборок данных:

Ограничения:

- Низкая производительность труда и повторяемость результатов.
- Высокий уровень субъективизма в результатах (низкие уровни стат. значимости).
- Низкая обнаружительная способность – пропуск значимых эффектов.

Автоматизированная обработка средних и больших выборок данных:

Ограничения:

- Квалификация специалиста в области параллельного программирования.
- Существенные затраты на сбор релевантных данных.
- Существенные затраты на разработку аналитических инструментов.

Тотальный автоматический скрининг всех доступных наборов данных:

Ограничения:

- Высокие затраты на интегрирование разнородных данных.
- Высокие затраты на разработку и поддержание аналитической системы.
- Высокий риск получения результата: «гора родила мышь».

Проблематика построения аналитических систем

Основные направления задач требующих решения:

- Высоко-производительные вычислительные системы.
- Специализированные математические методы.
- Специализированное алгоритмическое и программное обеспечение.

Проблематика построения аналитических систем

Основные направления задач требующих решения:

- Высоко-производительные вычислительные системы.
- Специализированные математические методы.
- Специализированное алгоритмическое и программное обеспечение.

Основные классы задач требующих решения:

- Сбор, структурирование и хранение первичных разнородных данных.
- Выявление паттернов, идентификация структурных и динамических свойств изучаемых систем и процессов.
- Понижение размерности данных и когнитивная визуализация.
- Модели предметной области для возможностей изучения систем, явлений и процессов, прогнозирования и управления.

Граф обработки (базовые положения)

ГРАФ ОБРАБОТКИ – набор логически связанных вычислительных процедур, которые должны быть применены к каждому объекту (набору) данных с целью решения задач анализа и достижения поставленных целей исследования.

$$\Gamma = \langle \mathrm{V}\,\mathrm{E} \rangle, \quad \mathrm{V} - \text{процедуры}, \quad \mathrm{E} - \text{данные}.$$

Граф обработки (базовые положения)

ГРАФ ОБРАБОТКИ – набор логически связанных вычислительных процедур, которые должны быть применены к каждому объекту (набору) данных с целью решения задач анализа и достижения поставленных целей исследования.

$$\Gamma = \langle \mathrm{V}\, \mathrm{E} \rangle, \quad \mathrm{V} - \text{процедуры}, \quad \mathrm{E} - \text{данные}.$$

Два полюса вычислительных процедур:

Граф обработки (базовые положения)

ГРАФ ОБРАБОТКИ – набор логически связанных вычислительных процедур, которые должны быть применены к каждому объекту (набору) данных с целью решения задач анализа и достижения поставленных целей исследования.

$$\Gamma = \langle \mathrm{V}\, \mathrm{E} \rangle, \quad \mathrm{V} - \text{процедуры}, \quad \mathrm{E} - \text{данные}.$$

Два полюса вычислительных процедур:

Изолированная по данным – для выполнения не требуется доступ к данным смежных объектов (размер документа, частота слов в документе, фурье-преобразование сигнала с элемента ФАР, и т.п.) – хорошо реализуется через модель MapReduce, удобна для распараллеливания и конвейеризации.

Граф обработки (базовые положения)

ГРАФ ОБРАБОТКИ – набор логически связанных вычислительных процедур, которые должны быть применены к каждому объекту (набору) данных с целью решения задач анализа и достижения поставленных целей исследования.

$$\Gamma = \langle \mathrm{V}\, \mathrm{E} \rangle, \quad \mathrm{V} - \text{процедуры}, \quad \mathrm{E} - \text{данные}.$$

Два полюса вычислительных процедур:

Изолированная по данным – для выполнения не требуется доступ к данным смежных объектов (размер документа, частота слов в документе, фурье-преобразование сигнала с элемента ФАР, и т.п.) – хорошо реализуется через модель MapReduce, удобна для распараллеливания и конвейеризации.

Полносвязная по данным – для выполнения требуется доступ к данным всех объектов выборки (поиск цитирований, построение компонент связности графа, пространственно-временная фильтрация сигнала, и т.п.) – требует моделей матрично-графовых вычислений, сложно распараллеливается, является высоконагруженной вычислительной задачей.

Граф обработки (базовые положения)

ГРАФ ОБРАБОТКИ – набор логически связанных вычислительных процедур, которые должны быть применены к каждому объекту (набору) данных с целью решения задач анализа и достижения поставленных целей исследования.

$$\Gamma = \langle \mathrm{V}\, \mathrm{E} \rangle, \quad \mathrm{V} - \text{процедуры}, \quad \mathrm{E} - \text{данные}.$$

Закон Амдала – иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей: «В случае, когда задача разделяется на несколько частей, суммарное время её выполнения на параллельной системе не может быть меньше времени выполнения самого длинного фрагмента».

$$V_{\mathrm{prl}} = \frac{1}{\alpha + \dfrac{1-\alpha}{p}},$$

V_{prl} – ускорение, p – число узлов (workers), α – доля последовательных вычислений от всего объёма вычислений.

Типы и назначения моделей

ОПИСАНИЕ – формализованное представление изучаемого явления.

Типы и назначения моделей

ОПИСАНИЕ – формализованное представление изучаемого явления.

ОБЪЯСНЕНИЕ – понимание изучаемого явления.

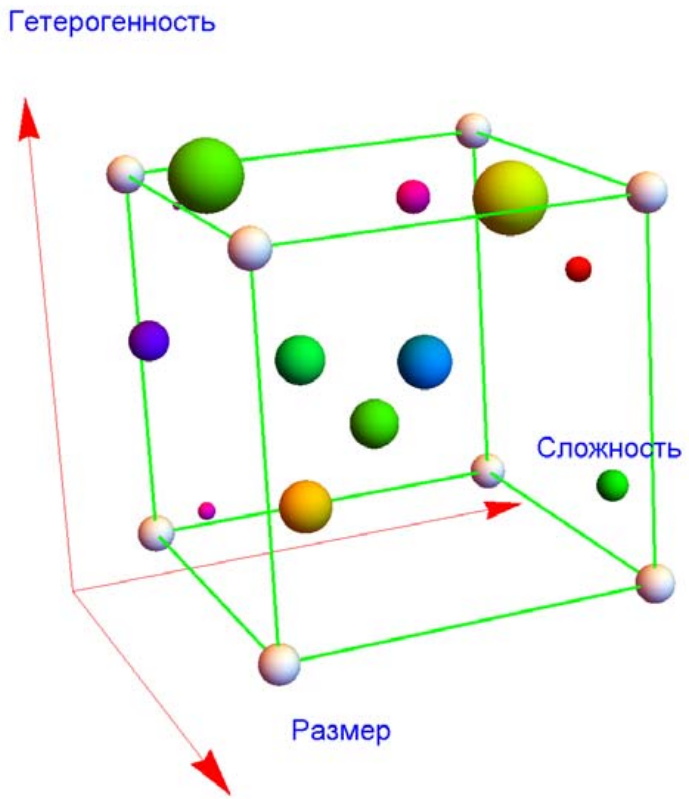
Типы и назначения моделей

ОПИСАНИЕ – формализованное представление изучаемого явления.

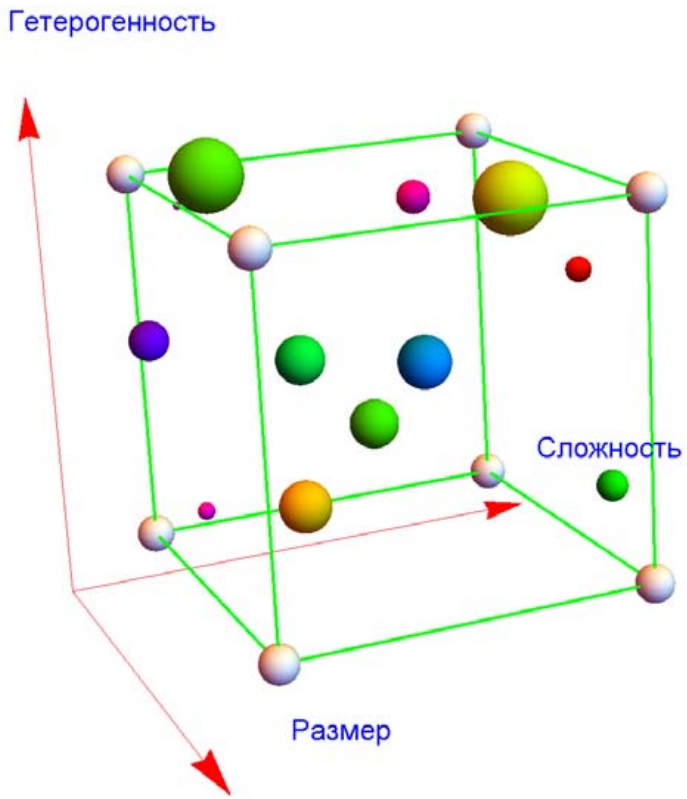
ОБЪЯСНЕНИЕ – понимание изучаемого явления.

ПРЕДСКАЗАНИЕ – выход нового знания.

Классификация задач Data Science

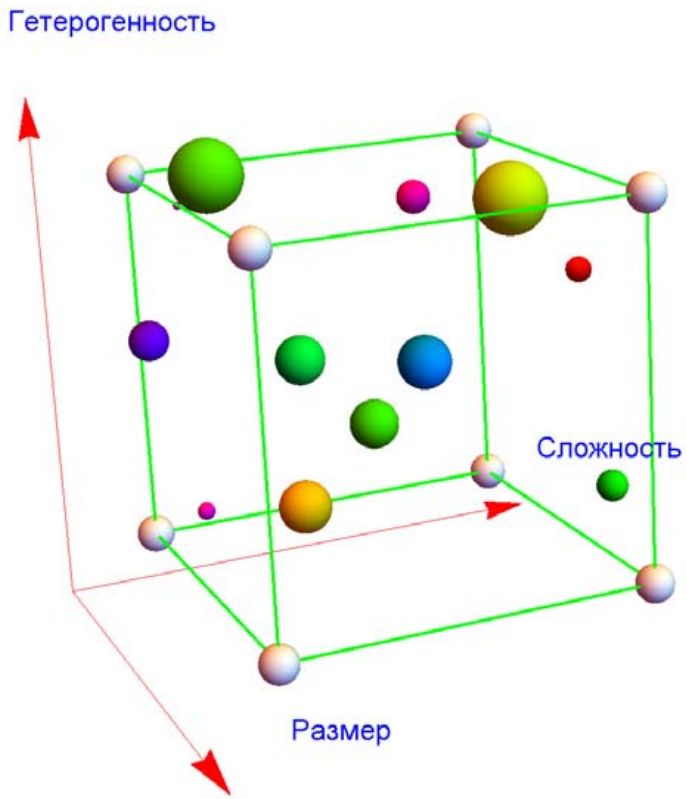


Классификация задач Data Science



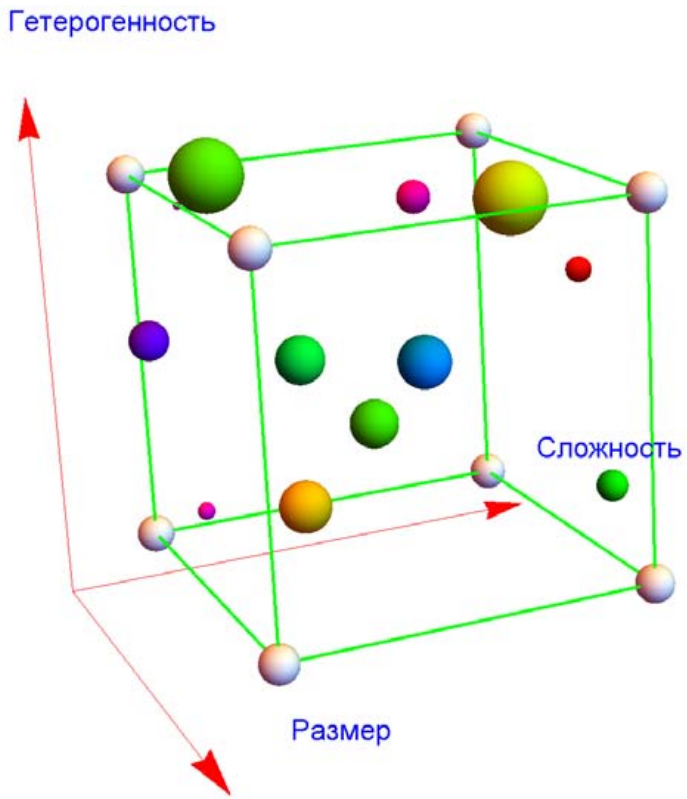
РАЗМЕР ЗАДАЧИ – определяется количеством операций и/или объёмом данных (на уровне размерности вектора и количества объектов), требуемых для решения задачи.

Классификация задач Data Science



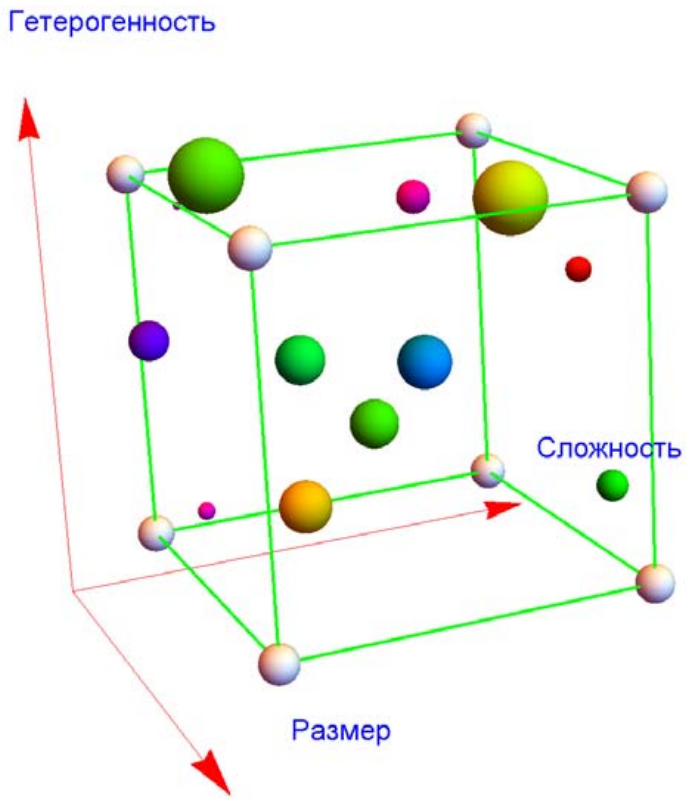
СЛОЖНОСТЬ ЗАДАЧИ – зависит от существования «вычислимого» алгоритма решения задачи и его переносимости на компьютерные платформы.

Классификация задач Data Science



ГЕТЕРОГЕННОСТЬ ЗАДАЧИ – определяется количеством разделов математики требующихся для решения задачи и разнородностью и/или неструктурированностью входных данных.

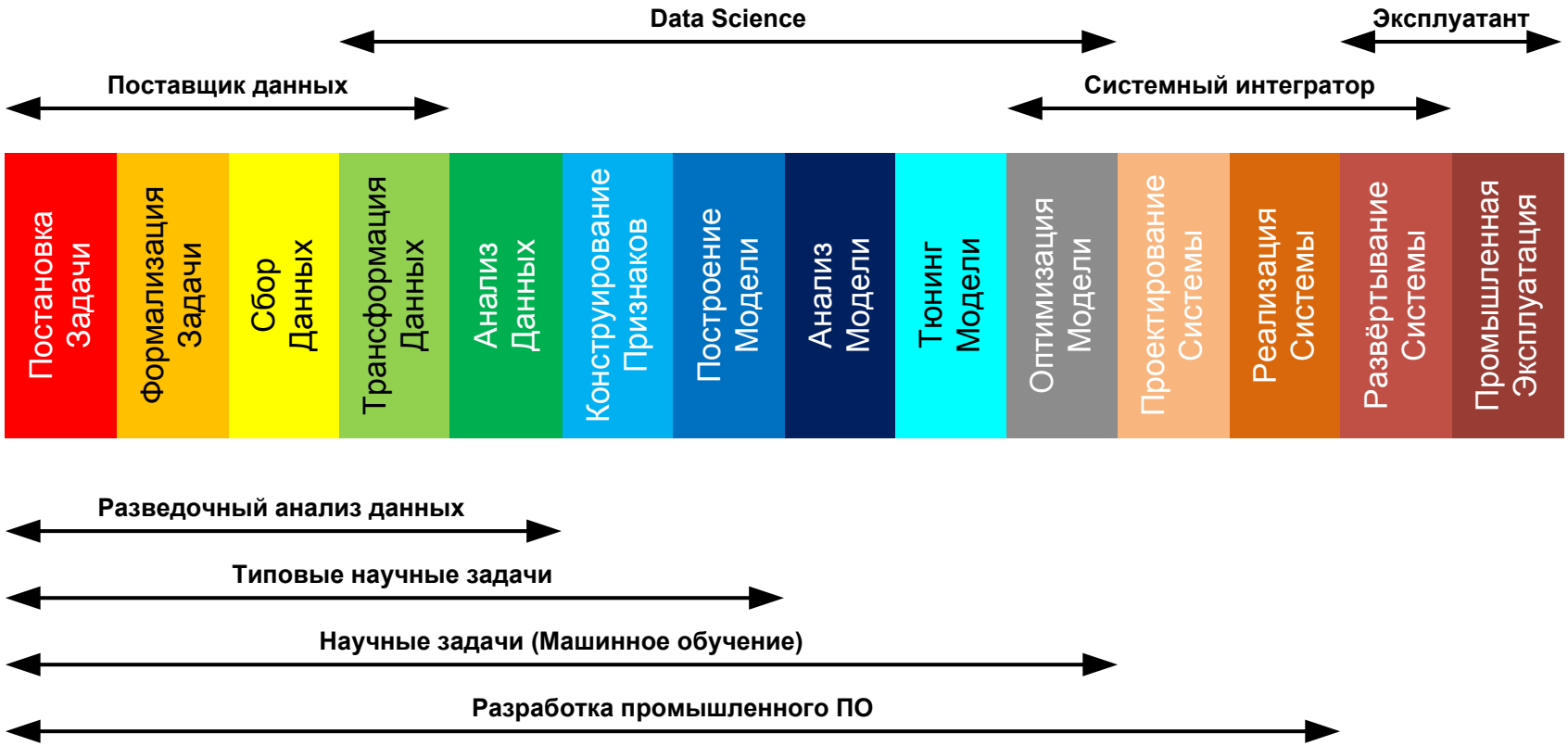
Классификация задач Data Science



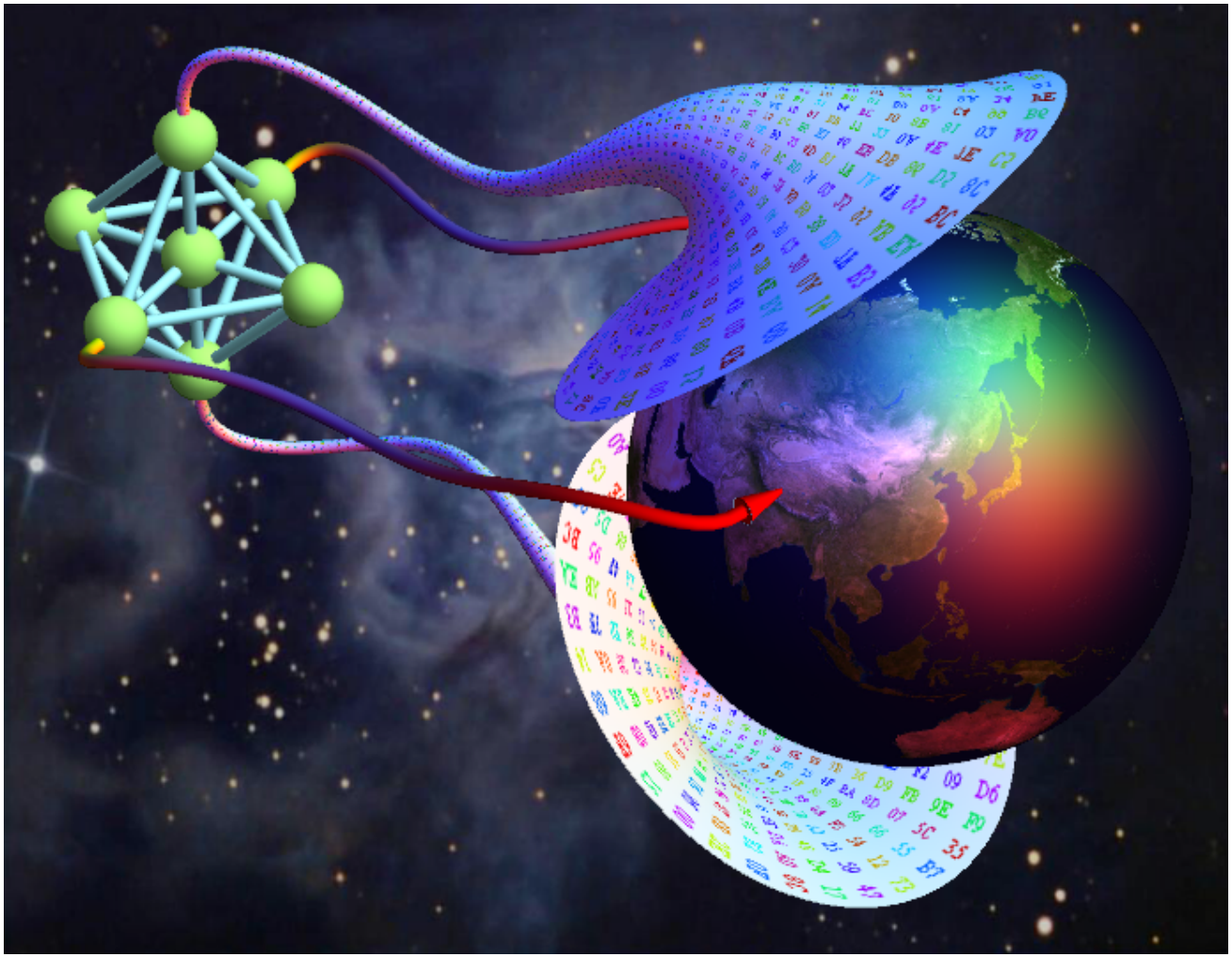
Важный контекст задач:

- *Медианные* – типовые потребности большинства Аналитиков.
- *Экстремальные* – ~~за гранью добра и зла~~ – редкие и специфичные задачи.

Конвейер Data Science



Смена технологических парадигм



Андрей Макаренко: От Network-Centric к Big Data – смена моды или закономерное развитие технологий?

Outline section

Контрольная работа

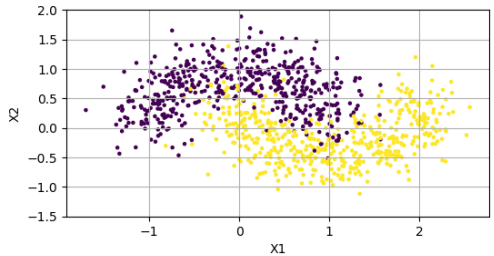
Задание для слушателей:

1. Изучить понятие confusion matrix (матрица ошибок), а также смысл мер: Precision, Recall, F_1; для случая бинарного классификатора.

2. Дескриптивная статистика:

Изучить внутренний датасет:

```
from sklearn.datasets import make_moons
X, y = make_moons(1000, noise = 0.275)
```



3. Набор `(X, y)` разбить на обучающую и тестовую выборки в соотношении 75/25. Изучить на нём функционирование следующих классификаторов (оформить Jupyter Notebook):

   - `from sklearn.linear_model import LogisticRegression`
   - `from sklearn.neighbors import KNeighborsClassifier`
   - `from sklearn.neighbors import KNeighborsClassifier`
   - `from sklearn.tree import DecisionTreeClassifier`
   - `from sklearn.ensemble import RandomForestClassifier`
   - `from sklearn.svm import SVC`