Снижение размерности

RS School Machine Learning course

Зачем нужно снижать размерность

- Обычно наблюдения даны в $\mathbb{R}^d$.
- «Проклятие размерности»: при больших d падает эффективность алгоритмов и перестаёт работать геометрическая интуиция.
  - У многих классов моделей время обучения зависит линейно от d
  - Разреженность данных: желательно, чтобы наблюдений в обучающей выборке было хотя бы в несколько раз больше d
  - Евклидово расстояние теряет чувствительность с ростом d
- Чтобы справиться с «проклятием размерности», данные отображают в $\mathbb{R}^{d'}$, $d' \ll d$.
- Когда $d' = 2$, можно визуализировать на плоскости.

Методы снижения размерности

- Отбор признаков (feature selection):
  - по статистическим свойствам
  - по скоррелированности друг с другом
  - по предсказательной силе модели на подмножестве признаков...
- Выделение признаков (feature extraction) – сегодня о нём.
- Линейные методы: искомое отображение $\mathbb{R}^d \to \mathbb{R}^{d'}$ можно представить матрицей $d \times d'$.
  - ▷ Метод главных компонент
- Нелинейные методы: отображение из произвольного метрического пространства, не обязательно записывается в явном виде.
  - ▷ Многомерное шкалирование
  - ▷ t-SNE
  - ▷ UMAP

Метод главных компонент

- Центрируем данные: вычтем среднее по каждому признаку.
- Найдём такую прямую, проходящую через ноль, проекции точек на которую дают наибольшую дисперсию.
- Среди всех перпендикуляров к ней опять найдём такую, что проекции точек на неё дают наибольшую дисперсию.
- ...И так далее, пока не наберётся d'.
- Можно продолжать до размерности d, получим новый базис в исходном пространстве.
- Это метод главных компонент (principal components analysis, PCA).
  - На практике обычно выполняют сингулярное разложение матрицы данных или считают собственные вектора ковариационной матрицы.

Многомерное шкалирование

- Что если попробовать как можно точнее сохранить расстояния между точками?
- ▷ Пусть $x_1, \ldots, x_N$ – исходные точки в (X, ρ).
- ▷ Заведём проекции этих точек $y_1, \ldots, y_N$ в $\mathbb{R}^{d'}$.
- ▷ Вычислим все попарные расстояния ρ_{ij} между исходными точками и δ_{ij} между проекциями, где δ – обычное евклидово расстояние.
- ▷ Минимизируем

$$\sum_i \sum_j (\rho_{ij} - \delta_{ij})^2$$

– можно градиентным спуском, двигая проекции.

- Это многомерное шкалирование (multidimensional scaling).

t-SNE

- Дальнейшее развитие этой идеи – алгоритм t-SNE (t-distributed stochastic neighbor embedding).
- ρ_{ij} между исходными точками и δ_{ij} между проекциями записываются через распределения вероятностей.
- Минимизируется расстояние между распределениями (KL-divergence).

UMAP

- Параметр n – число соседей.
- Строится ориентированный взвешенный граф:
  - вершины – точки данных
  - из каждой вершины исходит n рёбер к ближайшим соседям
  - сумма весов рёбер ограничена константой
- Веса рёбер симметризуются $\Rightarrow$ пропадает ориентация.
- Граф вкладывается в $\mathbb{R}^{d'}$, используется force-directed алгоритм: вершины притягиваются и отталкиваются, в зависимости от весов рёбер.