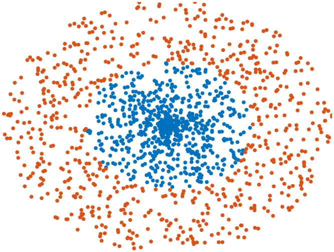
Introduction to Nonlinear Models

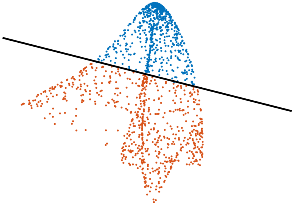
Numerical Methods for Deep Learning

Motivation: Nonlinear Models

In general, impossible to find a linear separator between classes



input features



transformed features

Goal/Trick
Embed the points in higher dimension and/or move the points to make them linearly separable

Example: Linear Fitting

Assume $\mathbf{C} \in \mathbb{R}^{n_c \times n}$, $\mathbf{Y} \in \mathbb{R}^{n_f \times n}$ and $n \gg n_f$. Goal: Find $\mathbf{W} \in \mathbb{R}^{n_c \times n_f}$ such that

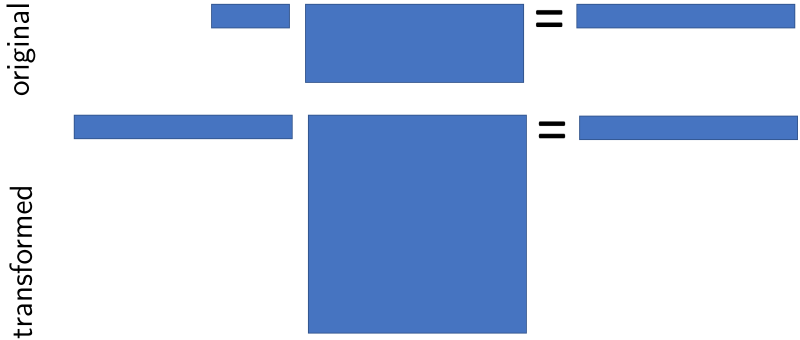
$$\mathbf{C} = \mathbf{W}\mathbf{Y}$$

If $\mathrm{rank}(\mathbf{Y}) < n$, may not be possible to fit the data.

Two options:

1. Regression: Solve $\min_{\mathbf{W}} \|\mathbf{W}\mathbf{Y} - \mathbf{C}\|_F^2 \rightsquigarrow$ always has solutions, but residual might be large
2. Nonlinear Model: Replace $\mathbf{Y}$ by $\sigma(\mathbf{K}\mathbf{Y})$, where σ is element-wise function (aka activation) and $\mathbf{K} \in \mathbb{R}^{m \times n_f}$ where $m \gg n_f$

Illustrating Nonlinear Models



Remarks

- instead of $\mathbf{W}\mathbf{Y} = \mathbf{C}$ solve $\hat{\mathbf{W}}\sigma(\mathbf{K}\mathbf{Y}) = \mathbf{C}$
- solve bigger problem $\rightsquigarrow$ memory, computation, . . .
- what happens to $\mathrm{rank}(\sigma(\mathbf{K}\mathbf{Y}))$ when $\sigma(x) = x$?

Universal Approximation Theorem

Given the data $\mathbf{Y} \in \mathbb{R}^{n_f \times n}$ and $\mathbf{C} \in \mathbb{R}^{n_c \times n}$ with $n \gg n_f$ There is nonlinear function $\sigma : \mathbb{R} \to \mathbb{R}$, a matrix $\mathbf{K} \in R^{m \times n_f}$, and a bias $b \in \mathbb{R}$ such that

$$\mathrm{rank}(\sigma(\mathbf{KY} + b)) = n.$$

Therefore, possible [**? ?**] to find $\mathbf{W} \in \mathbb{R}^{n_c \times m}$

$$\mathbf{W}\sigma(\mathbf{KY} + b)\mathbf{W} = \mathbf{C}$$

Choosing Nonlinear Model

$$\mathbf{W}\sigma(\mathbf{KY} + b) = \mathbf{C}$$

- how to choose σ?
  - early days: motivated by neurons
  - popular choice: $\sigma(x) = \tanh(x)$ (smooth, bounded, ...)
  - nowadays: $\sigma(x) = \max(x, 0)$ (aka ReLU, rectified linear unit, non-differentiable, not bounded, simple)
- how to choose **K** and b?
  - pick randomly $\rightsquigarrow$ branded as *extreme learning machines* [**?**]
  - train (optimize) $\rightsquigarrow$ deep learning (when we have multiple layers)

First Experiment: Random Transformation

Select activation function and choose $\mathbf{K}$ and b randomly and solve the least-squares/classification problem

The Pros:

- universal approximation theorem: can interpolate any function
- very easy to program
- can serve as a benchmark to more sophisticated methods

Some concerns:

- may require very large $\mathbf{K}$ (size of the data)
- may not generalize well
- large dense linear algebra

References