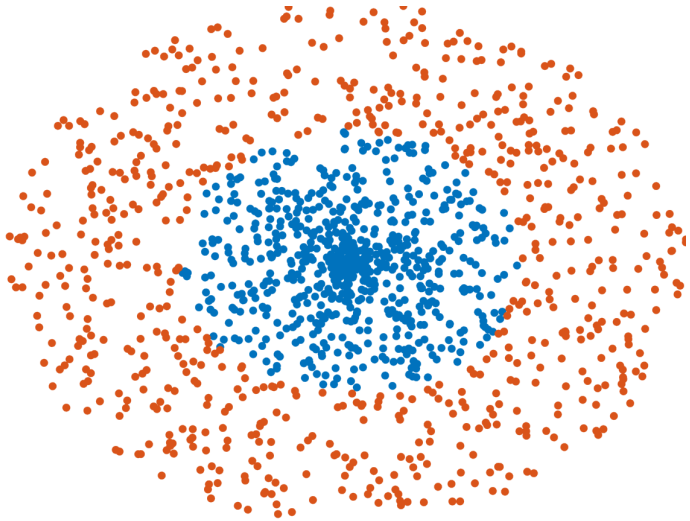
Introduction to Nonlinear Models

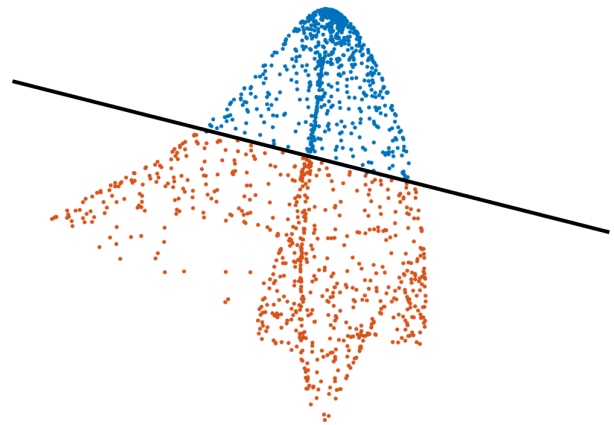
Numerical Methods for Deep Learning

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

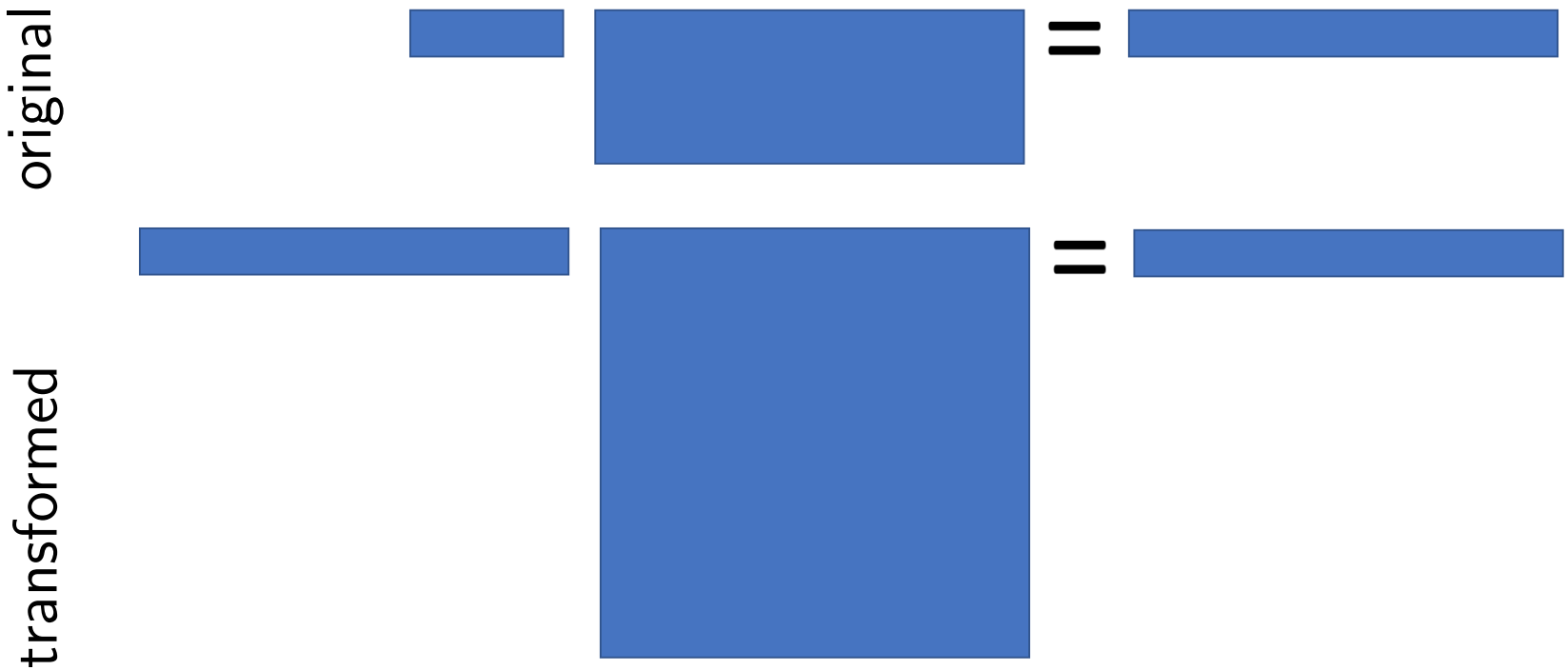
$$\mathbf{C} = \mathbf{WY}$$

If $\mathrm{rank}(\mathbf{Y}) < n$, there may be no solution.

Two options:

1. Regression: Solve $\min_{\mathbf{W}} \|\mathbf{WY} - \mathbf{C}\|_F^2 \rightsquigarrow$ always has solutions, but residual might be large
2. Nonlinear Model: Replace $\mathbf{Y}$ by $\sigma(\mathbf{KY})$ in regression, 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 $\mathbf{WY} = \mathbf{C}$ solve $\hat{\mathbf{W}}\sigma(\mathbf{KY}) = \mathbf{C}$
- solve bigger problem $\rightsquigarrow$ memory, computation, ...
- what happens to $\mathrm{rank}(\sigma(\mathbf{KY}))$ when $\sigma(x) = x$?

Conjecture: Universal Approximation Properties

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 $\mathbf{K} \in \mathbb{R}^{m \times n_f}$, and a bias $\mathbf{b} \in \mathbb{R}^m$ such that

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

Therefore, possible [1, 2] to find $\mathbf{W} \in \mathbb{R}^{n_c \times m}$

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

Choosing Nonlinear Model

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

- 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 $\mathbf{K}$ and $\mathbf{b}$?
  - pick randomly $\rightsquigarrow$ branded as *extreme learning machines* [3]
  - train (optimize) $\rightsquigarrow$ done for most neural network
  - *deep learning* when neural network has many layers

First Experiment: Random Transformation

Select activation function and choose $\mathbf{K}$ and $\mathbf{b}$ 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 $\mathbf{K}$ (scale with n, number of examples)
- may not generalize well
- large dense linear algebra

`EELM_Peaks.m`

References

[1] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[2] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, Dec. 2006.