Deep Networks Are Kernel Machines

Pedro Domingos

Paul G. Allen School of Computer Science & Engineering
University of Washington

Overview

- Claim: Deep networks discover new representations
- This talk: They're just kernel machines with a particular kernel
- True of all models learned by gradient descent
- Weights are a superposition of the training examples $\rightarrow$ Interpretability
- Architecture incorporates knowledge
- Many implications

Kernel Machines

$$y = g\left(\sum_i a_i K(x, x_i) + b\right)$$

y	Model output
$g(.)$	Optional nonlinearity
a_i	Learned parameters (typically $a'_i y^*_i$)
$K(.,.)$	Kernel (predefined or learned)
x	Query data point
x_i	Training data points
b	Learned parameter

Gradient Descent

$$w_{s+1} = w_s - \epsilon \nabla_w L(w_s)$$

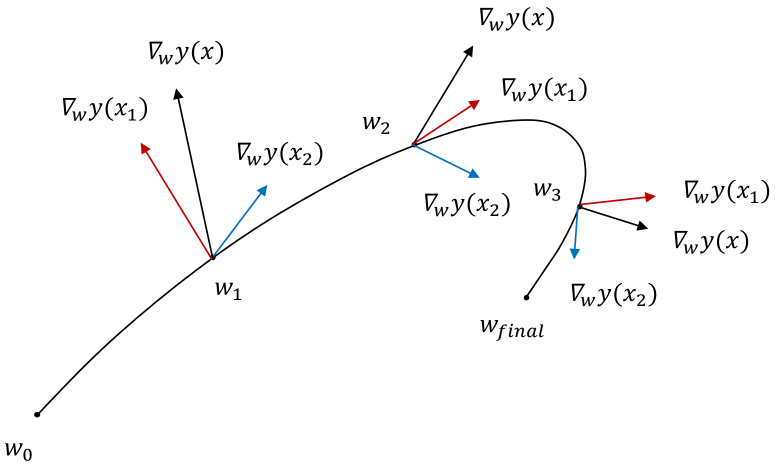
w	Weight vector
s	Step
ϵ	Learning rate
$L(.)$	Loss function

Path Kernels

$$K(x, x') = \int_{c(t)} \nabla_w y(x) \cdot \nabla_w y(x')\, dt$$

$K(.,.)$	Path kernel
x, x'	Data points
t	Time
w	Weight vector
$c(t)$	Path taken by w during gradient descent
$y(.)$	Model

Example



$\nabla_w y(x)$
$\nabla_w y(x)$
$\nabla_w y(x_1)$
$\nabla_w y(x_1)$
w_2
$\nabla_w y(x_2)$
$\nabla_w y(x_2)$
w_3
$\nabla_w y(x_1)$
$\nabla_w y(x)$
w_1
$\nabla_w y(x_2)$
w_{final}
w_0

Definitions

The *tangent kernel* associated with function $f_w(x)$ and parameter vector v is

$$K^g_{f,v}(x, x') = \nabla_w f_w(x) \cdot \nabla_w f_w(x')$$

with the gradients taken at v.

The *path kernel* associated with function $f_w(x)$ and curve $c(t)$ in parameter space is

$$K^p_{f,c}(x, x') = \int_{c(t)} K^g_{f,w(t)}(x, x')\, dt$$

Theorem

Suppose the model $y = f_w(x)$, with f a differentiable function of w, is learned from a training set $\{(x_i, y_i^*)\}_{i=1}^m$ by gradient descent with differentiable loss function $L = \sum_i L(y_i^*, y_i)$ and learning rate ϵ. Then

$$\lim_{\epsilon \to 0} y = \sum_{i=1}^{m} a_i K(x, x_i) + b$$

where $K(x, x_i)$ is the path kernel associated with $f_w(x)$ and the path taken by the parameters during gradient descent, a_i is the average $-\partial L / \partial y_i$ along the path weighted by the corresponding tangent kernel, and b is the initial model.

Proof (1)

Gradient flow:

$$\frac{dw}{dt} = -\nabla L(w)$$

Chain rule:

$$\frac{dy}{dt} = \sum_{j=1}^{d} \frac{\partial y}{\partial w_j} \frac{dw_j}{dt} = \nabla y \cdot \frac{dw}{dt}$$

Combining the two:

$$\frac{dy}{dt} = -\nabla y \cdot \nabla L$$

Proof (2)

Additivity of the loss:

$$\frac{dy}{dt} = -\sum_{i=1}^{m} \nabla y \cdot \nabla L_i$$

Chain rule:

$$\frac{dy}{dt} = -\sum_{i=1}^{m} L_i' \, \nabla y \cdot \nabla y_i$$

Integrating:

$$y = y_0 - \int_{c(t)} \sum_{i=1}^{m} L_i' \, \nabla y \cdot \nabla y_i \, dt$$

Proof (3)

Reordering and averaging L_i':

$$y = y_0 - \sum_{i=1}^{m} \overline{L_i'} \int_{c(t)} \nabla y \cdot \nabla y_i \, dt$$

Compare:

$$y = \sum_{i=1}^{m} a_i K(x, x_i) + b$$

QED

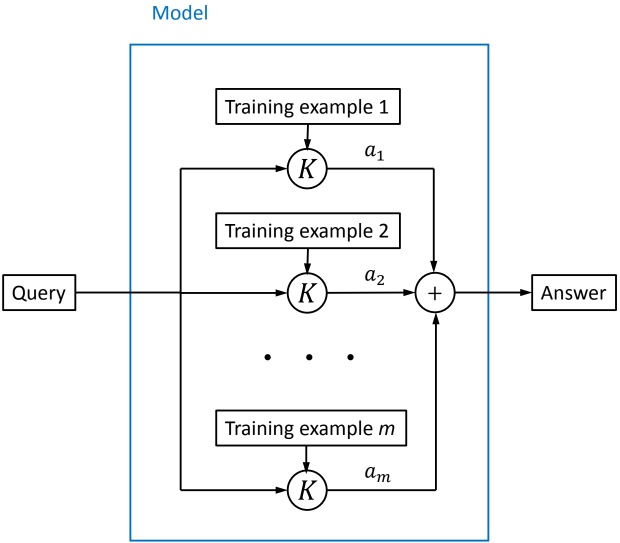
Remarks

- Coefficients depend on x, but only through kernels
- Alternate formulation: loss-weighted path kernel $\rightarrow a_i = -1$
- Applies to least squares, cross-entropy, likelihood, etc.
- Adding regularizer just adds term to b
- Readily extended to stochastic gradient
- Generalization of classic result for single-layer perceptrons
- Assumes learning rate is sufficiently small

Implications for Deep Learning

- Interpretability
- Empirical behavior
- Representation learning

Superposition



Implications for Kernel Machines

- Incorporating knowledge
- Curse of dimensionality
- Scalability

Other Implications

- Boosting
- Graphical models
- Convex learning problems

Research Directions

- Other learning algorithms
- Better gradient descent
- Representation learning
- Superposition learning