A Minimal Model of Representation Learning

Dan Roberts

MIT & Salesforce

December 14, 2021

Based on *The Principles of Deep Learning Theory* w/ Yaida and Hanin, 2106.10165, to be published by Cambridge University Press in 2022.

Machine Learning Models

$z_i(x; \theta)$

Machine Learning Models

$z_i(x; \theta)$

- x is an **input**, θ are the **parameters**

Machine Learning Models

$$z_i(x; \theta)$$

- x is an **input**, θ are the **parameters**
- $i = 1, \ldots, n_{\text{out}}$ is a **vectorial index**

Machine Learning Models

$z_i(x_\delta; \theta)$

- x is an **input**, θ are the **parameters**
- $i = 1, \ldots, n_{\text{out}}$ is a **vectorial index**
- $\delta \in \mathcal{D}$ is a **sample index**

Machine Learning Models

$z_{i;\delta}(\theta)$

- x is an **input**, θ are the **parameters**
- $i = 1, \ldots, n_{\text{out}}$ is a **vectorial index**
- $\delta \in \mathcal{D}$ is a **sample index**

Machine Learning Models

$$z_{i;\delta}(\theta) = z_{i;\delta}(\theta = 0) + \sum_{\mu=1}^{P} \theta_\mu \frac{dz_{i;\delta}}{d\theta_\mu}\Big|_{\theta=0} + \frac{1}{2} \sum_{\mu_1,\mu_2=1}^{P} \theta_{\mu_1}\theta_{\mu_2} \frac{d^2 z_{i;\delta}}{d\theta_{\mu_1} d\theta_{\mu_2}}\Big|_{\theta=0}$$

$$+ \frac{1}{3!} \sum_{\mu_1,\mu_2,\mu_3=1}^{P} \theta_{\mu_1}\theta_{\mu_2}\theta_{\mu_3} \frac{d^3 z_{i;\delta}}{d\theta_{\mu_1} d\theta_{\mu_2} d\theta_{\mu_3}}\Big|_{\theta=0} + \dots$$

- x is an **input**, θ are the **parameters**
- $i = 1, \dots, n_{\text{out}}$ is a **vectorial index**
- $\delta \in \mathcal{D}$ is a **sample index**
- generic models are **nonlinear** in θ

Machine Learning Models

$$z_{i;\delta}(\theta) = z_{i;\delta}(\theta = 0) + \sum_{\mu=1}^{P} \theta_\mu \frac{dz_{i;\delta}}{d\theta_\mu}\bigg|_{\theta=0}$$

- x is an **input**, θ are the **parameters**
- $i = 1, \ldots, n_{\text{out}}$ is a **vectorial index**
- $\delta \in \mathcal{D}$ is a **sample index**
- generic models are **nonlinear** in θ
- **linear models** are special

Why Study Nonlinear Models?

Why Study Nonlinear Models?

(i) **linear regression** goes back to Legendre and Gauss, and is a well understood type of *curve fitting*.

Why Study Nonlinear Models?

(i) **linear regression** goes back to Legendre and Gauss, and is a well understood type of *curve fitting*.

(ii) In **nonlinear models** the effective features *evolve* over the course of training:

$$\phi(x) \to \phi(x;\theta^{\star}),$$

allowing for nontrivial representation learning.

Why Study Nonlinear Models?

(i) **linear regression** goes back to Legendre and Gauss, and is a well understood type of *curve fitting*.

(ii) In **nonlinear models** the effective features *evolve* over the course of training:

$$\phi(x) \to \phi(x; \theta^\star),$$

allowing for nontrivial representation learning.

(iii) For **nonlinear models**, the solution depends on the method of training and optimization.

Why Study Nonlinear Models?

(i) **linear regression** goes back to Legendre and Gauss, and is a well understood type of *curve fitting*.

(ii) In **nonlinear models** the effective features *evolve* over the course of training:

$$\phi(x) \to \phi(x;\theta^\star)\,,$$

allowing for nontrivial representation learning.

(iii) For **nonlinear models**, the solution depends on the method of training and optimization.

Neural networks are nonlinear models with these two properties!

A Familiar Example

The simplest model is a (generalized) **linear model**:

$$z_{i;\delta}(\theta) = b_i + \sum_{j=1}^{n_f} W_{ij}\, \phi_j(x_\delta)\,.$$

A Familiar Example

The simplest model is a (generalized) **linear model**:

$$z_{i;\delta}(\theta) = b_i + \sum_{j=1}^{n_f} W_{ij}\,\phi_j(x_\delta)\,.$$

- Linear in both the parameters $\theta = \{b_i, W_{ij}\}$.

A Familiar Example

The simplest model is a (generalized) **linear model**:

$$z_{i;\delta}(\theta) = \sum_{j=0}^{n_f} W_{ij}\, \phi_j(x_\delta)\,.$$

- Linear in both the parameters $\theta = \{b_i, W_{ij}\}$.
- Here, we've subsumed the bias vector into the weight matrix by setting $\phi_0(x) \equiv 1$ and $W_{i0} \equiv b_i$.

A Familiar Example

The simplest model is a (generalized) **linear model**:

$$z_{i;\delta}(\theta) = \sum_{j=0}^{n_f} W_{ij}\, \phi_j(x_\delta)\,.$$

- Linear in both the parameters $\theta = \{b_i, W_{ij}\}$.
- Here, we've subsumed the bias vector into the weight matrix by setting $\phi_0(x) \equiv 1$ and $W_{i0} \equiv b_i$.
- Fixed basis of **feature functions** $\phi_j(x)$ lets it approximate functions that are *nonlinear* transformations of the input.

A Familiar Example

The simplest model is a (generalized) **linear model**:

$$z_i(\theta) = W_{i0}1 + W_{i1}x + W_{i2}x^2 + W_{i3}x^3$$

- Linear in both the parameters $\theta = \{b_i, W_{ij}\}$.
- Here, we've subsumed the bias vector into the weight matrix by setting $\phi_0(x) \equiv 1$ and $W_{i0} \equiv b_i$.
- Fixed basis of **feature functions** $\phi_j(x)$ lets it approximate functions that are *nonlinear* transformations of the input.
- (e.g. for a 1-dimensional function we might pick a basis $\phi_j(x) = \{1, x, x^2, x^3\}$ and fit cubic curves.)

Linear Regression

Supervised learning with a *linear model* is **linear regression**

$$\mathcal{L}_{\mathcal{A}}(\theta) = \frac{1}{2} \sum_{\tilde{\alpha} \in \mathcal{A}} \sum_{i=1}^{n_{\text{out}}} \left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij} \phi_j(x_{\tilde{\alpha}}) \right]^2 ,$$

where $y_i \equiv f_i(x)$ is an observed true output or *label*.

Linear Regression

Supervised learning with a *linear model* is **linear regression**

$$\mathcal{L}_{\mathcal{A}}(\theta) = \frac{1}{2} \sum_{\tilde{\alpha} \in \mathcal{A}} \sum_{i=1}^{n_{\text{out}}} \left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij} \phi_j(x_{\tilde{\alpha}}) \right]^2 ,$$

where $y_i \equiv f_i(x)$ is an observed true output or *label*.

- We could solve by **direct optimization**:

$$0 = \left. \frac{d\mathcal{L}_{\mathcal{A}}}{dW_{ij}} \right|_{W=W^{\star}} .$$

Linear Regression

Supervised learning with a *linear model* is **linear regression**

$$\mathcal{L}_{\mathcal{A}}(\theta) = \frac{1}{2} \sum_{\tilde{\alpha} \in \mathcal{A}} \sum_{i=1}^{n_{\text{out}}} \left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij} \phi_j(x_{\tilde{\alpha}}) \right]^2 ,$$

where $y_i \equiv f_i(x)$ is an observed true output or *label*.

- We could solve by **direct optimization**:

$$0 = \left. \frac{d\mathcal{L}_{\mathcal{A}}}{dW_{ij}} \right|_{W=W^\star} .$$

- We could solve by **gradient descent**:

$$W_{ij}(t+1) = W_{ij}(t) - \eta \left. \frac{d\mathcal{L}_{\mathcal{A}}}{dW_{ij}} \right|_{W_{ij}=W_{ij}(t)} .$$

The Kernel

Let us introduce a new $N_{\mathcal{D}} \times N_{\mathcal{D}}$-dimensional symmetric matrix:

$$k_{\delta_1\delta_2} \equiv k(x_{\delta_1}, x_{\delta_2}) \equiv \sum_{j=0}^{n_f} \phi_j(x_{\delta_1})\, \phi_j(x_{\delta_2}) \ .$$

As an inner product of features, the **kernel** $k_{\delta_1\delta_2}$ is a measure of similarity between two inputs $x_{i;\delta_1}$ and $x_{i;\delta_2}$ in *feature space*.

We'll also denote an $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$-dimensional submatrix of the kernel evaluated on the training set as $\widetilde{k}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ with a tilde. This lets us write its **inverse** as $\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}$, which satisfies

$$\sum_{\tilde{\alpha}_2 \in \mathcal{A}} \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2} \widetilde{k}_{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3} \, .$$

Linear Models and Kernel Methods

Two forms of a solution for a **linear model**:

- *parameter space – linear regression*

$$z_i(x_{\dot\beta};\theta^\star) = \sum_{j=0}^{n_f} W_{ij}^\star \phi_j(x_{\dot\beta})$$

- *sample space – kernel methods*

$$z_i(x_{\dot\beta};\theta^\star) = \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k_{\dot\beta\tilde\alpha_1}\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2}\,.$$

Linear Models and Kernel Methods

Two forms of a solution for a **linear model**:

- *parameter space – linear regression*

$$z_i(x_{\dot{\beta}};\theta^\star) = \sum_{j=0}^{n_f} W_{ij}^\star \phi_j(x_{\dot{\beta}})$$

- *sample space – kernel methods*

$$z_i(x_{\dot{\beta}};\theta^\star) = \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_1} \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2}\,.$$

Features of this model, expressed as $\phi_j(x)$ or $k_{\delta_1\delta_2}$, are *fixed*.

Nonlinear Models

To go beyond the linear paradigm, let's slightly *deform* it to get a **nonlinear model**, specifically a **quadratic model**:

$$z_{i;\delta}(\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1} W_{ij_2} \psi_{j_1 j_2}(x_\delta)$$

Nonlinear Models

To go beyond the linear paradigm, let's slightly *deform* it to get a **nonlinear model**, specifically a **quadratic model**:

$$z_{i;\delta}(\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1 j_2}(x_\delta)$$

- It's nonlinear because it's quadratic in the weights: $W_{ij_1}W_{ij_2}$.

Nonlinear Models

To go beyond the linear paradigm, let's slightly *deform* it to get a **nonlinear model**, specifically a **quadratic model**:

$$z_{i;\delta}(\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1 j_2}(x_\delta)$$

- It's nonlinear because it's quadratic in the weights: $W_{ij_1}W_{ij_2}$.
- $\epsilon \ll 1$ is small parameter that controls the size of the deformation.

Nonlinear Models

To go beyond the linear paradigm, let's slightly *deform* it to get a **nonlinear model**, specifically a **quadratic model**:

$$z_{i;\delta}(\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_\delta)$$

- It's nonlinear because it's quadratic in the weights: $W_{ij_1}W_{ij_2}$.
- $\epsilon \ll 1$ is small parameter that controls the size of the deformation.
- We've introduced $(n_f+1)(n_f+2)/2$ **meta feature functions**, $\psi_{j_1j_2}(x)$, with *two* feature indices.

Quadratic Models

To familiarize ourselves with this model, let's make a small change in the model parameters $W_{ij} \to W_{ij} + dW_{ij}$:

$$z_i(x_\delta; \theta + d\theta) = z_i(x_\delta; \theta) + \sum_{j=0}^{n_f} dW_{ij} \left[\phi_j(x_\delta) + \epsilon \sum_{j_1=0}^{n_f} W_{ij_1} \psi_{j_1 j}(x_\delta) \right] + \frac{\epsilon}{2} \sum_{j_1, j_2 = 0}^{n_f} dW_{ij_1} dW_{ij_2} \psi_{j_1 j_2}(x_\delta).$$

Quadratic Models

To familiarize ourselves with this model, let's make a small change in the model parameters $W_{ij} \to W_{ij} + dW_{ij}$:

$$z_i(x_\delta; \theta + d\theta) = z_i(x_\delta; \theta) + \sum_{j=0}^{n_f} dW_{ij} \left[\phi_j(x_\delta) + \epsilon \sum_{j_1=0}^{n_f} W_{ij_1} \psi_{j_1 j}(x_\delta) \right] + \frac{\epsilon}{2} \sum_{j_1, j_2=0}^{n_f} dW_{ij_1} dW_{ij_2} \psi_{j_1 j_2}(x_\delta).$$

Let us make a shorthand for the quantity in the square bracket,

$$\phi^{\mathrm{E}}_{ij}(x_\delta; \theta) \equiv \frac{dz_i(x_\delta; \theta)}{dW_{ij}} = \phi_j(x_\delta) + \epsilon \sum_{k=0}^{n_f} W_{ik} \psi_{kj}(x_\delta),$$

which is an **effective feature function**.

Effective Feature Learning

The quadratic model $z_i(x_\delta; \theta)$ behaves *effectively* as if it has a parameter-dependent feature function, $\phi^{\mathsf{E}}_{ij}(x_\delta; \theta)$.

Effective Feature Learning

The quadratic model $z_i(x_\delta;\theta)$ behaves *effectively* as if it has a parameter-dependent feature function, $\phi^{\mathrm{E}}_{ij}(x_\delta;\theta)$.

- The $\phi^{\mathrm{E}}_{ij}(x_\delta;\theta)$ *learns* with update dW_{ik}:

$$\phi^{\mathrm{E}}_{ij}(x_\delta;\theta+d\theta) = \phi^{\mathrm{E}}_{ij}(x_\delta;\theta) + \epsilon \sum_{k=0}^{n_f} dW_{ik}\,\psi_{kj}(x_\delta)\,.$$

Effective Feature Learning

The quadratic model $z_i(x_\delta;\theta)$ behaves *effectively* as if it has a parameter-dependent feature function, $\phi^{\mathsf{E}}_{ij}(x_\delta;\theta)$.

- The $\phi^{\mathsf{E}}_{ij}(x_\delta;\theta)$ *learns* with update dW_{ik}:

$$\phi^{\mathsf{E}}_{ij}(x_\delta;\theta+d\theta) = \phi^{\mathsf{E}}_{ij}(x_\delta;\theta) + \epsilon \sum_{k=0}^{n_f} dW_{ik}\,\psi_{kj}(x_\delta)\,.$$

- For comparison, for the linear model we'd have:

$$z_i(x_\delta;\theta+d\theta) = z_i(x_\delta;\theta) + \sum_{j=0}^{n_f} dW_{ij}\,\phi_j(x_\delta)$$

Effective Feature Learning

The quadratic model $z_i(x_\delta;\theta)$ behaves *effectively* as if it has a parameter-dependent feature function, $\phi^{\mathsf{E}}_{ij}(x_\delta;\theta)$.

- The $\phi^{\mathsf{E}}_{ij}(x_\delta;\theta)$ *learns* with update dW_{ik}:

$$\phi^{\mathsf{E}}_{ij}(x_\delta;\theta+d\theta)=\phi^{\mathsf{E}}_{ij}(x_\delta;\theta)+\epsilon\sum_{k=0}^{n_f}dW_{ik}\,\psi_{kj}(x_\delta)\,.$$

- For comparison, for the linear model we'd have:

$$z_i(x_\delta;\theta+d\theta)=z_i(x_\delta;\theta)+\sum_{j=0}^{n_f}dW_{ij}\,\phi_j(x_\delta)$$

Thus quadratic model has a *hierarchical structure*, where the features evolve as if they are described by a linear model and the model's output evolves in a more complicated nonlinear way.

Quadratic Regression

Supervised learning a quadratic model doesn't have a particular name, but if it did, we'd all probably agree that its name should be **quadratic regression**:

$$\mathcal{L}_{\mathcal{A}}(\theta) = \frac{1}{2}\sum_{\tilde{\alpha}\in\mathcal{A}}\sum_{i=1}^{n_{\text{out}}}\left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij}\,\phi_j(x_{\tilde{\alpha}}) - \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_{\tilde{\alpha}})\right]^2 .$$

Quadratic Regression

Supervised learning a quadratic model doesn't have a particular name, but if it did, we'd all probably agree that its name should be **quadratic regression**:

$$\mathcal{L}_{\mathcal{A}}(\theta) = \frac{1}{2} \sum_{\tilde{\alpha} \in \mathcal{A}} \sum_{i=1}^{n_{\text{out}}} \left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij}\, \phi_j(x_{\tilde{\alpha}}) - \frac{\epsilon}{2} \sum_{j_1, j_2 = 0}^{n_f} W_{ij_1} W_{ij_2} \psi_{j_1 j_2}(x_{\tilde{\alpha}}) \right]^2 .$$

The loss is now *quartic* in the parameters, and in general

$$0 = \left. \frac{d\mathcal{L}_{\mathcal{A}}}{dW_{ij}} \right|_{W = W^{\star}} ,$$

doesn't give analytical solutions or a tractable practical method.

Quadratic Regression

Supervised learning a quadratic model doesn't have a particular name, but if it did, we'd all probably agree that its name should be **quadratic regression**:

$$\mathcal{L}_{\mathcal{A}}(\theta) = \frac{1}{2} \sum_{\tilde{\alpha} \in \mathcal{A}} \sum_{i=1}^{n_{\text{out}}} \left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij}\, \phi_j(x_{\tilde{\alpha}}) - \frac{\epsilon}{2} \sum_{j_1, j_2 = 0}^{n_f} W_{ij_1} W_{ij_2} \psi_{j_1 j_2}(x_{\tilde{\alpha}}) \right]^2 .$$

The loss is now *quartic* in the parameters, but we can optimize with *gradient descent*:

$$W_{ij}(t+1) = W_{ij}(t) - \eta \frac{d\mathcal{L}_{\mathcal{A}}}{dW_{ij}} \bigg|_{W_{ij} = W_{ij}(t)} .$$

This will find a minimum in practice.

Quadratic Model Gradient Descent Dynamics

The weights will update as

$$
\begin{aligned}
W_{ij}(t+1) &= W_{ij}(t) - \eta \left. \frac{d\mathcal{L}_{\mathcal{A}}}{dW_{ij}} \right|_{W_{ij}=W_{ij}(t)} \\
&= W_{ij}(t) - \eta \sum_{\tilde{\alpha}} \phi^{\mathsf{E}}_{ij;\tilde{\alpha}}(t) \left(z_{i;\tilde{\alpha}}(t) - y_{i;\tilde{\alpha}} \right).
\end{aligned}
$$

While the model and effective features update as

$$
\begin{aligned}
z_{i;\delta}(t+1) =& z_{i;\delta}(t) + \sum_{j} dW_{ij}(t)\, \phi^{\mathsf{E}}_{ij;\delta}(t) \\
& + \frac{\epsilon}{2} \sum_{j_1,j_2} dW_{ij_1}(t)\, dW_{ij_2}(t)\, \psi_{j_1 j_2}(x_\delta), \\
\phi^{\mathsf{E}}_{ij;\delta}(t+1) =& \phi^{\mathsf{E}}_{ij;\delta}(t) + \epsilon \sum_{k=0}^{n_f} dW_{ik}(t)\, \psi_{kj}(x_\delta).
\end{aligned}
$$

Aside: Meta Kernel

Useful to define a **meta kernel**:

$$\mu_{\delta_0\delta_1\delta_2} \equiv \sum_{j_1,j_2=0}^{n_f} \epsilon\, \psi_{j_1 j_2}(x_{\delta_0})\, \phi_{j_1}(x_{\delta_1})\, \phi_{j_2}(x_{\delta_2}).$$

Aside: Meta Kernel

Useful to define a **meta kernel**:

$$\mu_{\delta_0\delta_1\delta_2} \equiv \sum_{j_1,j_2=0}^{n_f} \epsilon\, \psi_{j_1 j_2}(x_{\delta_0})\, \phi_{j_1}(x_{\delta_1})\, \phi_{j_2}(x_{\delta_2}).$$

- This is a *parameter-independent* tensor given entirely in terms of the fixed $\phi_j(x)$ and $\psi_{j_1 j_2}(x)$ that define the model.

Aside: Meta Kernel

Useful to define a **meta kernel**:

$$\mu_{\delta_0\delta_1\delta_2} \equiv \sum_{j_1,j_2=0}^{n_f} \epsilon\, \psi_{j_1 j_2}(x_{\delta_0})\, \phi_{j_1}(x_{\delta_1})\, \phi_{j_2}(x_{\delta_2}).$$

- This is a *parameter-independent* tensor given entirely in terms of the fixed $\phi_j(x)$ and $\psi_{j_1 j_2}(x)$ that define the model.
- For a fixed input x_{δ_0}, $\mu_{\delta_0\delta_1\delta_2}$ computes a different feature-space inner product between the two inputs, x_{δ_1} & x_{δ_2}.

Aside: Meta Kernel

Useful to define a **meta kernel**:

$$\mu_{\delta_0\delta_1\delta_2} \equiv \sum_{j_1,j_2=0}^{n_f} \epsilon\, \psi_{j_1 j_2}(x_{\delta_0})\, \phi_{j_1}(x_{\delta_1})\, \phi_{j_2}(x_{\delta_2}).$$

- This is a *parameter-independent* tensor given entirely in terms of the fixed $\phi_j(x)$ and $\psi_{j_1 j_2}(x)$ that define the model.
- For a fixed input x_{δ_0}, $\mu_{\delta_0\delta_1\delta_2}$ computes a different feature-space inner product between the two inputs, x_{δ_1} & x_{δ_2}.
- Due to the inclusion of ϵ into the definition of $\mu_{\delta_0\delta_1\delta_2}$, we should think of it as being parametrically small too.

Solution

$$
\begin{aligned}
& z_{i;\dot\beta}(\infty) \\
&= \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k_{\dot\beta\tilde\alpha_1}\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} \\
&\quad + \sum_{\tilde\alpha_1,\dots,\tilde\alpha_4\in\mathcal{A}} \left[\mu_{\tilde\alpha_1\dot\beta\tilde\alpha_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6\in\mathcal{A}} k_{\dot\beta\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_5\tilde\alpha_6}\mu_{\tilde\alpha_1\tilde\alpha_6\tilde\alpha_2}\right] Z_{\mathrm{A}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} y_{i;\tilde\alpha_3}\, y_{i;\tilde\alpha_4} \\
&\quad + \sum_{\tilde\alpha_1,\dots,\tilde\alpha_4\in\mathcal{A}} \left[\mu_{\dot\beta\tilde\alpha_1\tilde\alpha_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6\in\mathcal{A}} k_{\dot\beta\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_5\tilde\alpha_6}\mu_{\tilde\alpha_6\tilde\alpha_1\tilde\alpha_2}\right] Z_{\mathrm{B}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} y_{i;\tilde\alpha_3}\, y_{i;\tilde\alpha_4}
\end{aligned}
$$

where the **algorithm projectors** are given by

$$
Z_{\mathrm{A}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} \equiv \widetilde{k}^{\tilde\alpha_1\tilde\alpha_3}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_4} - \sum_{\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_5} X_{\mathrm{II}}^{\tilde\alpha_1\tilde\alpha_5\tilde\alpha_3\tilde\alpha_4}\,,
$$

$$
Z_{\mathrm{B}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} \equiv \widetilde{k}^{\tilde\alpha_1\tilde\alpha_3}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_4} - \sum_{\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_5} X_{\mathrm{II}}^{\tilde\alpha_1\tilde\alpha_5\tilde\alpha_3\tilde\alpha_4} + \frac{\eta}{2} X_{\mathrm{II}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4}\,.
$$

Here, an **inverting tensor** is implicitly defined:

$$\begin{aligned}
&\delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_5}\delta^{\tilde{\alpha}_2}_{\tilde{\alpha}_6} \\
&= \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\frac{1}{\eta}\left[\delta_{\tilde{\alpha}_3\tilde{\alpha}_5}\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} - (\delta_{\tilde{\alpha}_3\tilde{\alpha}_5} - \eta\widetilde{k}_{\tilde{\alpha}_3\tilde{\alpha}_5})(\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} - \eta\widetilde{k}_{\tilde{\alpha}_4\tilde{\alpha}_6})\right] \\
&= \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\left(\widetilde{k}_{\tilde{\alpha}_3\tilde{\alpha}_5}\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} + \delta_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{k}_{\tilde{\alpha}_4\tilde{\alpha}_6} - \eta\widetilde{k}_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{k}_{\tilde{\alpha}_4\tilde{\alpha}_6}\right) .
\end{aligned}$$

Solution

$$
\begin{aligned}
& z_{i;\dot\beta}(\infty) \\
= & \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k_{\dot\beta\tilde\alpha_1}\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} \\
& + \sum_{\tilde\alpha_1,\ldots,\tilde\alpha_4\in\mathcal{A}} \left[\mu_{\tilde\alpha_1\dot\beta\tilde\alpha_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6\in\mathcal{A}} k_{\dot\beta\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_5\tilde\alpha_6}\mu_{\tilde\alpha_1\tilde\alpha_6\tilde\alpha_2}\right] Z_{\mathrm{A}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} y_{i;\tilde\alpha_3}\, y_{i;\tilde\alpha_4} \\
& + \sum_{\tilde\alpha_1,\ldots,\tilde\alpha_4\in\mathcal{A}} \left[\mu_{\dot\beta\tilde\alpha_1\tilde\alpha_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6\in\mathcal{A}} k_{\dot\beta\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_5\tilde\alpha_6}\mu_{\tilde\alpha_6\tilde\alpha_1\tilde\alpha_2}\right] Z_{\mathrm{B}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} y_{i;\tilde\alpha_3}\, y_{i;\tilde\alpha_4}
\end{aligned}
$$

where the **algorithm projectors** are given by

$$
Z_{\mathrm{A}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} \equiv \widetilde{k}^{\tilde\alpha_1\tilde\alpha_3}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_4} - \sum_{\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_5} X_{\mathrm{II}}^{\tilde\alpha_1\tilde\alpha_5\tilde\alpha_3\tilde\alpha_4}\,,
$$

$$
Z_{\mathrm{B}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4} \equiv \widetilde{k}^{\tilde\alpha_1\tilde\alpha_3}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_4} - \sum_{\tilde\alpha_5}\widetilde{k}^{\tilde\alpha_2\tilde\alpha_5} X_{\mathrm{II}}^{\tilde\alpha_1\tilde\alpha_5\tilde\alpha_3\tilde\alpha_4} + \frac{\eta}{2} X_{\mathrm{II}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4}\,.
$$

Nearly-Kernel Methods

When the prediction is computed in this way, we can think of it as a *nearly-kernel machine* or **nearly-kernel methods**.

Nearly-Kernel Methods

When the prediction is computed in this way, we can think of it as a *nearly-kernel machine* or **nearly-kernel methods**.

Unlike *kernel methods*, this depends on the *learning algorithm*.

Nearly-Kernel Methods

When the prediction is computed in this way, we can think of it as a *nearly-kernel machine* or **nearly-kernel methods**.

Unlike *kernel methods*, this depends on the *learning algorithm*.

- If we'd optimized by *direct optimization*, we'd have found:

$$Z_{\mathrm{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = 0, \qquad Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = \frac{1}{2}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4}.$$

Nearly-Kernel Methods

When the prediction is computed in this way, we can think of it as a *nearly-kernel machine* or **nearly-kernel methods**.

Unlike *kernel methods*, this depends on the *learning algorithm*.

- If we'd optimized by *direct optimization*, we'd have found:

$$Z_{\mathrm{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = 0, \qquad Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = \frac{1}{2}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4}.$$

- In the ODE limit, we get different predictions

$$Z_{\mathrm{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4} - \sum_{\tilde{\alpha}_5}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_5}X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_5\tilde{\alpha}_3\tilde{\alpha}_4},$$

$$\sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\left(\widetilde{k}_{\tilde{\alpha}_3\tilde{\alpha}_5}\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} + \delta_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{k}_{\tilde{\alpha}_4\tilde{\alpha}_6}\right) = \delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_5}\delta^{\tilde{\alpha}_2}_{\tilde{\alpha}_6},$$

Representation Learning

For simplicity, let's pick the **direct optimization** solution:

$$Z_{\mathrm{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = 0, \qquad Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = \frac{1}{2}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4}.$$

Representation Learning

For simplicity, let's pick the **direct optimization** solution:

$$Z_{\mathrm{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = 0, \qquad Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = \frac{1}{2}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4}.$$

Then, we can define a ***trained kernel*** whose functional form effectively *depends on the data*:

$$k^{\sharp}_{ii;\delta_1\delta_2}(\theta^{\star}) \equiv k_{\delta_1\delta_2} + \frac{1}{2}\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} (\mu_{\delta_1\delta_2\tilde{\alpha}_1} + \mu_{\delta_2\delta_1\tilde{\alpha}_1})\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} + O\left(\epsilon^2\right) \,.$$

Representation Learning

For simplicity, let's pick the **direct optimization** solution:

$$Z_{\mathrm{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = 0, \qquad Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = \frac{1}{2}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4}.$$

Then, we can define a ***trained kernel*** whose functional form effectively *depends on the data*:

$$k^{\sharp}_{ii;\delta_1\delta_2}(\theta^\star) \equiv k_{\delta_1\delta_2} + \frac{1}{2}\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}}(\mu_{\delta_1\delta_2\tilde{\alpha}_1} + \mu_{\delta_2\delta_1\tilde{\alpha}_1})\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}y_{i;\tilde{\alpha}_2} + O\left(\epsilon^2\right) .$$

Now the nearly-kernel prediction formula can be compressed,

$$z_i(x_{\dot{\beta}};\theta^\star) = \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} k^{\sharp}_{ii;\dot{\beta}\tilde{\alpha}_1}\widetilde{k^{\sharp}_{ii}}^{\tilde{\alpha}_1\tilde{\alpha}_2}y_{i;\tilde{\alpha}_2} + O\left(\epsilon^2\right) ,$$

taking the form of a *kernel prediction*, but with the benefit of nontrivial feature evolution incorporated into the trained kernel.

Quadratic Models vs. Deep Learning

- Quadratic models are *minimal models* of feature learning:

$$z_i(x_\delta;\theta^\star) = \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k^\sharp_{ii;\delta\tilde\alpha_1}\widetilde{k^\sharp}_{ii}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} + O\left(\epsilon^2\right) ,$$

$$k^\sharp_{ii;\delta_1\delta_2}(\theta^\star) \equiv k_{\delta_1\delta_2} + \frac{1}{2}\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} \left(\mu_{\delta_1\delta_2\tilde\alpha_1} + \mu_{\delta_2\delta_1\tilde\alpha_1}\right)\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} .$$

Quadratic Models vs. Deep Learning

- Quadratic models are *minimal models* of feature learning:

$$z_i(x_\delta;\theta^\star) = \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k^\sharp_{ii;\delta\tilde\alpha_1}\widetilde{k^\sharp}_{ii}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} + O\left(\epsilon^2\right)\,,$$

$$k^\sharp_{ii;\delta_1\delta_2}(\theta^\star) \equiv k_{\delta_1\delta_2} + \frac{1}{2}\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}}\left(\mu_{\delta_1\delta_2\tilde\alpha_1}+\mu_{\delta_2\delta_1\tilde\alpha_1}\right)\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2}\,.$$

- MLPs at large-but-finite width are *cubic models*

$$z_i(x_\delta;\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{1}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_\delta)$$

$$+\frac{1}{6}\sum_{j_1,j_2,j_3=0}^{n_f} W_{ij_1}W_{ij_2}W_{ij_3}\Psi_{j_1j_2j_3}(x_\delta)$$

Quadratic Models vs. Deep Learning

- Quadratic models are *minimal models* of feature learning:

$$z_i(x_\delta;\theta^\star) = \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} k^{\sharp}_{ii;\delta\tilde{\alpha}_1} \widetilde{k^{\sharp}}_{ii}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} + O\left(\epsilon^2\right) \,,$$

$$k^{\sharp}_{ii;\delta_1\delta_2}(\theta^\star) \equiv k_{\delta_1\delta_2} + \frac{1}{2}\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} (\mu_{\delta_1\delta_2\tilde{\alpha}_1} + \mu_{\delta_2\delta_1\tilde{\alpha}_1})\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} \,.$$

- MLPs at large-but-finite width are *cubic models*

$$z_i(x_\delta;\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{1}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_\delta) + \frac{1}{6}\sum_{j_1,j_2,j_3=0}^{n_f} W_{ij_1}W_{ij_2}W_{ij_3}\Psi_{j_1j_2j_3}(x_\delta)$$

- The amount of representation learning is set by the depth-to-width ratio, $\epsilon \equiv \frac{L}{n}$, with the depth L and width n.

Quadratic Models vs. Deep Learning

- Quadratic models are *minimal models* of feature learning:

$$z_i(x_\delta;\theta^\star) = \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k^\sharp_{ii;\delta\tilde\alpha_1}\widetilde{k^\sharp}_{ii}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} + O\left(\epsilon^2\right) \,,$$

$$k^\sharp_{ii;\delta_1\delta_2}(\theta^\star) \equiv k_{\delta_1\delta_2} + \frac{1}{2}\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} (\mu_{\delta_1\delta_2\tilde\alpha_1} + \mu_{\delta_2\delta_1\tilde\alpha_1})\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2} \,.$$

- MLPs at large-but-finite width are *cubic models*

$$z_i(x_\delta;\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{1}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_\delta) + \frac{1}{6}\sum_{j_1,j_2,j_3=0}^{n_f} W_{ij_1}W_{ij_2}W_{ij_3}\Psi_{j_1j_2j_3}(x_\delta)$$

- The amount of representation learning is set by the depth-to-width ratio, $\epsilon \equiv \frac{L}{n}$, with the depth L and width n.
- The $\phi_j(x_\delta)$, $\psi_{j_1j_2}(x_\delta)$, $\Psi_{j_1j_2j_3}(x_\delta)$ are *random*.

Some Takeaways

- The deep learning framework makes it easy to define and train *nonlinear* models, letting us approximate functions that are often easy for humans to do – *is there a cat in that image?* – but hard for humans to program: a.k.a AI.
- These nonlinear models are much richer than classical statistical models such as linear regression.

Thank You!