On Regression: From Least Squares to Kernels and Neural Tangent Kernels

Advanced Machine Learning - ETH Zürich 2026

Reading guide. These notes expand the blackboard derivations. We assume undergrad-level linear algebra, probability, and vector calculus. Shapes are shown next to key equations to keep dimensions straight.

Contents

1 Problem setup and notation

- Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$; stack $\mathbf{X} = [\mathbf{x}_1^\top; \ldots; \mathbf{x}_n^\top] \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$.
- Loss: mean squared error (MSE): $L(f) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2$.
- Noise model for sections 2–3: $y_i = f^\star(\mathbf{x}_i) + \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d.

2 Act I: Linear regression is (can be) unstable

2.1 MLE equals OLS

Assume a linear nonparametric model (L-NPM): $f_{\boldsymbol{\beta}(\mathbf{x})=\mathbf{x}^\top\boldsymbol{\beta}}$ with $\boldsymbol{\beta} \in \mathbb{R}^d$. The likelihood is

$$p(\mathbf{y} \mid \boldsymbol{\beta}) \propto \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2\right).$$

Maximizing the likelihood is equivalent to minimizing MSE. The normal equations:

$$\mathbf{X}^\top\mathbf{X}\,\hat{\boldsymbol{\beta}} = \mathbf{X}^\top\mathbf{y}.$$

Proposition 1 (OLS/Pseudoinverse form)**.** *If* $\mathbf{X}$ *has full column rank,*

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}.$$

In general, $\hat{\boldsymbol{\beta}} = \mathbf{X}^\dagger\mathbf{y}$*, the minimum-norm least-squares solution.*

2.2 Bias–variance–noise decomposition (test error sketch)

State the standard decomposition for a new point $\mathbf{x}$:

$$\mathbb{E}\Big[(y - \mathbf{x}^\top\hat{\boldsymbol{\beta}})^2\Big] = \underbrace{\Big(f^\star(\mathbf{x}) - \mathbb{E}[\mathbf{x}^\top\hat{\boldsymbol{\beta}}]\Big)^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(\mathbf{x}^\top\hat{\boldsymbol{\beta}})}_{\text{variance}} + \underbrace{\sigma^2}_{\text{irreducible noise}}.$$

2.3 Instability via SVD

Let the thin SVD of $\mathbf{X}$ be $\mathbf{X} = UDV^\top$ with singular values $\{s_i\}$ on D. Then

$$\hat{\boldsymbol{\beta}} = VD^{-1}U^\top\mathbf{y} \quad \text{and} \quad \mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1} = \sigma^2 VD^{-2}V^\top = \sigma^2\sum_i \frac{1}{s_i^2}\, v_i v_i^\top.$$

Remark 1 (Why OLS can blow up)**.** *In high dimension or collinearity, some* $s_i \approx 0$*, making* $\mathrm{Var}(\hat{\boldsymbol{\beta}})$ *huge. This is the core instability motivating regularization.*

3 Act II: Stabilizing with regularization (Bayesian view of ridge)

3.1 Gaussian prior and posterior

Place a prior $\boldsymbol{\beta} \sim \mathcal{N}(0, \tau^2\mathbf{I})$. The posterior is Gaussian:

$$p(\boldsymbol{\beta} \mid \mathbf{y}) \propto \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 - \frac{1}{2\tau^2} \|\boldsymbol{\beta}\|^2\right).$$

Proposition 2 (MAP = Ridge)**.** *With* $\lambda = \sigma^2/\tau^2$*,*

$$\hat{\boldsymbol{\beta}}_{MAP} = \arg\min_{\boldsymbol{\beta}} \ \frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2 = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}.$$

3.2 Variance shrinkage in eigen-directions

Using the SVD of $\mathbf{X}$,

$$\mathrm{Var}(\mathbf{x}^\top \hat{\boldsymbol{\beta}}_{\mathrm{ridge}}) = \sigma^2 \sum_i \frac{s_i^2}{(s_i^2+\lambda)^2} (\mathbf{x}^\top v_i)^2,$$

and for the coefficient vector

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}_{\mathrm{ridge}}) = \sigma^2 \sum_i \frac{s_i^2}{(s_i^2+\lambda)^2} v_i v_i^\top.$$

Remark 2. *Small singular directions ($s_i \approx 0$) are heavily shrunk, trading variance for bias and restoring stability.*

4 Act III: Beyond linear—features and kernels

4.1 Feature maps and kernel trick

Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be (possibly infinite) features and $\Phi \in \mathbb{R}^{n\times p}$ be the design with rows $-\phi(\mathbf{x}_i)^\top$ (sign chosen to match the board algebra). Linear regression in feature space:

$$\hat{\boldsymbol{\beta}} = \Phi^\top(\Phi\Phi^\top + \lambda \mathbf{I})^{-1}\mathbf{y}, \quad \hat{f}(x_\star) = \phi(x_\star)^\top \hat{\boldsymbol{\beta}} = k(x_\star)^\top (K+\lambda \mathbf{I})^{-1}\mathbf{y},$$

where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i),\, \phi(\mathbf{x}_j)\rangle$ and $k(x_\star) = (k(x_\star, \mathbf{x}_i))_{i=1}^n$.

4.2 A polynomial (countably infinite) feature map yielding an RBF kernel

Define multi-index $\alpha \in \mathbb{N}^d$, $|\alpha| = \sum_j \alpha_j$, $\mathbf{x}^\alpha = \prod_j x_j^{\alpha_j}$, $\alpha! = \prod_j \alpha_j!$. Consider the (normalized) map

$$\phi(\mathbf{x}) = e^{-\frac{1}{2}\|\mathbf{x}\|^2} \left(\frac{\mathbf{x}^\alpha}{\sqrt{\alpha!}} \right)_{\alpha \in \mathbb{N}^d}.$$

Then

$$\left\langle \phi(\mathbf{x}),\, \phi(\mathbf{x}') \right\rangle = \exp\!\left(-\tfrac{1}{2}\|\mathbf{x}\|^2 - \tfrac{1}{2}\left\|\mathbf{x}'\right\|^2\right) \sum_\alpha \frac{(\mathbf{x}^\top \mathbf{x}')^{|\alpha|}}{\alpha!} = \exp\!\left(-\tfrac{1}{2}\left\|\mathbf{x}-\mathbf{x}'\right\|^2\right),$$

the RBF kernel.[1]

4.3 Kernelized regression

State the final predictor (with or without λ):

$$\boxed{\hat{f}(x_\star) = k(x_\star)^\top (K+\lambda \mathbf{I})^{-1}\mathbf{y}}.$$

Discuss computational trade-offs: $O(n^3)$ to invert K, independence from feature dimension.

5 Act IV: Neural networks & gradient descent as kernel regression

5.1 Model and training

One-hidden-layer network:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^m \alpha_r\, \varphi(\overset{\top}{r}\mathbf{x}), \quad \boldsymbol{\theta} = \{\alpha_r, r\}_{r=1}^m, \quad \boldsymbol{\theta}_0 \sim \mathcal{N}(0, \omega^2 \mathbf{I}).$$

[1] Use the multinomial theorem and the series for $\exp(t)$.

Gradient descent on MSE with step η:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{2n} \|\mathbf{y} - f_t\|^2, \qquad \nabla_{\boldsymbol{\theta}} \frac{1}{2n} \|\mathbf{y} - f_t\|^2 = -\frac{1}{n} \Phi_t^\top (\mathbf{y} - f_t),$$

where $f_t = (f_{\boldsymbol{\theta}_t}(\mathbf{x}_i))_{i=1}^n$ and

$$\Phi_t \in \mathbb{R}^{n \times p}, \qquad (\Phi_t)_{i,:} = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_t}(\mathbf{x}_i)^\top.$$

5.2 Linearization/Taylor step at initialization

First-order Taylor around $\boldsymbol{\theta}_0$:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) \approx f_{\boldsymbol{\theta}_0}(\mathbf{x}) + \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\mathbf{x})^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_0).$$

Stacked over the training points:

$$f \approx f_0 + \Phi_0 (\boldsymbol{\theta} - \boldsymbol{\theta}_0).$$

Solving the least-squares relation for $(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$ gives (Moore–Penrose):

$$\boldsymbol{\theta} - \boldsymbol{\theta}_0 \approx \Phi_0^\top (\Phi_0 \Phi_0^\top)^{-1} (f - f_0).$$

For a test point $x_\star$,

$$f(x_\star) \approx f_0(x_\star) + \underbrace{\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(x_\star)^\top \Phi_0^\top}_{k(x_\star)^\top} (\Phi_0 \Phi_0^\top)^{-1} (f - f_0).$$

5.3 Training dynamics and the NTK

Define the (empirical) Neural Tangent Kernel (NTK) matrix

$$K := \Phi_0 \Phi_0^\top \in \mathbb{R}^{n \times n}, \qquad K_{ij} = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\mathbf{x}_i)^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\mathbf{x}_j).$$

Under the linearized model and small η,

$$f_{t+1} = f_t - \eta K (f_t - \mathbf{y}).$$

Hence

$$f_t = \mathbf{y} + (I - \eta K)^t (f_0 - \mathbf{y}) \xrightarrow[t \to \infty]{} \mathbf{y} \quad \text{(on the training set if } \eta < 2/\lambda_{\max}(K)).$$

At a test point,

$$\boxed{\hat{f}(x_\star) = k(x_\star)^\top K^{-1} \mathbf{y}}, \qquad k(x_\star)_i = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(x_\star)^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\mathbf{x}_i).$$

Remark 3 (Wide-network limit (the "hard claims"))**.** *As width $m \to \infty$, with standard random initialization, K and $k(\cdot)$ concentrate around deterministic limits; gradient descent therefore emulates* kernel regression with the NTK.

6 Summary and takeaways

- OLS is MLE for Gaussian noise but is unstable when $\mathbf{X}$ is ill-conditioned or rank-deficient.
- Ridge (MAP with a Gaussian prior) shrinks high-variance directions and stabilizes estimation.
- Infinite feature maps + the kernel trick yield predictors $\hat{f}(x_\star) = k(x_\star)^\top (K + \lambda I)^{-1} \mathbf{y}$ without explicit features.
- In the linearized regime, gradient descent training of wide neural nets performs kernel regression with the NTK.

Appendix (optional material to expand)

A. Full derivation of the bias–variance decomposition.

B. Conjugate Gaussian identities used in Section 3.

C. Derivation of the RBF kernel from the polynomial feature map (multi-index calculus).

D. Convergence condition for the linearized GD dynamics ($\eta < 2/\lambda_{\max}(K)$).

E. Notes on computational complexity and numerics (Cholesky for $(K+\lambda I)^{-1}$, conditioning, centering).