Machine Learning Theory

The Kernel Trick

David A. Hirshberg

March 6, 2025

Emory University

A Problem

- These Fourier series representations are great in theory.
- They make theoretical analysis easy. We'll start on that in a couple weeks.
- And they make regression look like regression in linear models.
- But using them for computation is hard.
  - Those linear models are infinite-dimensional.
  - We can use finite-dimensional approximations.
  - But sometimes that means allocating a 316 GB vector.

Context

$$\hat{\mu} = \underset{\text{increasing } m}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \{ Y_i - m(X_i) \}^2$$

When we solved for our estimator in most of our 1D models,
we optimized over n parameters. Not infinitely many.

You can parameterize in terms of the values of m at the sorted X_i.

$$m \in \mathbb{R}^n \quad \text{with} \quad m_i = m(X_i)$$

Or the jumps from one to the next.

$$b \in \mathbb{R}^n \quad \text{with} \quad b_1 = m(X_1) \text{ and } b_i = m(X_i) - m(X_{i-1}) \text{ for } i > 1.$$

This works for Lipschitz Regression, Bounded TV Regression, Monotone Regression, etc.

We'd like to do something similar for Sobolev Regression, and we will.

- We won't parameterize it the same way.
- But we'll get n parameters.

Notation

We'll reparameterize our models by scaling down our eigenvectors.

$$m = \sum_k m_k \phi_k = \sum_k \underbrace{m_k \sqrt{\lambda_k}}_{\tilde{m}_k} \underbrace{\phi_k / \sqrt{\lambda_k}}_{\tilde{\phi}_k}.$$

This makes our special inner product a little easier to express.

$$\langle T\ u, v \rangle = \sum_k \lambda_k u_k v_k = \sum_k \tilde{u}_k \tilde{v}_k = \langle \tilde{u}, \tilde{v} \rangle_2 = \sum_j \tilde{u}_j \tilde{v}_j.$$

And makes our models look a little more like a typical ridge regression.

$$\mathcal{M} = \{m : \|m\|_T \leq 1\} = \left\{ \sum_k \tilde{m}_k \tilde{\phi}_k : \|\tilde{m}\|_2 \leq 1 \right\}.$$

With this rescaling, our coefficients $\tilde{m}$ are in a sphere, not an ellipse.

Least Squares in Sobolev Spaces

$$\hat{\mu}(x) = \sum_k \hat{m}_k \tilde{\phi}_k(x) \quad \text{for} \quad \hat{m} = \operatorname*{argmin}_{\tilde{m}:\|\tilde{m}\|_2 \leq 1} \frac{1}{n} \sum_{i=1}^{n} \Big\{ \; Y_i - \Big\langle \tilde{m}, \; \tilde{\phi}(X_i) \Big\rangle_2 \; \Big\}^2$$

Is this really an optimization over infinitely many parameters?

Least Squares in Sobolev Spaces

$$\hat{\mu}(x) = \sum_k \hat{m}_k \tilde{\phi}_k(x) \quad \text{for} \quad \hat{m} = \underset{\tilde{m}:\|\tilde{m}\|_2 \leq 1}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left\{ Y_i - \left\langle \tilde{m},\ \tilde{\phi}(X_i) \right\rangle_2 \right\}^2$$

Is this really an optimization over infinitely many parameters?
No. It's an optimization over n parameters.

- Mean squared error only depends on n inner products with our basis vectors,

$$\langle \tilde{m}, \tilde{\phi}(X_1) \rangle_2 \ldots \langle \tilde{m}, \tilde{\phi}(X_n) \rangle_2.$$

- So there's a solution $\hat{m}$ that's spanned by our basis vectors.

$$\hat{m} = \sum_{j=1}^{n} \hat{\alpha}_j \tilde{\phi}(X_j).$$

- If we varied $\tilde{m}$ in a perpendicular direction:
  - We'll make the norm we're constraining bigger.
  - We won't improve squared error.

This lets us reparameterize our problem in terms of these n inner products.
There's a solution that can be written like this.

$$m(x) = \sum_k \underbrace{\sum_{j=1}^{n} \alpha_j \tilde{\phi}_k(X_j)}_{\tilde{m}_k} \tilde{\phi}_k(x)$$

The reparameterized problem

To estimate $\mu(x)$, we take a weighted average of inner products between

- The basis vectors at the observed X_i
- The basis vector $\phi(x)$ at the point we're trying to predict.

All we've got to do is learn the n weights via least squares. That is, we predict

$$\hat{\mu}(x) = \sum_{j=1}^{n} \hat{\alpha}_j \left\langle \tilde{\phi}(X_j),\ \tilde{\phi}(x) \right\rangle$$

where

$$\hat{\alpha} = \underset{\substack{\alpha \in \mathbb{R}^n \\ \left\| \sum_{j=1}^n \alpha_j \tilde{\phi}(X_j) \right\|_2 \leq 1 \\ \text{i.e. } \sum_j \tilde{m}_j^2 \leq 1}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left\{ Y_i - \sum_{j=1}^{n} \alpha_j \left\langle \tilde{\phi}(X_j),\ \tilde{\phi}(X_i) \right\rangle_2 \right\}^2.$$

- We've still got to compute inner products between infinite dimensional vectors.
- But that's all we have to do with them. The vectors only appear in inner products.
- Can we somehow do this without actually computing the vectors themselves?

The Kernel Trick

Suppose we knew how to evaluate these inner products—we've coded up a function.

$$K(x, x') = \left\langle \tilde{\phi}(x),\ \tilde{\phi}(x') \right\rangle_2 \quad \text{is called a } \textit{Kernel}.$$

Then we are set. All we've got to compute is a ridge regression with n parameters.

$$\hat{\mu}(x) = \sum_{j=1}^{n} \hat{\alpha}_j\ K(X_j, x)$$

where

$$\hat{\alpha} = \underset{\substack{\alpha \in \mathbb{R}^n \\ \sum_{ij} \alpha_i \alpha_j K(X_j, X_k) \leq 1}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\{ \ Y_i - \sum_{j=1}^{n} \alpha_j K(X_j, X_i) \ \right\}^2$$

$$= \underset{\substack{\alpha \in \mathbb{R}^n \\ \alpha^T K \alpha \leq 1}}{\operatorname{argmin}} \| Y - K\alpha \|_2^2 \quad \text{where} \quad K_{ij} = K(X_i, X_j).$$

The Kernel Trick in General

We've done nothing specific to least squares here. Here's what we've used.

- We've minimizing a cost that depends on only two aspects of the curve m
  1. Its values $m(X_1) \dots m(X_n)$ at the observed covariates
  2. Its Sobolev norm $\|m\|_T = \langle Tm, m \rangle$
- And that it prefers that norm to be small.

That means it's enough to optimize over coefficients $\tilde{m}_k$ spanned by the observed basis vectors $\tilde{\phi}(X_i)$.
Otherwise, we'd increase cost for no reason.

Conveniently

- The two aspects of m involved in our cost can be expressed in terms of inner products between these basis vectors.
- And therefore in terms of the kernel.

$$m_\alpha(x) = \sum_{j=1}^{n} \alpha_j \underbrace{\left\langle \tilde{\phi}(X_j),\ \tilde{\phi}(x) \right\rangle_2}_{K(X_j, x)} \quad \text{and} \quad \|m_\alpha\|_T = \sum_{ij} \alpha_i \alpha_j \underbrace{\left\langle \tilde{\phi}(X_j),\ \tilde{\phi}(X_j) \right\rangle_2}_{K(X_i, X_j)}$$

Other Applications of the Kernel Trick

Support Vector Machine Classification

$$\hat{\mu} = \underset{\|m\|_T \leq 1}{\operatorname{argmin}} \quad -\sum_{i=1}^{n} \max\{0,\, Y_i m(X_i)\} \text{ for labels } \; Y_i \in \{\pm 1\}.$$

It minimizes *hinge loss*, an approximation to classification error.
We classify based on the sign of $\hat{\mu}(x)$.

Where this leaves us

If we're using a Sobolev model, least squares is easy if we know the model's kernel.

$$K(x, x') = \left\langle \tilde{\phi}(x),\ \tilde{\phi}(x') \right\rangle_2 = \sum_k \lambda_k^{-1}\ \phi_k(x)\ \phi_k(x').$$

Same goes for SVM Classification. Or more or less anything else.

One option is to just make up a kernel and use that.

- The downside is that you're using whatever model it happens to be the kernel for.
- You may not know what this model is, and if you did, you might not like it.

People who take this approach often use the *Gaussian Kernel*.

$$K(x, x') = \exp\left(-\|x - x'\|_2^2\right)$$

This is the kernel for a model that includes only very smooth functions.
The result is very fast convergence to a very smooth approximation to what they want.

Where this leaves us

If we're using a Sobolev model, least squares is easy if we know the model's kernel.

$$K(x, x') = \left\langle \tilde{\phi}(x),\ \tilde{\phi}(x') \right\rangle_2 = \sum_k \lambda_k^{-1}\ \phi_k(x)\ \phi_k(x').$$

Same goes for SVM Classification. Or more or less anything else.

Another option is to choose a model and compute its kernel exactly.

- There are techniques for this, but often you wind up with another hard problem.
- To evaluate the kernel $K(x, x')$, you'd maybe have to solve a differential equation.
- Sometimes this is fast, sometimes it's not.

Where this leaves us

If we're using a Sobolev model, least squares is easy if we know the model's kernel.

$$K(x, x') = \left\langle \tilde{\phi}(x),\ \tilde{\phi}(x') \right\rangle_2 = \sum_k \lambda_k^{-1}\ \phi_k(x)\ \phi_k(x').$$

Same goes for SVM Classification. Or more or less anything else.

A third option is to choose a model, then compute its kernel approximately.

- When you do this, you're effectively using a different model.
- You're using the model that your approximation is actually the kernel for.
- But if your approximation is good, that'll be pretty similar to the model you chose.

If you're using a popular model, usually someone's done this already.

- This means you can look up the kernel — just google your model.
- But be a little careful.
- Often people aren't all that clear about what model it's really the kernel for.

Computing Kernels

The Sobolev Model in 1D

$$\mathcal{M} = \left\{ m : \langle m',\ m' \rangle_{L_2} \leq 1 \right\} = \left\{ m : \langle -m'',\ m \rangle_{L_2} \leq 1 \right\}$$

$$K(x, x') = \sum_{k=0}^{\infty} \underbrace{(\pi k)^{-2}}_{\lambda_k^{-1}} \ \underbrace{\sqrt{2}\cos(\pi k x)}_{\phi_k(x)} \ \underbrace{\sqrt{2}\cos\left(\pi k x'\right)}_{\phi_k(x')}$$

Ideas

The Sobolev Model in 1D

$$\mathcal{M} = \left\{ m : \langle m',\ m' \rangle_{L_2} \leq 1 \right\} = \left\{ m : \langle -m'',\ m \rangle_{L_2} \leq 1 \right\}$$

$$K(x, x') = \sum_{k=0}^{\infty} \underbrace{(\pi k)^{-2}}_{\lambda_k^{-1}} \ \underbrace{\sqrt{2}\cos(\pi k x)}_{\phi_k(x)} \ \underbrace{\sqrt{2}\cos(\pi k x')}_{\phi_k(x')}$$

Ideas

- Cosine product formula.

$$\cos(a)\cos(b) = \{\cos(a+b) + \cos(a-b)\}/2$$

- Integral approximation

$$\sum_{k=0}^{\infty} f(k) \approx \int_0^{\infty} f(k)$$

- Don't divide by zero—use a slightly different model.

$$\lambda_k = \epsilon^2 + (\pi k)^2 \quad \text{is the } k\text{th eigenvalue of} \quad T = \epsilon^2 - \frac{d^2}{dx^2}$$

- Let a computer integrate for us. I use wolfram alpha most of the time.

$$\text{integrate } \cos(\pi k z)/(\epsilon^2 + \pi^2 k^2) \text{ for } k \text{ from } 0 \text{ to } \infty \implies \epsilon e^{-|\epsilon z|} \ / \ 2\epsilon^2.$$

Worksheet

$$K(x, x') = \sum_{k=0}^{\infty} \underbrace{(\epsilon^2 + \pi^2 k^2)^{-1}}_{\lambda_k^{-1}} \; \underbrace{\sqrt{2}\cos(\pi k x)}_{\phi_k(x)} \; \underbrace{\sqrt{2}\cos(\pi k x')}_{\phi_k(x')}$$

Formulas

$$\cos(a)\cos(b) = \{\cos(a+b) + \cos(a-b)\}/2$$

$$\int_0^{\infty} (\epsilon^2 + \pi^2 k^2)^{-1} \cos(\pi k z) = \epsilon e^{-|\epsilon z|} \; / \; 2\epsilon^2.$$

What now?

- Implement it.
- Compare to the approximation-based approximation error we've been using to check our approximation.
- Generalize to other models we might want to use.
  - e.g. the Gaussian Sobolev Model.
  - e.g. the Isotropic Multivariate Sobolev Model.