Stat 205: Introduction to Nonparametric Statistics

Lecture 14 : Single Hidden-Layer Neural Nets: Random Initialization and Dynamics

Instructor David Donoho; TA: Yu Wang

The images in this lecture are scraped from Google Images. Many similar images are available. The intent is merely to make the lecture more vivid by providing 'eye candy'. No attempt is made to identify all sources.

Some Background Reading

- On the Approximation Capabilities of ReLU Neural Networks and Random ReLU Features
  Sun, Gilbert, Tewari (2018)
- Gradient Dynamics of Shallow Univariate ReLU Networks
  Williams et al. (2019)

Lecture 13 Topics

- History
- Terminology
- In 1-d, Single Hidden Layer NN w/Relu:
  Produces Piecewise Linear continuous spline
- Piecewise Linear splines can approximate *all* smooth functions:
  'Just make knots equispaced, increase density, use second derivative weights'.
- Some penalties (L1) use Relu's 'at data points'
  'Sparse Neuron weights'
- Some penalties (L2, RKHS) use ReLu's everywhere
  'Dense Neuron firing'
- Actual paradigm (weight decay):
  suprisingly, is L_1, i.e. sparse neuron

Lecture 14 Topics

- 2-layer neural nets in dimension 1:
  special type of linear spline.
- In principle *linear splines with adjustable knots*.
- Can fit any 'nice' function with *fixed, equispaced knots*.
  But might need many knots.
- Standard NN practice:
  random initialization + dynamic knots
  - Random initialization: Random Features Model.
  - Dynamics: move knots where most help.
- Adaptive knot positioning: smaller MSE
  heuristically: better generalization

Modern Neural Nets Terminology, 1

- Inputs
  $x \in \mathbf{R}^d$ eg an image.
- Layers
  - Intermediate results: $x \mapsto h^1 \mapsto h^2 \mapsto \ldots \mapsto h^L$
  - Activations:

$$h^1 \in \mathbf{R}^{d_1}, \ldots, h^\ell \in \mathbf{R}^{d_\ell}, \ldots, h^L \in \mathbf{R}^{d_L}.$$

  - $\ell = 1$: first layer; $\ell = L$: last layer.
- Outputs
  Regression $f_h(x) = \sum_j w_j^L h_j^L$;
  Classification $f_h(x) = \mathrm{argmax}_{c=1}^C h_j^L$.

Modern Neural Nets Terminology, 2

- Weights $W^\ell = (W^\ell_{i,j})$ where each W^ℓ is $d_{\ell-1} \times d_\ell$.
- Nonlinearity

$$\begin{aligned} \text{relu}(x) &= (x)_+ \\ &= \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \end{aligned}$$

- Preactivations

$$z^\ell = h^{\ell-1} W^\ell; \text{ meaning } z^\ell_j = \sum_i h^{\ell-1}_i W^\ell_{i,j}$$

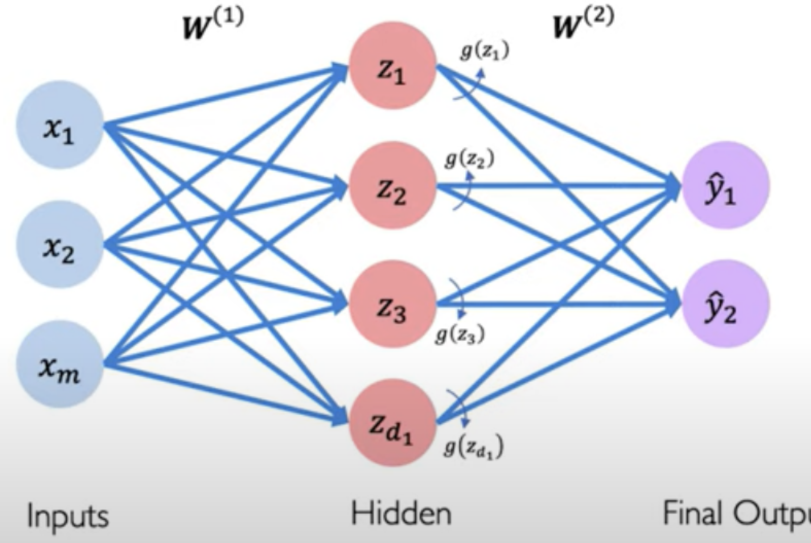
- Activations

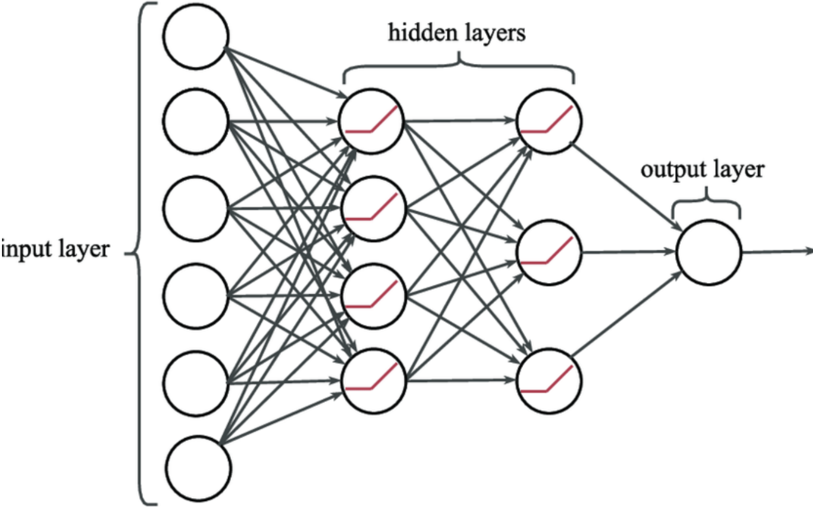
$$h^\ell = \text{relu}(z^\ell - b^\ell); \text{ meaning } h^\ell_j = \text{relu}([\sum_i h^{\ell-1}_i W^\ell_{i,j}] - b^\ell_j).$$

- Biases:

$$\text{relu}(x - b) = (x - b)_+$$

b is the location of a knot or 'kink' in the relu:


$W^{(1)}$
$W^{(2)}$
x_1
x_2
x_m
z_1
z_2
z_3
z_{d_1}
$g(z_1)$
$g(z_2)$
$g(z_3)$
$g(z_{d_1})$
$\hat{y}_1$
$\hat{y}_2$
Inputs
Hidden


hidden layers
output layer
input layer

Single-Hidden Layer, Dimension 1

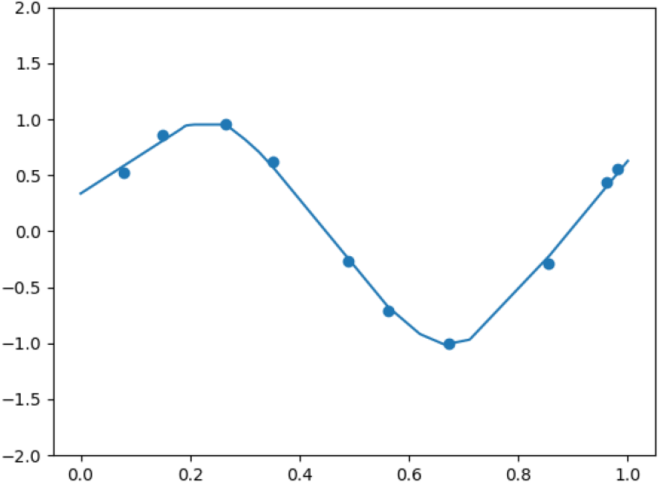
- Single-Hidden Layer $L = 2$, arbitrary dimension

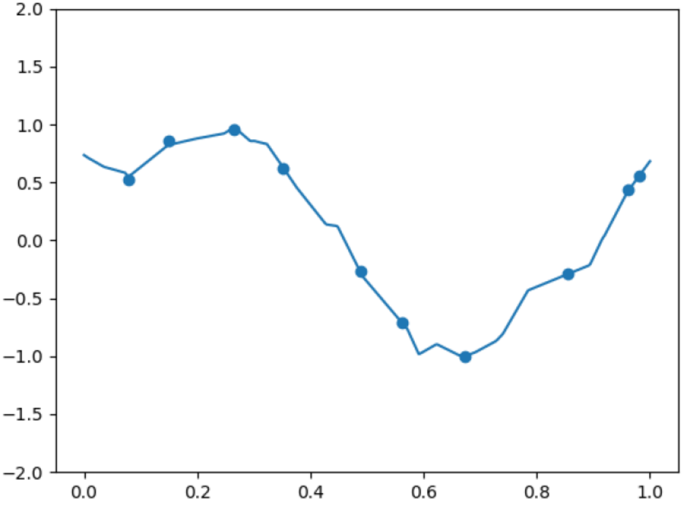
$$f(x) = \sum_j w_j^2 \mathrm{relu}([x.W_{\cdot,j}^1] - b_j)$$

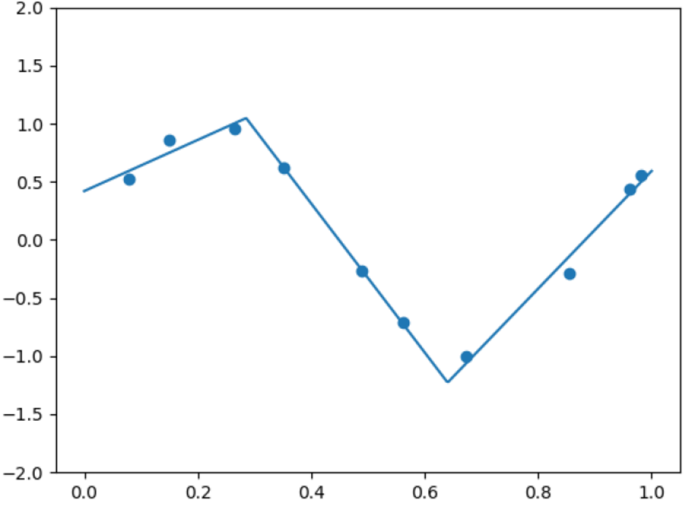
- Simplified notation for dimension 1: i.e. $x \in \mathbb{R}^1$.

$$f(x) = \sum_j c_j \cdot \mathrm{relu}(x - b_j)$$

- **Observation:**
  In the $L = 2$, $d = 1$, regression setting, $f(x)$ is a piecewise linear function on $\mathbf{R}$, with knots at the $(b_j)_{j=1}^{d_1}$.


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0

Example: *equispaced* knots

- Example: $x \in \mathbf{R}^1$, $L = 2$

$$x \in [0,1], \qquad b_j^1 = j/d_1, \qquad j = 1, \ldots d_1.$$

Single-hidden layer $L = 2$ simplifies to:

$$f(x) = \sum_j c_j \cdot \mathrm{relu}(x - j/d_1)$$

Namely: linear spline with equispaced knots.

Example: *data-point* knots

- Example: $x \in \mathbf{R}^1$, $L = 2$

$$x \in [0, 1], \qquad b_j^1 = x_j, \qquad j = 1, \dots d_1 = n.$$

Single-hidden layer $L = 2$ simplifies to:

$$f(x) = \sum_j c_j \cdot \mathrm{relu}(x - x_j)$$

Namely: linear spline with knots at data points.

Possible knot distributions

- Equispaced (slide $\cdot - 2$)
- Data-points (slide $\cdot - 1$)
- Random, independent of data (coming)
- Variable, data-aware (coming)

The last two regimes are well understood/ often used in machine learning!

Example: *random* knots

- Example: $x \in \mathbf{R}^1,\ L = 2$

  $$x \in [0, 1], \qquad b_j^1 \sim_{iid} Unif[0, 1], \qquad j = 1, \ldots d_1 = n.$$

  Single-hidden layer $L = 2$ simplifies to:

  $$f(x) = \sum_j c_j \cdot \mathrm{relu}(x - b_j^1)$$

  Namely: linear spline with knots at *random design* points.
  Random *design*: extensive statistical precedent
  Random *initialization*: standard deep net training practice.

Random Features Regression, 1

- $\phi_b(x) = relu(x - b)$
- μ is the *random features measure*.
  Example: μ is uniform distribution on $(0, 1)$.
- Random Feature Parameters: Draw

$$b_j, \qquad j = 1, \ldots, J, \qquad b_j \sim \mu$$

- Random Feature Map.

$$\phi(x) = (\phi_{b_j}(x) : j = 1, \ldots, D)^T.$$

- Random Feature Regression

$$\tilde{\theta}_{\lambda,G} = (\Phi^T \Phi + \lambda G)^{\dagger} \Phi^T Y$$

$$\tilde{f}_{\lambda,G}(x) = \Phi(x) \tilde{\theta}_{\lambda,G}$$

See for example, prize-winning paper of Rahimi and Recht 2007
Here G should be easy-to-compute and (task,f)-appropriate regularizer, eg $G = I$.

Motivating Random Features Regression

- Expect to get similar results:

$$\mathcal{K}(x,z) = \int \phi_b(x)\phi_b(z)\mu(db) = E\left[Ave_{j=1}^{D}\phi_{b_j}(x)^T\phi_{b_j}(z)\right].$$

  Implementing the one on the left might involve matrices of size $n \times n$, while the one on the right involves of size $D \times D$.

- Faster computations
  - 'Traditional' kernel method calculations: $n \times n$ matrices.
  - In RF formulation, reformulate as $D \times D$.
  - Choose $D < n$: get reduced computational complexity.
- Generic Initialization
  - Can initialize an algorithm (eg Deep Learning training)
  - Random initialization 'unbiased' , algo. doesn't get 'stuck'

Random Features Regression, 2

- Kernel RKHS
$$\mathcal{L}_\lambda(f) = n^{-1}\|Y - (f(x_i))\|_2^2 + \lambda\|f\|_{\mathcal{H}}^2$$
for some $\mathcal{H}$ TBD
- Optimizer
$$\hat{f}^* = \text{argmin}_f \mathcal{L}_\lambda(f)$$
- RF Hilbert space $f \in \mathcal{H}_{RF}$ iff
$$f(x) = \int \phi_b(x) g(b) d\mu(db),$$
where $\|g\|_{L^2(d\mu)} < \infty$.
$$\|f\|_{\mathcal{H}} = \inf\{\|g\|_{L^2(d\mu)} : f = \int \phi_b \cdot g(b)\mu(db)\}$$
- Random Feature Ridge Regression
$$\tilde{\theta}_{\lambda,I} = (\Phi^T\Phi + \lambda I)^\dagger \Phi^T Y$$
This is a $D \times D$ matrix.
$$\tilde{f}_{\lambda,I}(x) = \Phi(x)\tilde{\theta}_{\lambda,I}$$
We anticipate that: $\tilde{f}_{\lambda,I}(x) \approx \hat{f}^*$

Random Features for Large-Scale Kernel Machines

Ali Rahimi and Ben Recht

Abstract

To accelerate the training of kernel machines, we propose to map the input data to a randomized low-dimensional feature space and then apply existing fast linear methods. Our randomized features are designed so that the inner products of the transformed data are approximately equal to those in the feature space of a user specified shift-invariant kernel. We explore two sets of random features, provide convergence bounds on their ability to approximate various radial basis kernels, and show that in large-scale classification and regression tasks linear machine learning algorithms that use these features outperform state-of-the-art large-scale kernel machines.

Notation in Rahimi and Recht

Table: default

Object	Our Notation	Their Notation
Kernel	$\mathcal{K}(x, z)$	$k(x, y)$
Feature Map	$\Phi(x)$	$\mathbf{z}(x)$

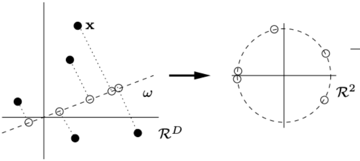
Algorithm 1 Random Fourier Features.

Require: A positive definite shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$.
Ensure: A randomized feature map $\mathbf{z}(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^D$ so that $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y})$.

Compute the Fourier transform p of the kernel k: $p(\omega) = \frac{1}{2\pi} \int e^{-j\omega'\delta} k(\delta)\, d\Delta$.

Draw D iid samples $\omega_1, \cdots, \omega_D \in \mathcal{R}^d$ from p and D iid samples $b_1, \ldots, b_D \in \mathcal{R}$ from the uniform distribution on $[0, 2\pi]$.

Let $\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{2}{D}} \left[\cos(\omega_1'\mathbf{x}+b_1) \ \cdots \ \cos(\omega_D'\mathbf{x}+b_D) \right]'$.



Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\|\Delta\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\|\omega\|_2^2}{2}}$
Laplacian	$e^{-\|\Delta\|_1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\|\Delta\|_1}$

Figure 1: Random Fourier Features. Each component of the feature map $\mathbf{z}(\mathbf{x})$ projects $\mathbf{x}$ onto a random direction ω drawn from the Fourier transform $p(\omega)$ of $k(\Delta)$, and wraps this line onto the unit circle in $\mathcal{R}^2$. After transforming two points $\mathbf{x}$ and $\mathbf{y}$ in this way, their inner product is an unbiased estimator of $k(\mathbf{x}, \mathbf{y})$. The mapping $z(\mathbf{x}) = \cos(\omega'\mathbf{x} + b)$ additionally rotates this circle by a random amount b and projects the points onto the interval $[0, 1]$. The table lists some popular shift-invariant kernels and their Fourier transforms. To deal with non-isotropic kernels, we can first whiten the data and apply one of these kernels

Algorithm 2 Random Binning Features.

Require: A point $\mathbf{x} \in \mathcal{R}^d$. A kernel function $k(\mathbf{x}, \mathbf{y}) = \prod_{m=1}^{d} k_m(|x^m - y^m|)$, so that $p_m(\Delta) \equiv \Delta \ddot{k}_m(\Delta)$ is a probability distribution on $\Delta \geq 0$.

Ensure: A randomized feature map $\mathbf{z}(\mathbf{x})$ so that $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y})$.

for $p = 1 \dots P$ **do**

Draw grid parameters $\delta, \mathbf{u} \in \mathcal{R}^d$ with the pitch $\delta^m \sim p_m$, and shift u^m from the uniform distribution on $[0, \delta^m]$.

Let z return the coordinate of the bin containing $\mathbf{x}$ as a binary indicator vector $z_p(\mathbf{x}) \equiv \text{hash}(\lceil \frac{x^1 - u^1}{\delta^1} \rceil, \cdots, \lceil \frac{x^d - u^d}{\delta^d} \rceil)$.

end for

$\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{1}{P}} \left[z_1(\mathbf{x}) \cdots z_P(\mathbf{x}) \right]'$.

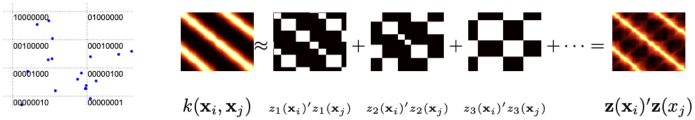


Figure 2: Random Binning Features. (left) The algorithm repeatedly partitions the input space using a randomly shifted grid at a randomly chosen resolution and assigns to each point $\mathbf{x}$ the bit string $z(\mathbf{x})$ associated with the bin to which it is assigned. (right) The binary adjacency matrix that describes this partitioning has $z(\mathbf{x}_i)'z(\mathbf{x}_j)$ in its ijth entry and is an unbiased estimate of kernel matrix.

RF Model

- Prestigious and mathematically well-specified model
- Initialization of modern deep learning training has RF interpretation
- Later stages will evolve away from *random features* towards *adaptive features*.

Gradient Dynamics of Shallow Univariate ReLU Networks

Francis Williams* **Matthew Trager***

Claudio Silva **Daniele Panozzo** **Denis Zorin** **Joan Bruna**

New York University

Abstract

We present a theoretical and empirical study of the gradient dynamics of overparameterized shallow ReLU networks with one-dimensional input, solving least-squares interpolation. We show that the gradient dynamics of such networks are determined by the gradient flow in a non-redundant parameterization of the network function. We examine the principal qualitative features of this gradient flow. In particular, we determine conditions for two learning regimes: *kernel* and *adaptive*, which depend both on the relative magnitude of initialization of weights in different layers and the asymptotic behavior of initialization coefficients in the limit of large network widths. We show that learning in the kernel regime yields smooth interpolants, minimizing curvature, and reduces to *cubic splines* for uniform initializations. Learning in the adaptive regime favors instead *linear splines*, where knots cluster adaptively at the sample points.

Notation in Williams et al.

- Our notation: single-Hidden Layer $L = 2$, $d = 1$

$$f(x) = \sum_j w_j^2 \mathrm{relu}([w_j^1 x.] - b_j)$$

- Their initial parametrization: $z_j = (a_j, b_j, c_j)$; $\mathbf{z} = (z_j)$;

$$f_{\mathbf{z}}(x) = \sum_j c_j \cdot \mathrm{relu}(a_j x + b_j)$$

- Their Canonical Reparametrization: $w_j = (r_j, \theta_j)$; $\mathbf{w} = (w_j)$;

$$\tilde{f}_{\mathbf{w}}(x) = \sum_j r_j \cdot \langle \tilde{x}, d(\theta_j) \rangle_+,$$

  - Extended variable $\tilde{x} = (1, x)$
  - $d(\theta) = (sin(\theta), cos(\theta))$
- Their Loss

$$\tilde{L}(w) = \|(y_i) - \tilde{f}_{\mathbf{w}}(x_i)\|^2_{2,n}.$$

Training in Williams et al.

- In 'standard' neural nets training:
  - $\mathbf{w}(0)$ *random* initial parametrization
  - Steps are discrete $t = 0, \ldots, T$ eg $T = 300$ epochs
  - Each step of training is proportional to gradient of $\tilde{L}$
  - Step length determined by learning rate.

$$w(t) = w(t-1) - \eta \cdot \nabla \tilde{L}(\mathbf{w}(t))$$

- In Williams et al.,
  - $\mathbf{w}(0)$ some initial parametrization;
  - $t \geq 0$ is a continuous variable;
  - $t \mapsto \mathbf{w}(t)$ follows gradient flow trajectory during training:

$$\mathbf{w}'(t) = -\nabla \tilde{L}(\mathbf{w}(t))$$

Dynamics in Williams et al.

- $f_{\mathbf{w}}$ determined by collection of relus $(r_j \langle \tilde{x}, d(\theta_j) \rangle_+)_{j=1}^m$
- Collection viewed as set of points (r_j, θ_j) in $\mathbf{R}^2$
- Pointset viewed as a probability distribution, $\mu^{(t)}$ say, at each epoch t of training.
- During training, $\mu^{(t)}$ evolves.
- How does it evolve over time?

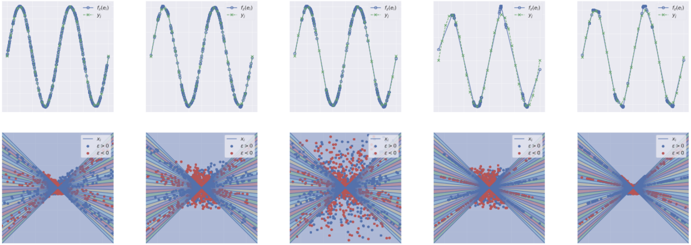
Williams et al conclusions:

- If $c_j^2 \ll a_j^2 + b_j^2$:
  inner details of j-th *relu* term effectively *frozen* during training.
- If $c_j^2 \gg a_j^2 + b_j^2$:
  inner details of j-th *relu* term effectively *dynamic* during training.
- Static features are random features
- Dynamic features move where ‘needed’.
- Needed where f differs most from linear; ie. max curvature.

Table: default

$\delta = c^2 - (a^2 + b^2)$	Configuration	Description
$\delta \gg 0$	dense	high curvature
$\delta \ll 0$	sparse	straight regions

Figure 1: *Left:* A network function $f_{\mathbf{z}}(x)$ interpolating input samples (blue x's). The knots of $f_{\mathbf{z}}(x)$ as a piecewise linear function are plotted as green circles. *Right:* The canonical parameters of the network visualized as in (6). Each particle represents a neuron and the color indicates the sign of ϵ_i. The samples x_j correspond to lines $ux_j + v = 0$. The colored regions which correspond to different activation patterns of neurons on the training data.



$\delta = -\infty$ $\delta = -1$ $\delta = 0$ $\delta = 1$ $\delta = \infty$

Figure 4: Comparison of fitting the network function to a sinusoid as δ varies (10000 epochs).

Interpretation of previous plot

- $\delta = \infty$ means $\delta = c^2 - (a^2 + b^2) \gg 0$
  - Such points concentrate along stable *stripes*
  - Corresponding knots concentrate in *curved regions of f*
- $\delta = -\infty$ means $\delta = c^2 - (a^2 + b^2) \gg 0$
  - Such points *spread out* through conical part of small *disk*.
  - Corresponding knots quasi-equispaced in *linear regions of f*
  - Corresponding function behaves as quasi cubic spline
  - Predicted by random features models

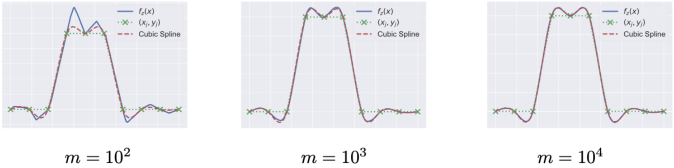


Figure 3: A cubic spline with vanishing second derivative at its endpoints (blue line) is approximated by a neural network ($\delta = -100$) while varying the number m of neurons.

Journal of Applied Physics ARTICLE scitation.org/journal/jap

On the optimization of knot allocation for B-spline parameterization of the dielectric function in spectroscopic ellipsometry data analysis

Cite as: J. Appl. Phys. **129**, 034903 (2021); doi: 10.1063/5.0035456
Submitted: 28 October 2020 · Accepted: 31 December 2020 ·
Published Online: 19 January 2021

View Online Export Citation CrossMark

D. V. Likhachev[a)]

AFFILIATIONS

GLOBALFOUNDRIES Dresden Module One LLC & Co. KG, Wilschdorfer Landstr. 101, D-01109 Dresden, Germany

Interpretation of this physics paper

- Scientist exploring functions of interest in his field
- Studies *splines with variable knots*.
- Shows
  - where function *flat or linear*:
    only need few knots sparsely spaced
  - where function *rapidly changing*:
    need many knots, densely spaced
- Adaptive knot positioning achieves:
  smaller error for given number of parameters:

  Parsimony
  Ockham's Razor

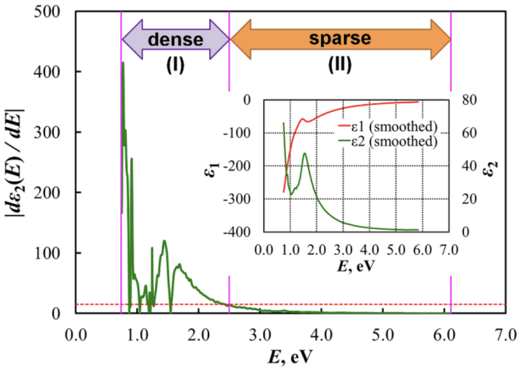


FIG. 7. Absolute value of the first derivative of the aluminum target DF (green solid line) and suggested density of knots (sparse or dense) based on selected threshold value (red dashed line). Thus, whole spectral range can be roughly divided into two separate intervals: (I) [0.74,2.5), with dense uniform knot distribution; (II) [2.5,6.1], with sparse uniform knot distribution. These intervals are marked with purple vertical lines. Inset shows the smoothed target DF $\varepsilon_{tgt}(E)$ of aluminum.

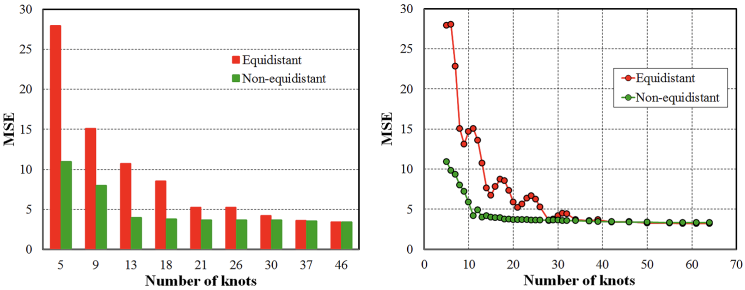


FIG. 8. Left: A comparison of the goodness-of-fit estimator at selected numbers of knots obtained based on the improved knot placement (green colored bars) with the assessment from the ordinary equidistant knot allocation (red colored bars). Right: Mean squared error as a function of total number of knots for the knot vector with small step sizes. Such small stepping reveals well-pronounced fluctuations in the MSE behavior observed for the equidistant knot allocation.

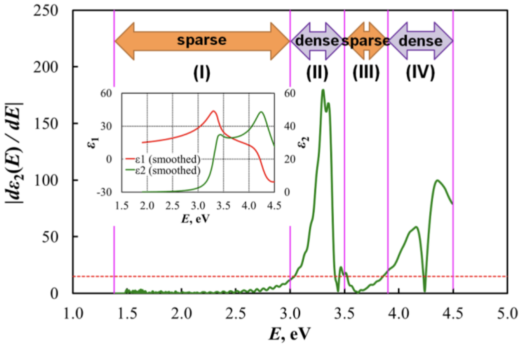


FIG. 11. Absolute value of the first derivative of the sSOI target DF (green solid line) and suggested density of knots (sparse or dense) based on selected threshold value (red dashed line). Thus, whole spectral range can be roughly divided into four separate intervals: (I) [1.4,3.0) and (III) (3.5,3.9), with sparse uniform knot distribution; (II) [3.0,3.5], and (IV) [3.9,4.6], with dense uniform knot distribution. These intervals are marked with purple vertical lines. The inset shows the smoothed target DF $\varepsilon_{tgt}(E)$ of sSOI.

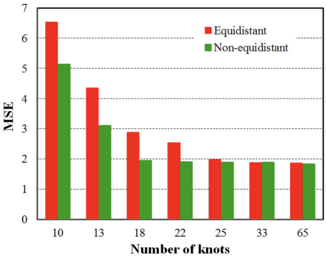


FIG. 12. A comparison of the goodness-of-fit estimator at selected numbers of knots obtained based on the improved knot placement (green colored bars) with the assessment from the ordinary equidistant knot allocation (red colored bars). Thus, the same performance can be achieved just with 18 non-uniformly distributed knots (MSE = 1.943) as compared to 25 uniformly allocated ones (MSE = 1.964).

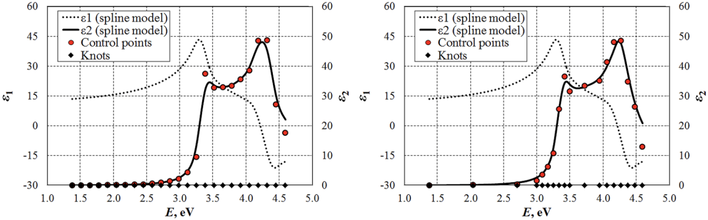


FIG. 13. Two B-spline representations of the sSOI dielectric function showing the control points and knots. Left: an equidistant knot placement for the whole analysis range from 1.38 eV to 4.59 eV (25 knots, MSE = 1.964); right: an optimized knot placement with four specially-spaced spectral regions (18 knots, MSE = 1.943).

Generalization benefits of parsimony 1/3

In estimating linear model:

$$y_i = f(x_i; \theta) + z_i, \qquad i = 1, \ldots, n.$$

Model is sum of relus at fixed knots (b_j).

$$f = \sum_j \theta_j relu(x - b_j)$$

Generalization error:

$$\begin{aligned} PMSE &= E\|Y - \hat{f}\|_2^2 \\ &= \sum_{i=1}^{n} E(y_i - \hat{f}(x_i))^2 \end{aligned}$$

where (x_i, y_i) fresh, out-of-sample data from above model.

Generalization benefits of parsimony 2/3

$$
\begin{aligned}
PMSE &= E\|Y - \hat{f}\|_2^2 \\
&= Bias^2 + Variance
\end{aligned}
$$

$$
\begin{aligned}
Variance &= tr(Cov(\hat{f})) \\
&= \sigma^2 \cdot \#\{\text{free parameters}\}
\end{aligned}
$$

Variability of predictions generated by this model.

$$Bias^2 = \|f - E\hat{f}\|_2^2$$

(Squared) Approximation error of underlying model at sample points.

Generalization benefits of parsimony 3/3

$$\begin{aligned} PMSE &= Bias^2 + Variance \\ Variance &= \sigma^2 \cdot \#\{\text{free parameters}\} \\ Bias^2 &= \|f - E\hat{f}\|_2^2 \end{aligned}$$

Adaptive knot placement

- minimize approximation error $\|f - E\hat{f}\|_2^2$ given # and placement knots.
- Same as: minimize $Bias^2$ for given variance.
- Same as: minimize PMSE for given number, placement of knots
- Global optimum of PMSE achieved by sweeping across hyperparameters
  - choose optimal # knots.
  - optimally position those knots
- Modern training aligns with this narrative

Conclusion

- 2-layer neural nets in dimension 1:
  special type of linear spline.
- Specifically *linear splines with variable knots*.
- Can fit any 'nice' function with *equispaced knots*.
  But might need many knots.
- Standard NN practice:
  dynamic knots + random initialization
  - Random initialization: Random Features Model.
  - Dynamics: move features where help most.
- Adaptive knot positioning: fewer knots
  heuristically better generalization