TTIC 31230, Fundamentals of Deep Learning

David McAllester

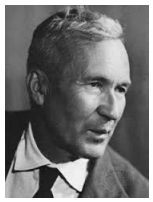
Generalization Theory

The Occam Generalization Guarantee

aka: The Free Lunch Theorem

Chomsky vs. Kolmogorov and Hinton

Noam Chomsky: Natural language grammar cannot be learned by a universal learning algorithm. This position is supported by the "no free lunch theorem".



Andrey Kolmogorov, Geoff Hinton: Universal learning algorithms exist. This position is supported by the "free lunch theorem".

The No Free Lunch Theorem

Without prior knowledge, such as universal grammar, it is impossible to make a prediction for an input you have not seen in the training data.

Proof: Select a predictor h uniformly at random from all functions from $\mathcal{X}$ to $\mathcal{Y}$ and then take the data distribution to draw pairs $(x, h(x))$ where x is drawn uniformly from $\mathcal{X}$. No learning algorithm can predict $h(x)$ where x does not occur in the training data.

The Occam Guarantee (Free Lunch Theorem)

Consider a classifier f written in Python with an arbitrarily large standard library.

Let $|f|$ be the number of bits needed to represent f (any compression algorithm is allowed).

The Occam Guarantee (Free Lunch Theorem)

$$0 \leq \mathcal{L}(h, x, y) \leq L_{\max}$$

$$\begin{aligned} \mathcal{L}(h) &= E_{(x,y)\sim \mathrm{Pop}}\, \mathcal{L}(h, x, y) \\ \hat{\mathcal{L}}(h) &= E_{(x,y)\sim \mathrm{Train}}\, \mathcal{L}(h, x, y) \end{aligned}$$

Theorem: With probability at least $1 - \delta$ over the draw of the training data the following holds simultaneously for all f.

$$\mathcal{L}(f) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(f) + \frac{5L_{\max}}{N_{\mathrm{Train}}}\left((\ln 2)|f| + \ln\frac{1}{\delta}\right)\right)$$

Occam Guarantee (Probability Form)

Code length is inter-convertable with with probability.

$$P(h) = 2^{-|h|} \quad \text{or} \quad |h| = -\log_2 P(h)$$

Instead of fixing the language (e.g., Python with a large library) we fix a prior $P(h)$.

Theorem: With probability at least $1 - \delta$ over the draw of training data the following holds simultaneously for all h.

$$\mathcal{L}(h) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(h) + \frac{5L_{\max}}{N_{\text{Train}}}\left(-\ln \delta P(h)\right)\right)$$

Occam vs. Bayes

For $\mathcal{L}(h, x, y) = -\ln P_h(y|x) \leq L_{\max}$ we have

$$\text{Occam:} \qquad \mathcal{L}(h) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(h) + \frac{5L_{\max}}{N_{\text{Train}}}(-\ln \delta P(h))\right)$$

$$h^* = \operatorname*{argmin}_h \hat{\mathcal{L}}(h) + \frac{5L_{\max}}{N_{\text{Train}}}(-\ln P(h))$$

$$\text{Bayes:} \qquad h^* = \operatorname*{argmax}_h P(h|\text{Train})$$

$$h^* = \operatorname*{argmin}_h \hat{\mathcal{L}}(h) + \frac{1}{N_{\text{Train}}}(-\ln P(h))$$

Proof

Define

$$\epsilon(h) = \sqrt{\frac{2\mathcal{L}(h)\left(-\ln \delta P(h)\right)}{L_{\max} N_{\text{Train}}}}.$$

By the relative Chernov bound we have

$$P_{\text{Train}\sim\text{Pop}}\left(\frac{\hat{\mathcal{L}}(h)}{L_{\max}} \leq \frac{\mathcal{L}(h)}{L_{\max}} - \epsilon(h)\right) \leq e^{-N_{\text{Train}}\frac{\epsilon(h)^2 L_{\max}}{2\mathcal{L}(h)}} = \delta P(h).$$

Proof

$$P_{\text{Train}\sim\text{Pop}}\left(\hat{\mathcal{L}}(h) \leq \mathcal{L}(h) - L_{\max}\epsilon(h)\right) \leq \delta P(h).$$

$$P_{\text{Train}\sim\text{Pop}}\left(\exists h\ \hat{\mathcal{L}}(h) \leq \mathcal{L}(h) - L_{\max}\epsilon(h)\right) \leq \sum_{h} \delta P(h) = \delta$$

$$P_{\text{Train}\sim\text{Pop}}\left(\forall h\ \mathcal{L}(h) \leq \hat{\mathcal{L}}(h) + L_{\max}\epsilon(h)\right) \geq 1 - \delta$$

Proof

$$\mathcal{L}(h) \leq \widehat{\mathcal{L}}(h) + L_{\max}\sqrt{\mathcal{L}(h)\left(\frac{2L_{\max}\left(-\ln \delta P(h)\right)}{N_{\text{Train}}}\right)}$$

using

$$\sqrt{ab} = \inf_{\lambda>0} \frac{a}{2\lambda} + \frac{\lambda b}{2}$$

we get

$$\mathcal{L}(h) \leq \widehat{\mathcal{L}}(h) + \frac{\mathcal{L}(h)}{2\lambda} + \frac{\lambda L_{\max}\left(-\ln \delta P(h)\right)}{N_{\text{Train}}}$$

Proof

$$\mathcal{L}(h) \leq \widehat{\mathcal{L}}(h) + \frac{\mathcal{L}(h)}{2\lambda} + \frac{\lambda L_{\max}\left(-\ln \delta P(h)\right)}{N_{\text{Train}}}$$

Solving for $\mathcal{L}(h)$ yields

$$\mathcal{L}(h) \leq \frac{1}{1 - \frac{1}{2\lambda}} \left(\hat{\mathcal{L}}(h) + \frac{\lambda L_{\max}}{N_{\text{Train}}} \left(-\ln \delta P(h)\right) \right)$$

Setting $\lambda = 5$ brings the leading factor to $10/9$ which seems sufficiently close to 1 that larger values of λ need not be considered.

The PAC-Bayes Guarantee

Let p be any "prior" and q be any "posterior" on any (possibly continuous) model space. Define

$$L(q) = E_{h \sim q} \, L(h)$$

$$\hat{L}(q) = E_{h \sim q} \, \hat{L}(h)$$

For any p and any $\lambda > \frac{1}{2}$, with probability at least $1 - \delta$ over the draw of the training data, the following holds simultaneously for all q.

$$L(q) \leq \frac{1}{1 - \frac{1}{2\lambda}} \left(\hat{L}(q) + \frac{\lambda L_{\max}}{N_{\text{Train}}} \left(KL(q, p) + \ln \frac{1}{\delta} \right) \right)$$

Adding Noise Simulates Limiting Precision

Assume $0 \leq \mathcal{L}(\Phi, x, y) \leq L_{\max}$.

Define:

$$\mathcal{L}_\sigma(\Phi) \;=\; E_{\,(x,y)\sim \text{Pop},\ \epsilon\sim\mathcal{N}(0,\sigma)^d}\ \mathcal{L}(\Phi+\epsilon, x, y)$$

$$\hat{\mathcal{L}}_\sigma(\Phi) \;=\; E_{\,(x,y)\sim \text{Train},\ \epsilon\sim\mathcal{N}(0,\sigma)^d}\ \mathcal{L}(\Phi+\epsilon, x, y)$$

Theorem: With probability at least $1-\delta$ over the draw of training data the following holds **simultaneously** for all Φ.

$$\mathcal{L}_\sigma(\Phi) \;\leq\; \frac{10}{9}\left(\hat{\mathcal{L}}_\sigma(\Phi) + \frac{5L_{\max}}{N_{\text{Train}}}\left(\frac{||\Phi||^2}{2\sigma^2} + \ln\frac{1}{\delta}\right)\right)$$

Non-Vacuous Generalization Guarantees

Model compression has recently been used to achieve "non-vacuous" PAC-Bayes generalization guarantees for ImageNet classification — error rate guarantees less than 1.

Non-Vacuous PAC-Bayes Bounds at ImageNet Scale.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, Peter Orbanz

ICLR 2019

Implicit Regularization

Any stochastic learning algorithm, such as SGD, determines a stochastic mapping from training data to models.

The algorithm, especially with early stopping, can implicitly incorporate a preference or bias for models.

Implicit Regularization in Linear Regression

Linear regression (minimizing the L_2 loss of a linear predictor) where we have more parameters than data points has many solutions.

But SGD converges to the minimum norm solution (L_2-regularized solution) without the need for explicit regularization.

Implicit Regularization in Linear Regression

For linear regression SGD maintains the invariant that Φ is a linear combination of the (small number of) training vectors.

Any zero-loss (squared loss) solution can be projected on the span of training vectors to give a smaller (or no larger) norm solution.

It can be shown that when the training vectors are linearly independent any zero loss solution in the span of the training vectors is a least-norm solution.

Implict Priors

Let A be any algorithm mapping a training set Train to a probability density $p(\Phi|\text{Train})$.

For example, the algorithm might be SGD where we add a small amount of noise to the final parameter vector so that $p(\Phi|\text{Train})$ is a smooth density.

But in general we can consider any leaning algorithm that produces a smooth density $p(\Phi|\text{Train})$.

Implicit Priors

Drawing Train from Pop^N and Φ from $P(\Phi|\text{Train})$ defines a joint distribution on Train and Φ. We can take the marginal distribution on Φ to be a prior distribution (independent of any training data).

$$p(\Phi) = E_{\left(\text{Train}\sim\text{Pop}^N\right)} \; p(\Phi \; |\text{Train})$$

It can be shown that the implicit prior $p(\Phi)$ is an optimal prior for the PAC-Bayesian generalization guarantees applied to the algorithm defining $p(\Phi|\text{Train})$

A PAC-Bayes Analysis of Implicit Regularization

$$\mathcal{L}(\text{Train}) = E_{\langle x, y\rangle \sim \text{Pop},\ \Phi \sim p(\Phi|\text{Train})}\, \mathcal{L}(\Phi, x, y)$$

$$\hat{\mathcal{L}}(\text{Train}) = E_{\langle x, y\rangle \sim \text{Train},\ \Phi \sim p(\Phi|\text{Train})}\, \mathcal{L}(\Phi, x, y)$$

A PAC-Bayes Analysis of Implicit Regularization

With probability at least $1-\delta$ over the draw of Train we have

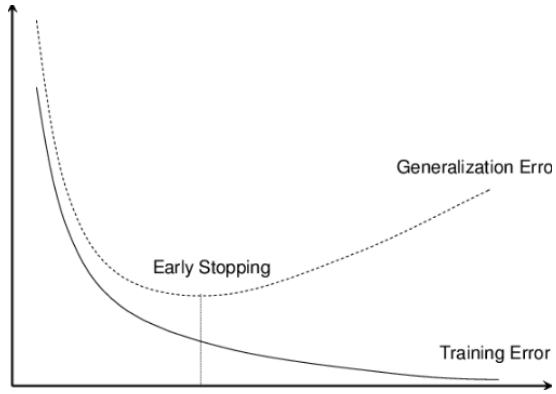
$$\mathcal{L}(\text{Train}) \leq \frac{10}{9}\left(\hat{\mathcal{L}}(\text{Train}) + \frac{5L_{\max}}{N_{\text{Train}}}\left(KL(p(\Phi|\text{Train}), p(\Phi))\right) + \ln\frac{1}{\delta}\right)$$

$$= \frac{10}{9}\left(\hat{\mathcal{L}}(\text{Train}) + \frac{5L_{\max}}{N_{\text{Train}}}\left(I(\Phi, \text{Train}) + \ln\frac{1}{\delta}\right)\right)$$

There is no obvious way to calculate this guarantee.

However, it can be shown that $p(\Phi)$ is the optimal PAC-Bayeisan prior for the given algorithm run on training data data drawn from Pop^N.

Over Confidence and Calibration



Validation error is larger than training error when we stop.

The model probabilities are tuned on training data statistics.

The probabilities are tuned to an unrealistically low error rate and are therefore over-confident.

This over-confidence occurs before the stopping point and damages validation loss (as opposed to validation error).

Shrinkage: L_2 regularization

We first give a Bayesian derivation. We put a prior probability on Φ and maximize the a-posteriori probability (MAP).

$$
\begin{aligned}
\Phi^* &= \operatorname*{argmax}_{\Phi} \; p(\Phi | \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle) \\
&= \operatorname*{argmax}_{\Phi} \; \frac{p(\Phi, \; \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)}{p(\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)} \\
&= \operatorname*{argmax}_{\Phi} \; p(\Phi, \; \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)
\end{aligned}
$$

Shrinkage: L_2 regularization

$$
\begin{aligned}
\Phi^* &= \operatorname*{argmax}_{\Phi}\ p(\Phi,\ \langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle) \\
&= \operatorname*{argmax}_{\Phi}\ p(\Phi) \prod_i \mathrm{Pop}(x_i) P_\Phi(y_i|x_i) \\
&= \left(\prod_i p(x_i)\right) \operatorname*{argmax}_{\Phi}\ p(\Phi) \prod_i P_\Phi(y_i|x_i) \\
&= \operatorname*{argmax}_{\Phi}\ p(\Phi) \prod_i P_\Phi(y_i|x_i)
\end{aligned}
$$

Shrinkage: L_2 Regularization

$$\begin{aligned}\Phi^* &= \underset{\Phi}{\operatorname{argmax}}\ p(\Phi) \prod_i P_\Phi(y_i|x_i) \\ &= \underset{\Phi}{\operatorname{argmin}} \sum_i -\ln P_\Phi(y_i|x_i) - \ln p(\Phi)\end{aligned}$$

We now take a Gaussian prior

$$p(\Phi) \propto \exp\left(-\frac{||\Phi||^2}{2\sigma^2}\right)$$

Shrinkage: L_2 Regularization

$$\Phi^* = \operatorname*{argmin}_{\Phi} \sum_{i=1}^{n} -\ln P_\Phi(y_i|x_i) + \frac{||\Phi||^2}{2\sigma^2}$$

$$= \operatorname*{argmin}_{\Phi} \frac{1}{N} \left(\sum_{i=1}^{n} -\ln P_\Phi(y_i|x_i) + \frac{||\Phi||^2}{2\sigma^2} \right)$$

$$= \operatorname*{argmin}_{\Phi} \left(E_{\langle x,\, y\rangle \sim \text{Train}} \; -\ln P_\Phi(y|x) \right) + \frac{1}{2N\sigma^2} ||\Phi||^2$$

Shrinkage: L_2 Regularization

$$\nabla_\Phi \; E_{(x,y)\sim\text{Train}} \left(\mathcal{L}(\Phi, x, y) + \frac{||\Phi||^2}{2N\sigma^2} \right)$$

$$= E_{(x,y)\sim\text{Train}} \left(g(\Phi, x, y) + \frac{\Phi}{N\sigma^2} \right)$$

$$\Phi_{i+1} = \Phi_i - \eta \left(\hat{g}_i - \frac{1}{N\sigma^2}\Phi_i \right)$$

The last term in the update equation is called "shrinkage".

Decoupling Shrikage and Training Data Size

The PyTorch parameters are the learning rate η and the **weight decay** γ:

$$\Phi_{i+1} = \Phi_i - \eta\left(\hat{g} + \frac{1}{N\sigma^2}\Phi_i\right) \qquad = \Phi_i - \eta(\hat{g} - \gamma\Phi_i)$$

To make SGD with shrinkage robust to changes in training size, batch size, learning rate (temperature) and momentum we can use

$$\eta = (1-\mu)B\eta_0 \qquad \gamma = \frac{1}{N_{\text{Train}}\sigma^2}$$

where η_0 is the robust temperature parameter and σ^2 is the robust shrinkage parameter.

Ensembles

Train several models Ens $= (\Phi_1, \ \ldots, \ \Phi_K)$ from different initializations and/or under different meta parameters.

We define the ensemble model by

$$P_{\text{Ens}}(y|x) = \frac{1}{K} \sum_k \ P_{\Phi_k}(y|x) = E_k \ P_k(y|x)$$

Ensemble models almost always perform better than any single model.

Ensembles Under Cross Entropy Loss

$$
\begin{aligned}
\mathcal{L}\left(P_{\text{Ens}}\right) &= E_{\langle x,\, y\rangle \sim \text{Pop}} \quad -\ln P_{\text{Ens}}(y|x) \\
&= E_{\langle x,\, y\rangle \sim \text{Pop}} \quad -\ln E_k P_k(y|x) \\
&\leq E_{\langle x,\, y\rangle \sim \text{Pop}}\, E_k \quad -\ln P_k(y|x) \\
&= E_k\, \mathcal{L}(P_k)
\end{aligned}
$$

Ensembles Under Cross Entropy Loss

It is important to note that

$$-\ln E_k \; P_k(y|x) \quad \leq E_k - \ln P_k(y|x)$$

for each individual pair $\langle x, y \rangle$.

$\forall z \; f(z) \leq g(z)$ is stronger than $(E_z \; f(z)) \leq (E_z \; g(z))$.

This may explain why in practice an ensemble model is typically better than any single component model.

Double Descent

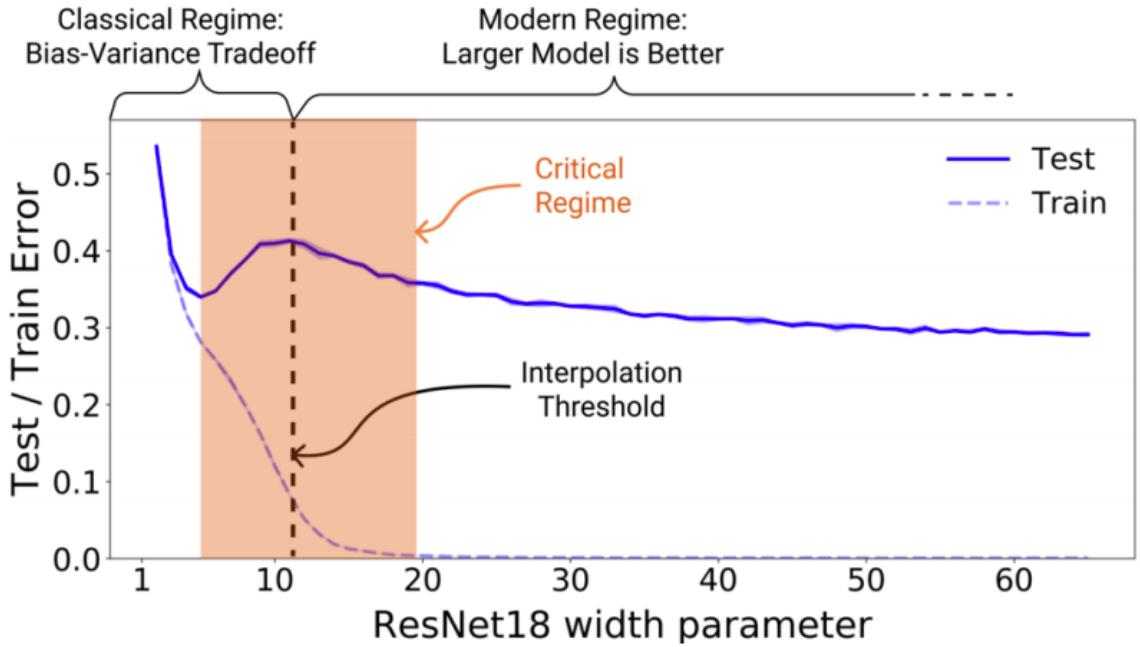
Reconciling modern machine learning practice and the bias-variance trade-off

Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal, arXiv December 2018.

Deep Double Descent: Where Bigger Models and More Data Hurt

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, Ilya Sutskever, ICLR 2020

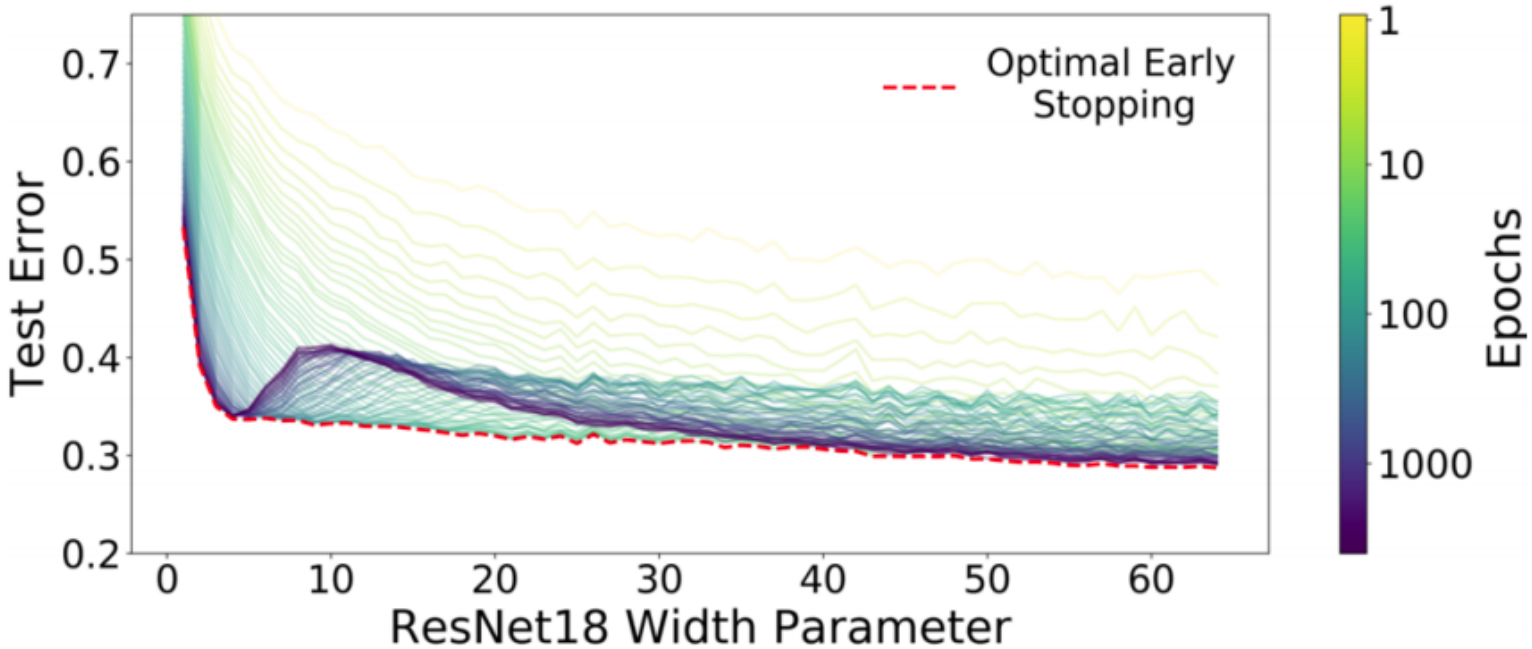
Double Descent



Deep Double Descent: Where Bigger Models and More Data Hurt

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, Ilya Sutskever, ICLR 2020

Double Descent



END