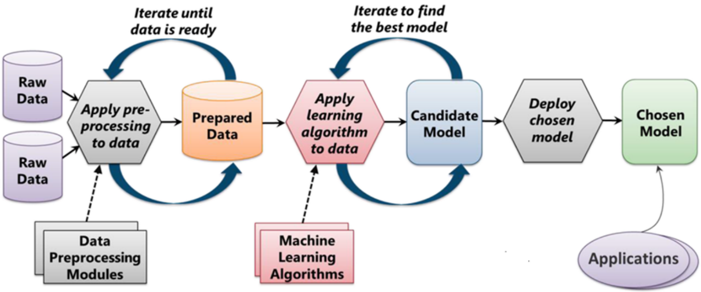
11-695: AI Engineering
Other Techniques

LTI/SCS

Spring 2020

Outline

Cleansing Data is often overlooked

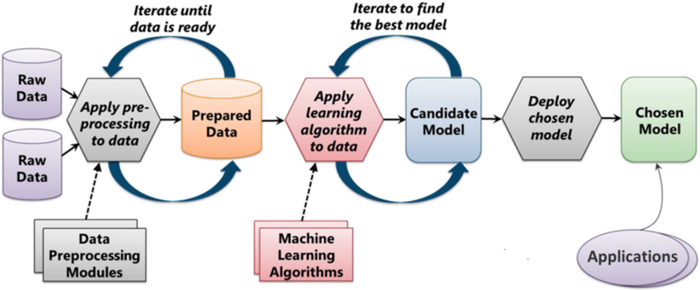


- The very first process, that is *essential*
- Data in real world are often noisy and unclean

Image credit: David Chappell

Cleansing Data is often overlooked

CarnegieMellon

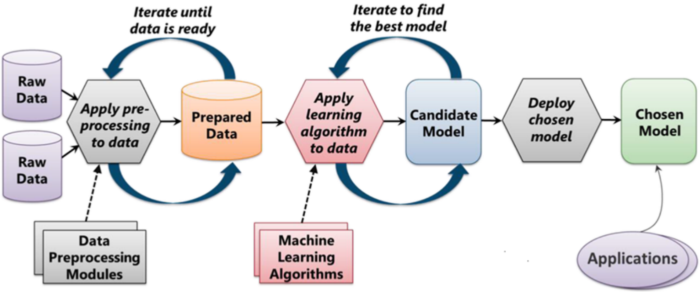


- The very first process, that is *essential*
- Data in real world are often noisy and unclean
- Even super models cannot perform well on unclean data

Image credit: David Chappell

Cleansing Data is often overlooked

Carnegie Mellon



- The very first process, that is *essential*
- Data in real world are often noisy and unclean
- Even super models cannot perform well on unclean data
- Sometimes it takes the majority of work in the full pipeline

Image credit: David Chappell

Common Techniques

- Examine carefully and build some useful statistics, charts, plots, ...
- Remove noisy, irrelevant samples
- Remove duplicates
- Fix labeling errors, such as matching/reconciliating 2 groups which should be one
- Handle outliers and missing data properly: dropping should be the last resort
- Some data has special techniques, *e.g.* for texts, sometimes we need to remove punctuation, stop words, special characters, ...

Takeaway Messages

- It is very important that we **understand** well your data before applying any techniques
- **Don't underestimate** data cleansing in practice

Outline

CarnegieMellon

1. Data Cleansing
2. Overfitting and Beyond
3. Batch Normalization

Bias-Variance Decomposition (MSE)[2]

- Expected risk or error[1] for a new sample x:

$$\mathbb{E}_D\left[\left(y - \hat{\mathbf{f}}(x, D)\right)^2\right] = \left(\mathrm{Bias}_D\left[\hat{\mathbf{f}}(x, D)\right]\right)^2 + \mathrm{Var}_D\left[\hat{\mathbf{f}}(x, D)\right] + \sigma^2$$

where

$$y = \mathbf{f}(x) + \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2)$$

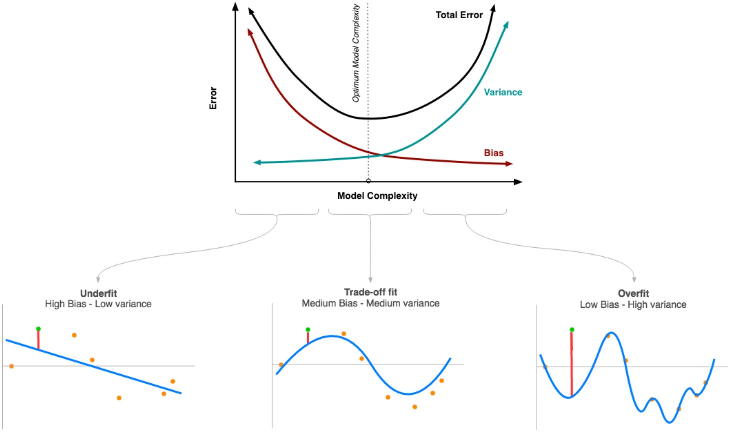
- $\mathrm{Bias}_D\left[\hat{\mathbf{f}}(x, D)\right] = \mathbb{E}_D\left[\hat{\mathbf{f}}(x, D)\right] - \mathbf{f}(x)$
- $\mathrm{Var}_D\left[\hat{\mathbf{f}}(x, D)\right] = \mathbb{E}_D\left[\hat{\mathbf{f}}^2(x, D)\right] - \left(\mathbb{E}_D\left[\hat{\mathbf{f}}(x, D)\right]\right)^2$
- σ^2 is *irreducible*

[1] Derivation credit: Trevor Hastie *et al.* The Elements of Statistical Learning (book).

[2] Derivation details: `https://en.wikipedia.org/wiki/Bias-variance_tradeoff`

Underfit, Goodfit, Perfectfit and Overfit

Carnegie Mellon

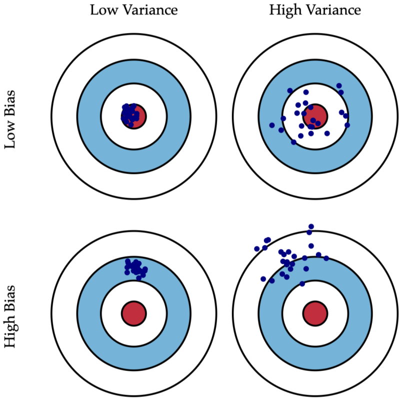


- Note the total error is a curve, not a constant as in many tradeoffs

Image credit: Scott Fortmann-Roe and ebs.cat

Underfit, Goodfit, Perfectfit and Overfit

Carnegie Mellon



- Good models should have both low bias and low variance
- Underfit: high empirical (train) risk (error or averaged loss values)
- Overfit: small empirical (train) risk but large true (test) risk

Image credit: Scott Fortmann-Roe

How about model complexity?[3]

Carnegie Mellon

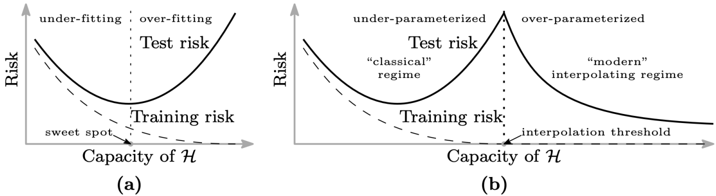


Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

- Overparameterization (↑ complexity) leads to a nice behavior too
- Classical bias-variance curve only tells a part of the story

[3] https://arxiv.org/pdf/1812.11118.pdf

Image credit: Mikhail Belkin *et al.*

Double Descent Curve: extended story

Carnegie Mellon

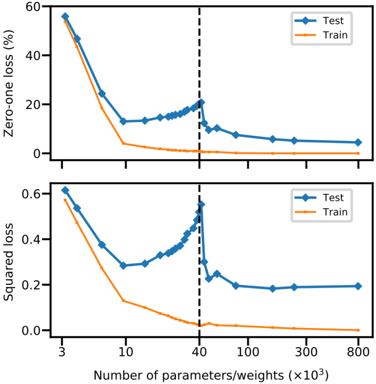


Figure 4: **Double descent risk curve for fully connected neural network on MNIST.** Training and test risks of network with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \cdot 10^3$, $d = 784$, $K = 10$ classes). The number of parameters is $(d+1) \cdot H + (H+1) \cdot K$. The interpolation threshold (black dotted line) is observed at $n \cdot K$.

Image credit: Mikhail Belkin *et al.*

Weights Space and Norm

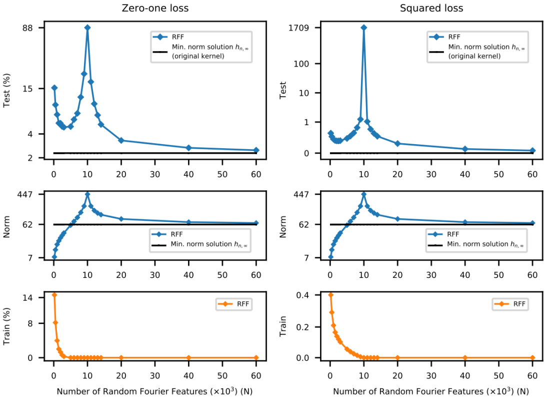


Figure 2: **Double descent risk curve for RFF model on MNIST.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes). The interpolation threshold is achieved at $N = 10^4$.

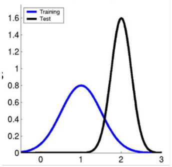
- Space increases will lead to lower norm of the solutions

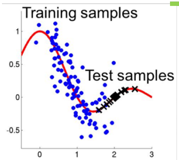
Image credit: Mikhail Belkin *et al.*

Outline

Carnegie Mellon

Batch Normalization[4]





Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

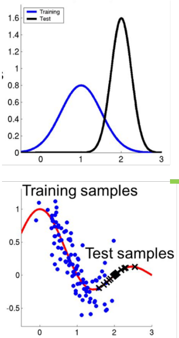
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Covariate Shift: different distributions between train and test sets

[4] https://arxiv.org/pdf/1502.03167.pdf

Image credit: Wei F an, Masashi Sugiyama and Sergey Ioffe, Christian Szegedy

Batch Normalization[4]



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

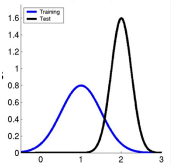
- Covariate Shift: different distributions between train and test sets
- Internal Covariate Shift: a covariate shift between 2 layers in NN

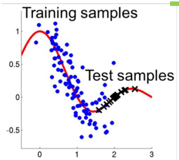
[4] `https://arxiv.org/pdf/1502.03167.pdf`

Image credit: Wei F an, Masashi Sugiyama and Sergey Ioffe, Christian Szegedy

Batch Normalization[4]

Carnegie Mellon





Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

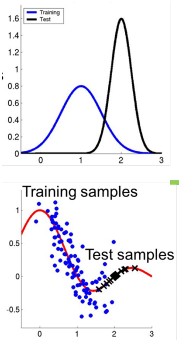
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Covariate Shift: different distributions between train and test sets
- Internal Covariate Shift: a covariate shift between 2 layers in NN
- Fix: scale unto unit Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$

[4] https://arxiv.org/pdf/1502.03167.pdf

Image credit: Wei F an, Masashi Sugiyama and Sergey Ioffe, Christian Szegedy

Batch Normalization (cont'd)



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

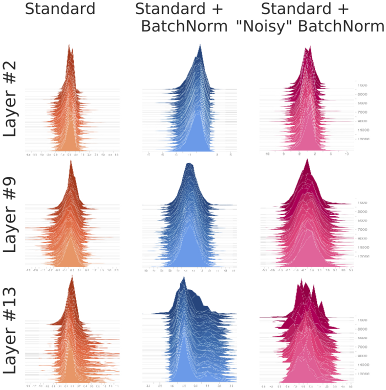
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

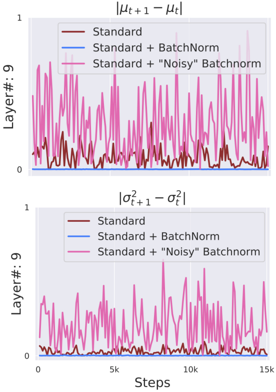
- Training time: Take batch, compute $(\mu_{tr}, \sigma_{tr}) \rightarrow$ new training values $\rightarrow$ feed to the next layer
- Test time: use the whole test set instead of each batch

Image credit: Ioffe and Szegedy

Argument: Not about Co-Variate Shift[5]

Carnegie Mellon



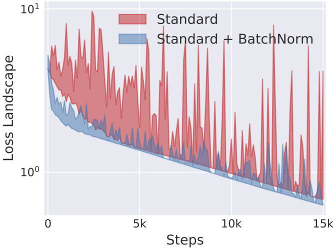


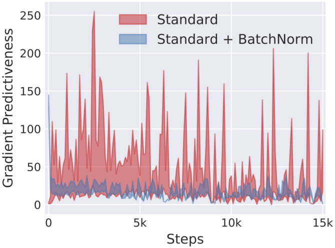
- Adding co-variate shift does not reduce the effect of batchnorm

[5] https://arxiv.org/abs/1805.11604

Image credit: Santurkar and Tsipras

Argument: Not about Co-Variate Shift





- Adding co-variate shift does not reduce the effect of batchnorm
- Batch norm seems to make the loss landscapes smoother for optimization

Image Credit: Santurkar and Tsipras