Evaluating Performance

Zhiyao Duan

Associate Professor of ECE and CS

University of Rochester

Some figures are copied from the following books

- **LWLS** - Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, *Machine Learning: A First Course for Engineers and Scientists*, Cambridge University Press, 2022.
- **WBK** - Jeremy Watt, Reza Borhani, Aggelos K. Katsaggelos, Machine Learning Refined: Foundations, Algorithms, and Applications (1st Edition), Cambridge University Press, 2016.

Motivating Questions

- How to evaluate performance of supervised models?
- What metrics to use?
- How to use those metrics?
- How to interpret evaluation results?

Classification Accuracy

- y: ground-truth class label, $\hat{y}$: predicted class label
  - Correctly classified: $y=\hat{y}$
  - Misclassified: $y \neq \hat{y}$

$$\text{Acc} = \frac{\#\text{correctly classified}}{\#\text{total examples}}$$

- $0 \leq Acc \leq 1$
- What is the average accuracy of a random guess for C-class classification?
  - 1/C

Balanced Accuracy

- Is classification accuracy a good metric for a highly imbalanced classification problem (e.g., 99% healthy + 1% ill)?
  - A naïve classifier that always diagnoses unseen patients as healthy achieves 99% accuracy, but it misclassifies all actual patients!

- Balanced accuracy: average over per-class accuracy

$$Acc_{balanced} = Average\left(\frac{\text{\#correctly classified for Class c}}{\text{\#total examples in Class c}}\right)$$

- $0 \leq Acc_{balanced} \leq 1$

- The above naïve classifier would only get 1/C balanced accuracy on average

Confusion Matrix

		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Figure from (Grandini, Bagli & Visani, “Metrics for multi-class classification: an overview”, 2020)

Precision & Recall

Be careful about which axis is ground-truth and which is predicted!

	$y=-1$	$y=1$	*total*
$\hat{y}(\mathbf{x})=-1$	True neg (TN)	False neg (FN)	N*
$\hat{y}(\mathbf{x})=1$	False pos (FP)	True pos (TP)	P*
total	N	P	n

- If we treat the positive class as the target class

$$Precision = \frac{TP}{P^*} = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{P} = \frac{TP}{TP+FN}$$

$$F_1 = \frac{2 \cdot Precision \cdot recall}{precision + recall}$$

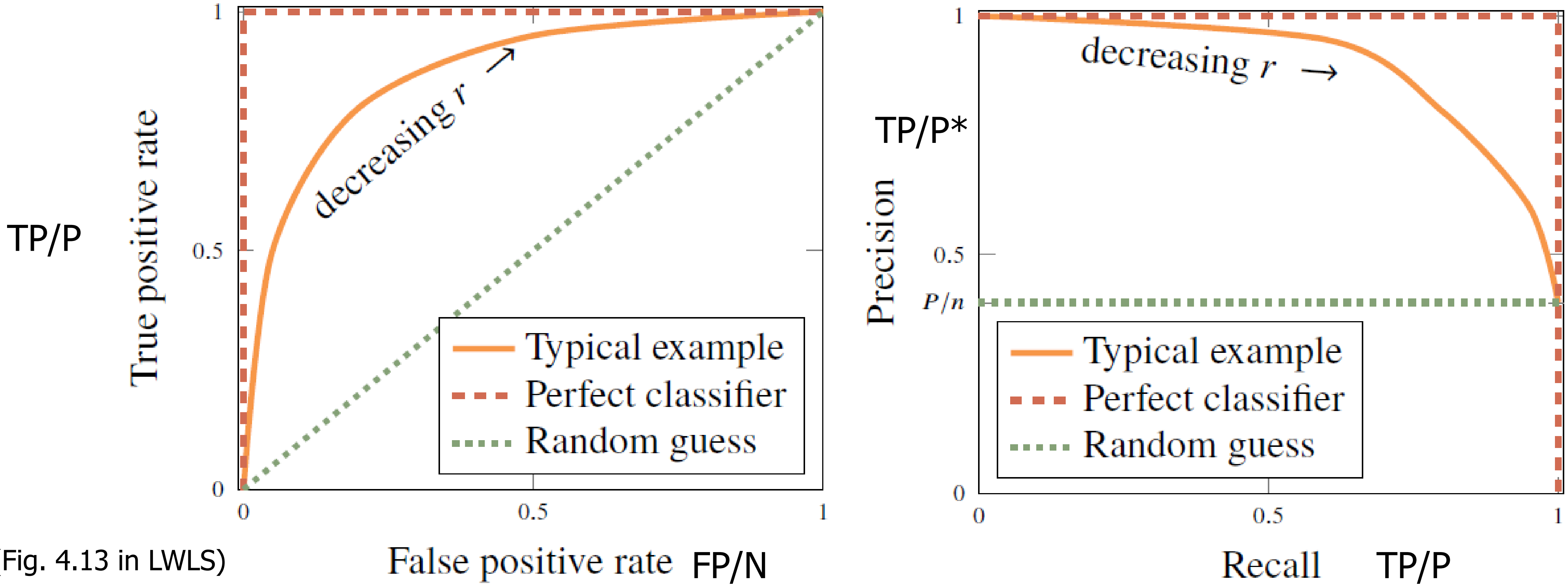
Common Terms related to Confusion Matrix

Ratio	Name
FP/N	False positive rate, Fall-out, Probability of false alarm
TN/N	True negative rate, Specificity, Selectivity
TP/P	True positive rate, Sensitivity, Power, *Recall*, Probability of detection
FN/P	False negative rate, Miss rate
TP/P*	Positive predictive value, *Precision*
FP/P*	False discovery rate
TN/N*	Negative predictive value
FN/N*	False omission rate
P/n	Prevalence
(FN + FP)/n	*Misclassification rate*
(TN + TP)/n	Accuracy, 1 − misclassification rate
2TP/(P* + P)	F_1 *score*
$(1 + \beta^2)$TP/($(1 + \beta^2)$TP + β^2FN + FP)	F_β *score*

(Table 4.1 in LWLS)

ROC Curve & Precision-Recall Curve

- Many classifiers uses a threshold r as the last step of classification
  - Decreasing r classifies more examples to the positive class
  - Area under the ROC curve (ROC-AUC): larger is better



(Fig. 4.13 in LWLS)

(a) The ROC curve

(b) The precision–recall curve

Regression Metrics

- Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

- Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}$$

- Mean Absolute Deviations (MAD), also called Mean Absolute Error (MAE)

$$MAD = \frac{1}{N}\sum_{i=1}^{N}\left|y^{(i)} - \hat{y}^{(i)}\right|$$

How to use these metrics?

- These metrics need to be computed on some data points
  - What are the differences between the metrics on training, validation and test sets?
- Training set: used to train the model
  - Make sure performance improves as training goes on and reaches a good level
  - Otherwise: underfitting - there are bugs in the training process, or the model is not appropriate, e.g., logistic regression for classes with intrinsically nonlinear boundaries
- Validation set: used to 1) tune hyper-parameters of model, and 2) decide when to stop training
  - Make sure validation performance is not too much lower than training performance, and stop training iterations when validation performance starts to decrease
  - Otherwise: overfitting – 1) model is too complex/flexible for the data, 2) training is too long
- Test set: used to report performance to customer
  - Should not be used in training or tuning hyperparameters

Randomness in Metrics

- Data points are randomly sampled from their underlying distribution
  - Computing metrics on different sets → different values
  - Training on different training sets → different model parameters
  - Tuning on different validation sets → different model hyperparameters

- Given an error definition between prediction and ground-truth $E(\hat{y}, y)$
  - Classification error: $E(\hat{y}, y) = \begin{cases} 0 & if\, \hat{y} = y \\ 1 & if\, \hat{y} \neq y \end{cases}$
  - Squared error for regression: $E(\hat{y}, y) = (\hat{y} - y)^2$
  - There can be many other error definitions

- We care about the error on new (unseen) examples, i.e., generalization!

Expected Error

- Assume data $(\boldsymbol{x}, y)$ follows distribution $p(\boldsymbol{x}, y)$
- Expected error of model trained on $\mathcal{T}$ and evaluated on new data (i.e., averaging over data distribution)

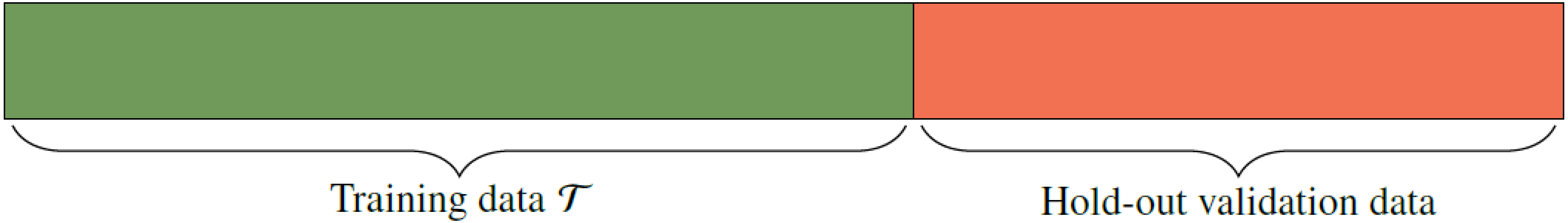
$$E_{new}(\mathcal{T}) \triangleq \mathbb{E}_{\boldsymbol{x},y}[E(\hat{y}(\boldsymbol{x};\mathcal{T}), y)]$$
$$= \int E(\hat{y}(\boldsymbol{x};\mathcal{T}), y) p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

- But $\mathcal{T}$ is also random.
- Take another expectation (i.e., averaging again) over all possible instantiations of training set

$$\bar{E}_{new} = \mathbb{E}_{\mathcal{T}}[E_{new}(\mathcal{T})]$$

But we do not know the data distribution!

- We can only estimate $E_{new}(\mathcal{T})$ and $\bar{E}_{new}$ on samples



(Fig. 4.1 in LWLS)

- Training error: $E_{train}(\mathcal{T}) \triangleq \frac{1}{N}\sum_{i=1}^{N} E\left(\hat{y}\left(\boldsymbol{x}^{(i)};\mathcal{T}\right), y^{(i)}\right)$
- Validation error: $E_{hold-out}(\mathcal{T}) \triangleq \frac{1}{N_v}\sum_{i=1}^{N_v} E\left(\hat{y}\left(\boldsymbol{x}_v^{(i)};\mathcal{T}\right), y_v^{(i)}\right)$
- Which is a better estimate for $E_{new}(\mathcal{T})$?
- Practice tips: shuffle data before splitting

K-Fold Cross Validation

(Fig. 4.2 in LWLS)

$\overline{\text{average} = E_{k\text{-fold}}}$

- The k models are trained on different (k-1 folds) training data
- Better estimate for $\bar{E}_{new} = \mathbb{E}_{\mathcal{T}}[E_{new}(\mathcal{T})]$, if hyper-parameters are not tuned on validation splits
- Practice tips: 1) shuffle data before splitting; 2) train on all data to deliver

Generalization Gap

- Expected training error: $\bar{E}_{train} \triangleq \mathbb{E}_{\mathcal{T}}[E_{train}(\mathcal{T})]$
- Expected test error: $\bar{E}_{new} \triangleq \mathbb{E}_{\mathcal{T}}[E_{new}(\mathcal{T})]$

- Generalization gap is the performance gap between training and test data

$$generalizatio\ gap \triangleq \bar{E}_{new} - \bar{E}_{train}$$

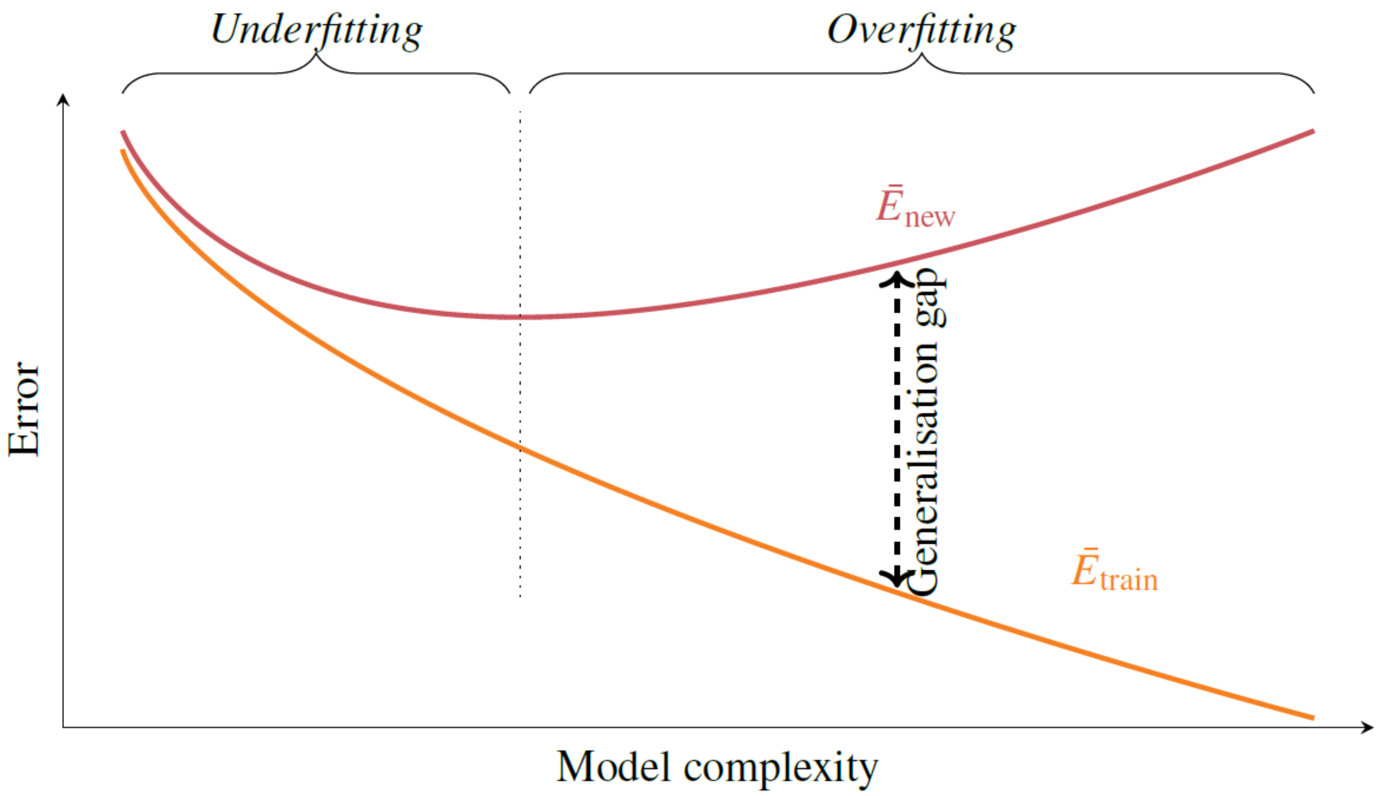
- Training error - generalization gap decomposition

$$\bar{E}_{new} = \bar{E}_{train} + generalizatio\ gap$$

Model Complexity Affects Generalization Gap

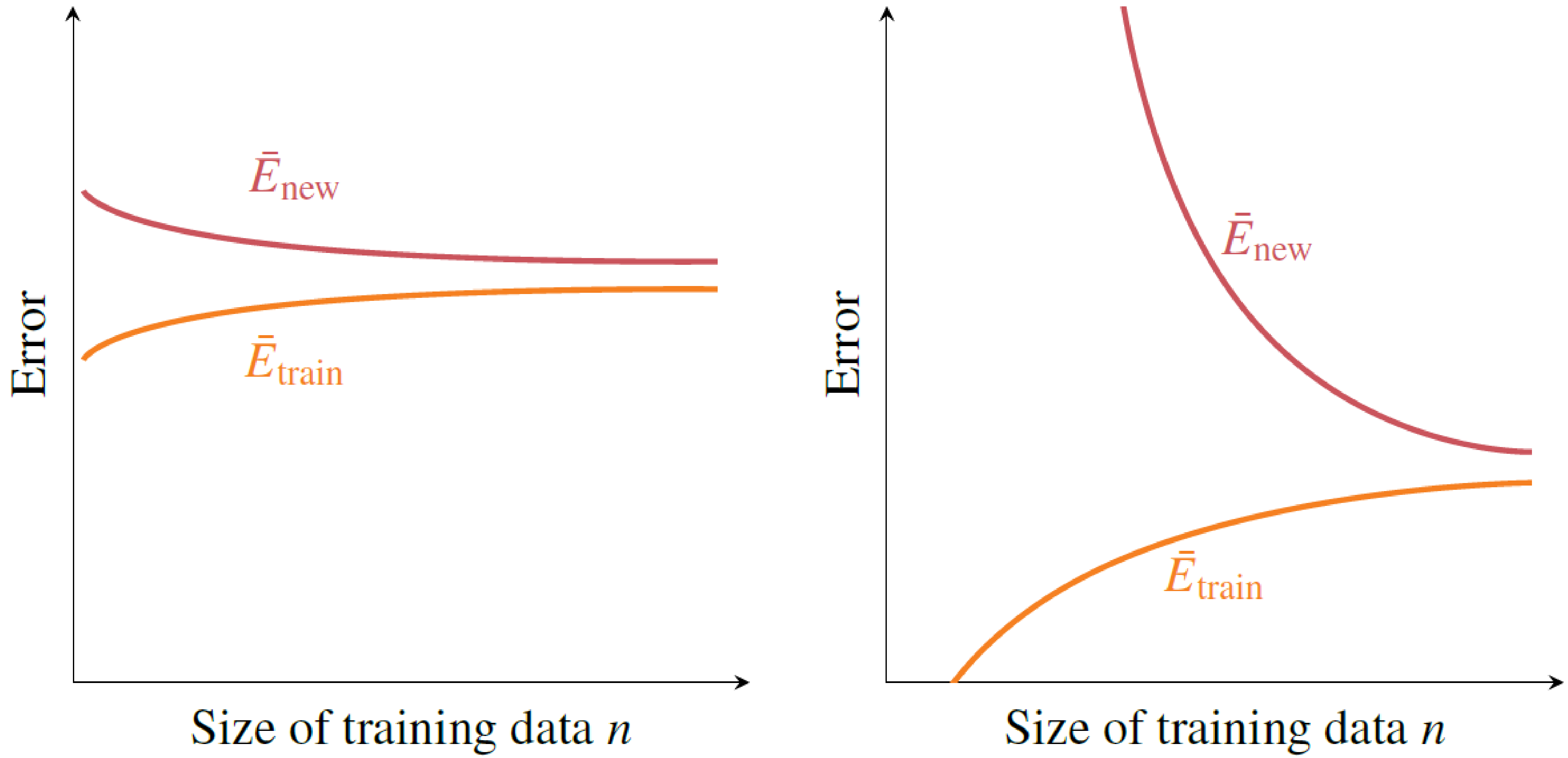
- Model complexity (flexibility) is vaguely defined about how much a model adapts to training data
  - High complexity: e.g., deep neural network, deep trees, k-NN with small k
  - Low complexity: e.g., logistic regression, k-NN with large k

- Related to the number of learnable parameters and the strength of regularization
- Some measures
  - Vapnik-Chervonenkis (VC) dimension
  - Minimum Description Length (MDL)



(Fig. 4.3 in LWLS)

Size of Training Set Affects Generalization Gap



(a) Simple model

(b) Complex model

(Fig. 4.6 in LWLS)

How to reduce $\bar{E}_{new}$?

$$\bar{E}_{new} = \bar{E}_{train} + generalizatio\ gap$$

- If training error is larger than the desired test error → problem is too hard or underfitting → redesign your model

- If validation error is similar to training error → likely underfitting → may need to increase model complexity (e.g., loosening regularization, increasing model order and parameters)
- If training error is very low but validation error is high → likely overfitting → may need to decrease model complexity (e.g., tightening regularization, reducing model order and parameters)

- Increase the size of training data to reduce generalization gap and $\bar{E}_{new}$

Bias-Variance Decomposition

- Let z_0 be a constant, z be our estimate
- z is a random variable; it varies when we make another try

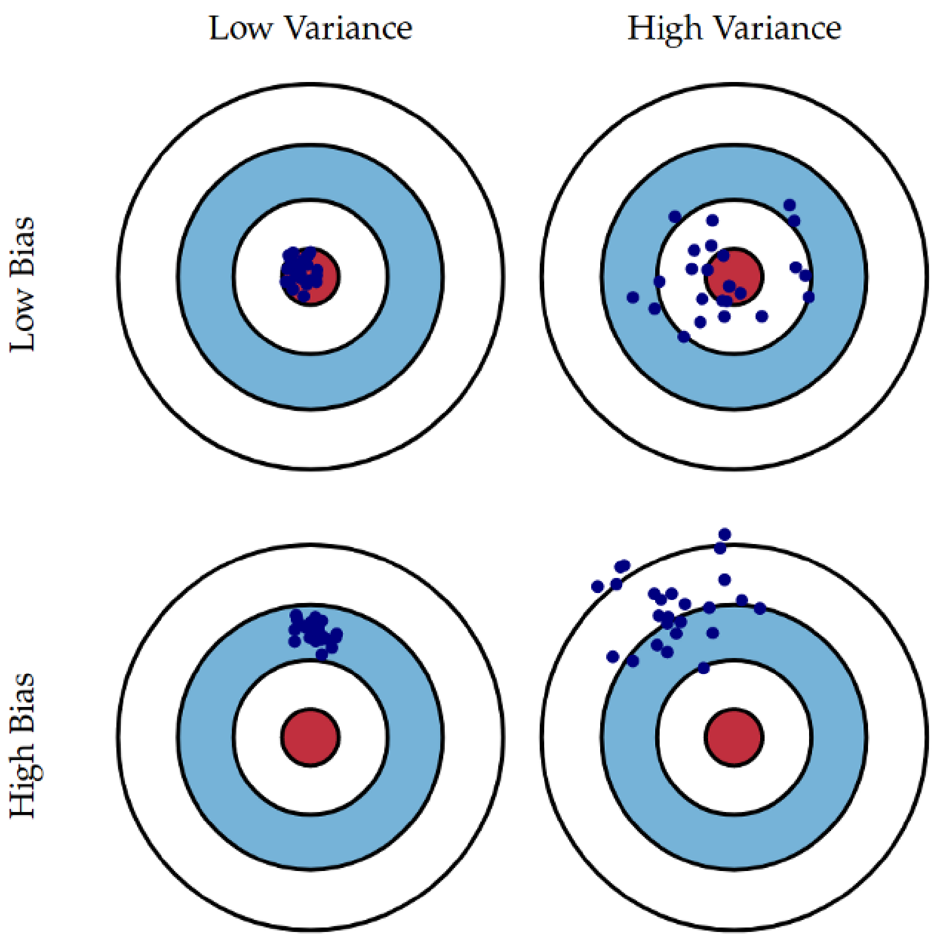
- Bias: $\mathbb{E}[z] - z_0 = \bar{z} - z_0$
- Variance: $\mathbb{E}[(z - \bar{z})^2]$

- Expected squared error

$$\mathbb{E}[(\mathrm{z} - \mathrm{z}_0)^2] = \mathbb{E}\left[\left((z - \bar{z}) + (\bar{z} - z_0)\right)^2\right]$$
$$= \mathbb{E}[(z - \bar{z})^2] + 2(\mathbb{E}[\mathrm{z}] - \bar{z})(\bar{z} - z_0) + (\bar{z} - z_0)^2$$
$$= \mathbb{E}[(z - \bar{z})^2] + (\bar{z} - z_0)^2$$

$Variance$ $\quad$ $Bias^2$

Bias vs. Variance



(Figure from http://scott.fortmann-roe.com/docs/BiasVariance.html)

Bias-Variance Decomposition of $\bar{E}_{new}$

- Let the true relation between $\boldsymbol{x}$ and y be $y = f_0(\boldsymbol{x}) + \epsilon$, where ϵ is independent noise, and $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon^2] = \sigma^2$
- Average output of models trained on different training data:

$$\bar{f}(\boldsymbol{x}) \triangleq \mathbb{E}_{\mathcal{T}}[\hat{y}(\boldsymbol{x}; \mathcal{T})]$$

- $\bar{E}_{new}$ using squared error

$$\bar{E}_{new} = \mathbb{E}_{\mathcal{T}}[E_{new}(\mathcal{T})] = \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}[(\hat{y}(\boldsymbol{x}; \mathcal{T}) - y)^2]\right]$$
$$= \mathbb{E}\left[\mathbb{E}_{\mathcal{T}}[(\hat{y}(\boldsymbol{x}; \mathcal{T}) - y)^2]\right] = \mathbb{E}\left[\mathbb{E}_{\mathcal{T}}[(\hat{y}(\boldsymbol{x}; \mathcal{T}) - f_0(\boldsymbol{x}) - \epsilon)^2]\right]$$

- Apply bias-variance decomposition, we have

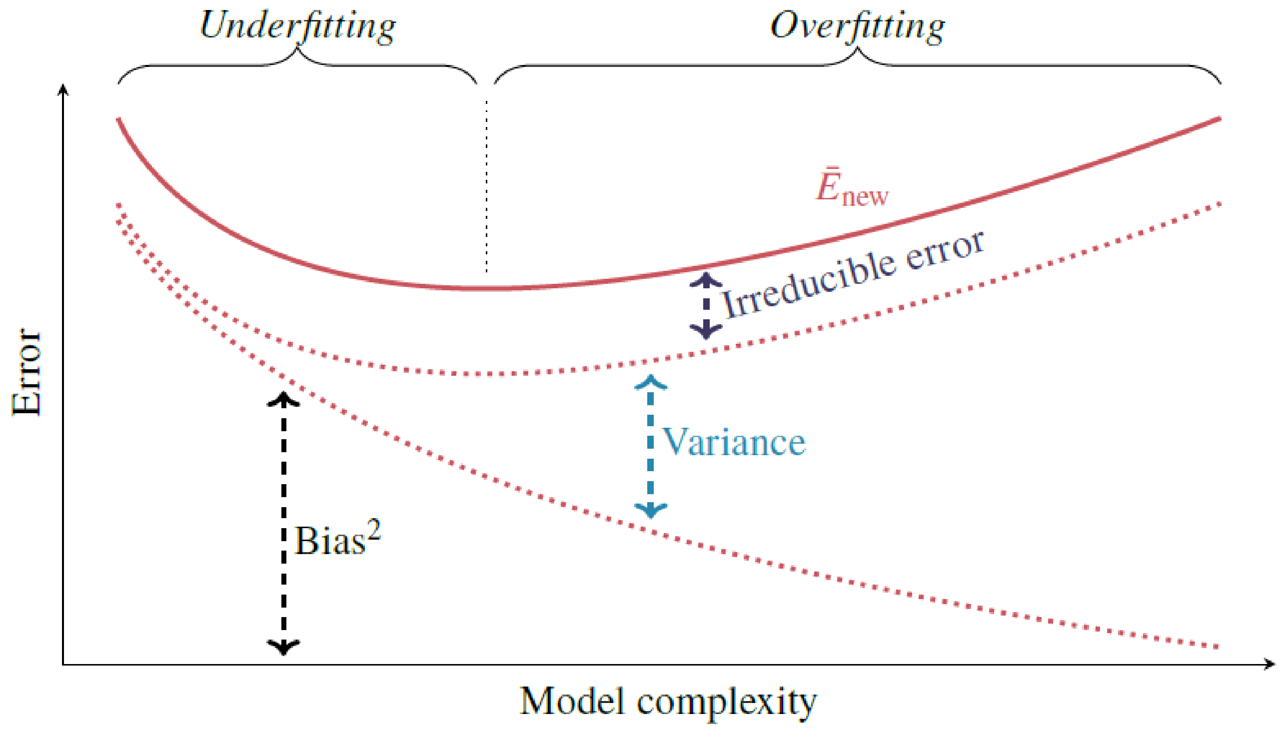
$$\mathbb{E}_{\mathcal{T}}[(\hat{y}(\boldsymbol{x}; \mathcal{T}) - f_0(\boldsymbol{x}) - \epsilon)^2] = \underbrace{\mathbb{E}_{\mathcal{T}}\left[\left(\hat{y}(\boldsymbol{x}; \mathcal{T}) - \bar{f}(\boldsymbol{x})\right)^2\right]}_{Variance} + \underbrace{\left(\bar{f}(\boldsymbol{x}) - f_0(\boldsymbol{x})\right)^2}_{Bias^2} + \underbrace{\epsilon^2}_{Irreducible\ error}$$

- Finally

$$\bar{E}_{new} = \underbrace{\mathbb{E}\left[\mathbb{E}_{\mathcal{T}}\left[\left(\hat{y}(\boldsymbol{x}; \mathcal{T}) - \bar{f}(\boldsymbol{x})\right)^2\right]\right]}_{Variance} + \underbrace{\mathbb{E}\left[\left(\bar{f}(\boldsymbol{x}) - f_0(\boldsymbol{x})\right)^2\right]}_{Bias^2} + \underbrace{\sigma^2}_{Irreducible\ error}$$

Bias-Variance Tradeoff

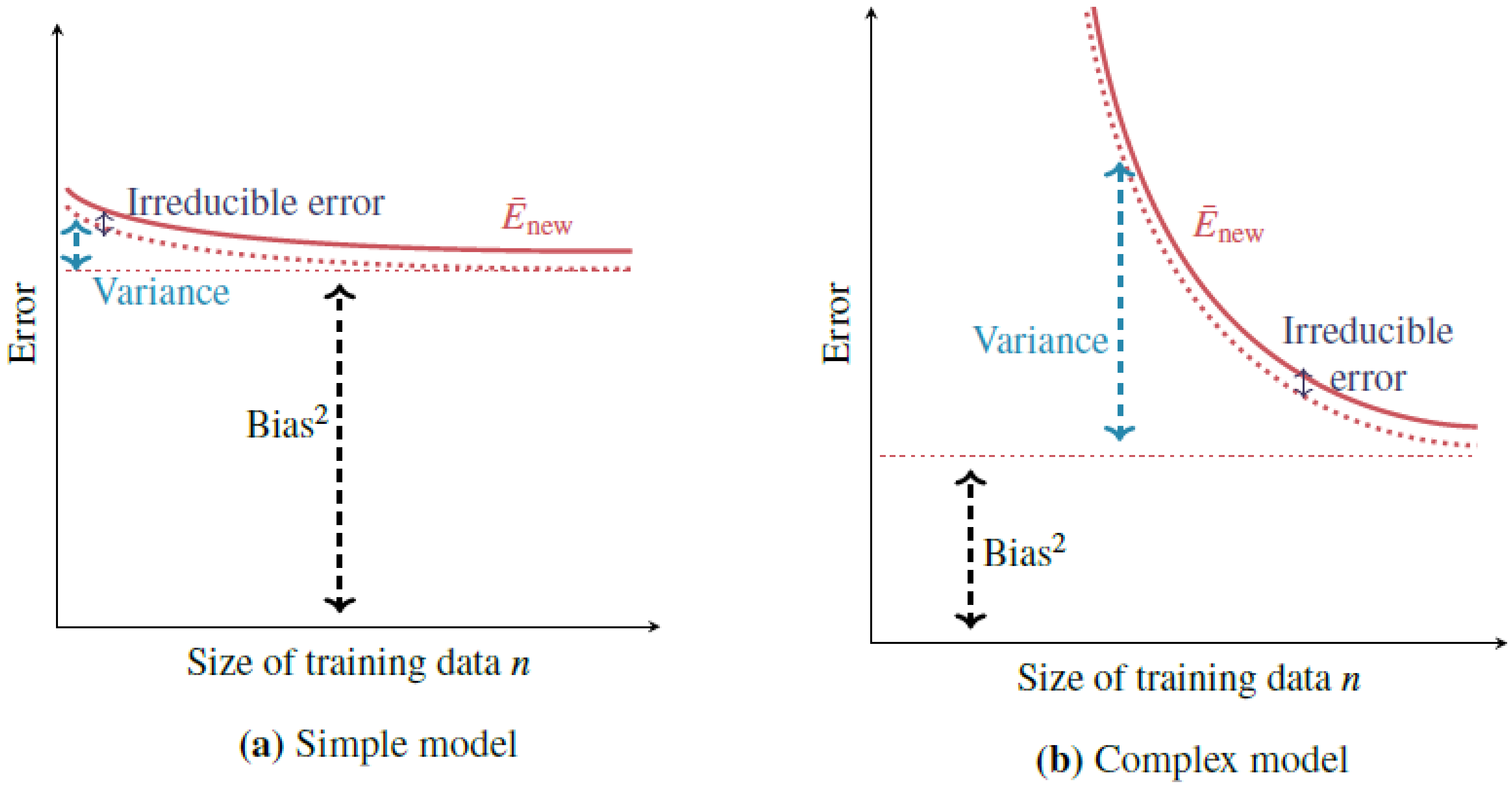
- Bias is due to the consistent error of model, averaged over all possible training sets
- Variance is due to the randomness of sampling a particular training set and randomness in the training procedure
- Higher complexity/flexibility → fits training data and its randomness better → lower bias and higher variance



(Fig. 4.8 in LWLS)

More Training Data → Lower Variance

- Especially for complex models (models with large capacity)



(Fig. 4.9 in LWLS)

Summary

- Different performance metrics (e.g., error, accuracy) for supervised models
- Metrics computed on training, validation and test sets have different use

- Error computed on hold-out validation set and through k-fold cross validation can be used to estimate model error on unseen data $\bar{E}_{new}$
  - If hyper-parameters are tuned on validation splits, then they underestimate error

- Training error $\bar{E}_{train}$ and generalization gap $\bar{E}_{new} - \bar{E}_{train}$

- Bias-variance decomposition of $\bar{E}_{new}$ with squared error
  - Bias is due to consistent error of model, average over all possible training sets
  - Variance is due to randomness of sampling a particular training set and randomness in the training procedure