Meta Learning

CS771: Introduction to Machine Learning

Purushottam Kar

Precap

We have in last several weeks covered a wide variety of ML approaches

__Problems__: Regression, (multiclass/label) classification, dim-redn, clustering

__Solutions__: linear models, prototypes, DT, kernels, NN

Today we will look at some techniques that are applicable in a largely problem and solution independent manner

Models in Machine Learning

The word “model” is often misused/overused in ML parlance

The values we learn using training data e.g. linear classifier in linear SVM, centroids in k-means, α_i in kernel SVM are often called the model

However, the above usage is not entirely correct – what we should have said in above settings is that we have learnt the model parameters *using train data*

What is the ML model then?

An ML model tells us what “kind” of ML algo we have decided to use

E.g. LwP is an ML model, linear SVM is an ML model, DT is an ML model, kNN, kernel SVM, PCA, RR, kernel RR, MLP, CNN, RNN, all are ML models

Note that when people talk about an ML model, they are not talking about the parameters being used by the model e.g. weight vectors, biases etc

Roughly, a model gives us a “broad” description of how we wish to make predictions on test data (e.g. using a tree, or using a NN, or using prototypes etc) whereas the model parameters tell us “precise” details of exactly what that predictor looks like

Models in Machine Learning

Several ML models include hyperparameters

- ***kNN***: *k (# of neighbors), metric (Euclidean, Mahalanobis)*
- ***DT***: *kind of stump being used, # children per node*
- ***Prob ML (RR)***: *choice of prior, likelihood, λ (regularization constant)*
- ***GMM and (kernel) PCA***: *k (# components)*
- ***Kernel SVM***: *kernel being used, misclassification cost C*
  - Gaussian kernel SVM itself has a hyperparameter – bandwidth γ
  - Polynomial kernel SVM itself has a hyperparameter – degree p and bias c
- ***MLP (FFNN)***: *# hidden layers/nodes, activation function*

Some people call instances of same model with different hyperparam. values as different models while others say those are the same model

- For instance, some people might say "kernel SVM" is a single model whereas others may call "Gaussian kernel SVM" and "Laplacian kernel SVM" as two separate models – a matter of convention and sometimes, friendly banter ☺

Model Selection

Let $\mathcal{M} = \{m_1, m_2, \ldots, m_k, \ldots\}$ be a set of models to choose from

Each m_i could represent a different approach (e.g. DT, SVM), or instances of the same model with different hyperparams, or both

For example, some of the m_i could be kernel SVMs, others could be NNs etc

Task: find the model (and params) that will perform the best on test

Popular considerations*: prediction performance, prediction time, model size*

$\theta_i = \text{TRAIN}(m_i, S)$ *model m_i trained on data S to get parameters θ_i*

Same model trained on different data points may give (slightly) different parameters

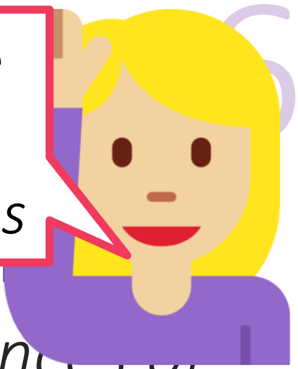
Different models (e.g. Gaussian SVM with $\gamma_1, \gamma_2, \ldots$) trained on the same dataset may give different parameters

$v_i = \text{TEST}(m_i, \theta_i, T)$ *model m_i with parameters θ_i tested on data T to get performance $v_i \in \mathbb{R}$*

v_i could denote misclassfn rate, least squares err, reconstruction error etc

Model Selection

If $\mathcal{M}$ contains variants of the same model (e.g. all are DTs) then $\mathcal{M}$ is called a *model class*



Let $\mathcal{M} = \{m_1, m_2, \ldots, m_k, \ldots\}$ be a set of models to choose fro

Each m_i could represent a different approach (e.g. DT, SVM), or instances of the same model with different hyperparams, or both

For example, some of the m_i could be kernel SVMs, others could be NNs etc

Task: find the model (and params) that will perform the best on test

Popular considerations: *prediction performance, prediction time, model size*

$\theta_i = \text{TRAIN}(m_i, S)$ *model m_i trained on data S to get parameters θ_i*

Same model trained on different data points may give (slightly) different parameters

Different models (e.g. Gaussian SVM with $\gamma_1, \gamma_2, \ldots$) trained on the same dataset may give different parameters

$v_i = \text{TEST}(m_i, \theta_i, T)$ *model m_i with parameters θ_i tested on data T to get performance $v_i \in \mathbb{R}$*

v_i could denote misclassfn rate, least squares err, reconstruction error etc

Model Selection 1: Held-out Validation

Split training set S into 2 parts S_1, S_2 randomly

Train each model on S_1, test on S_2. Choose model with best perf.

$$m^* = \arg\min_{m_i \in \mathcal{M}} \text{TEST}(m_i, \text{TRAIN}(m_i, S_1), S_2)$$

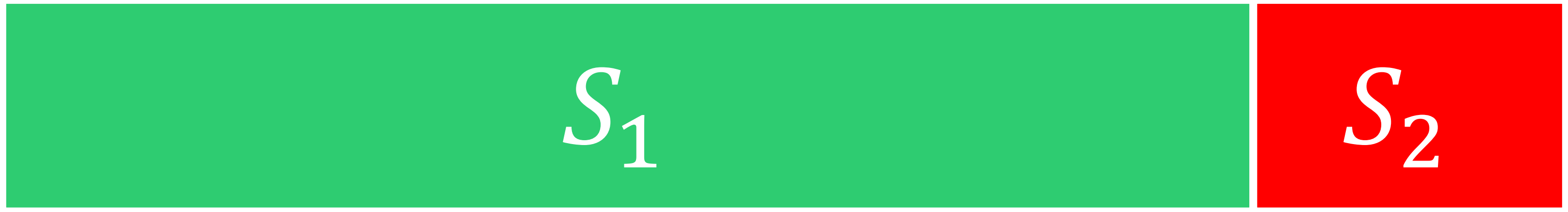
Very efficient, widely used in practice with 70-30, 80-20 splits popular

Wastes data as data points in S_2 are never used in training

Also makes us prone to risk of choosing an unfortunate split

If we are unlucky, S_2 may make the best model look worse and may instead make a suboptimal model look good

Model Selection 1: Held-out Validation



Split training set S into 2 parts S_1, S_2 randomly

Train each model on S_1, test on S_2. Choose model with best perf.

$$m^* = \arg\min_{m_i \in \mathcal{M}} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S_1), S_2)$$

Very efficient, widely used in practice with 70-30, 80-20 splits popular

Wastes data as data points in S_2 are never used in training

Also makes us prone to risk of choosing an unfortunate split

If we are unlucky, S_2 may make the best model look worse and may instead make a suboptimal model look good

Model Selection 2: k-fold Cross Validation

Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{\mathrm{j=1}}^{\mathrm{k}} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives "bad" advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation

S_1
S_2
S_3
S_4
S_5

Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

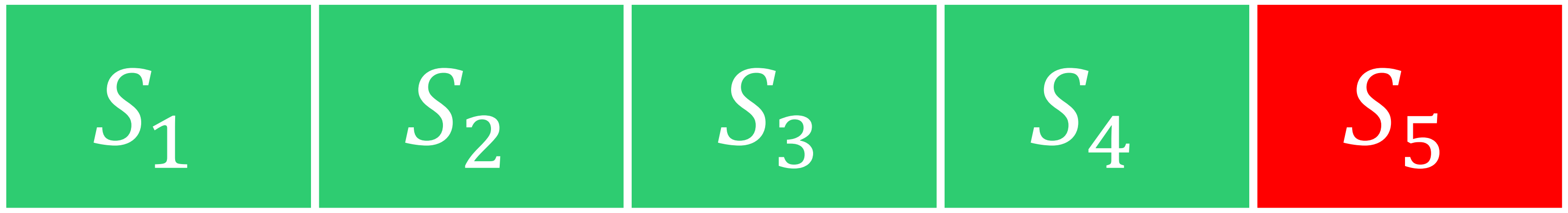
$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{\mathrm{j}=1}^{\mathrm{k}} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives "bad" advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation



Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

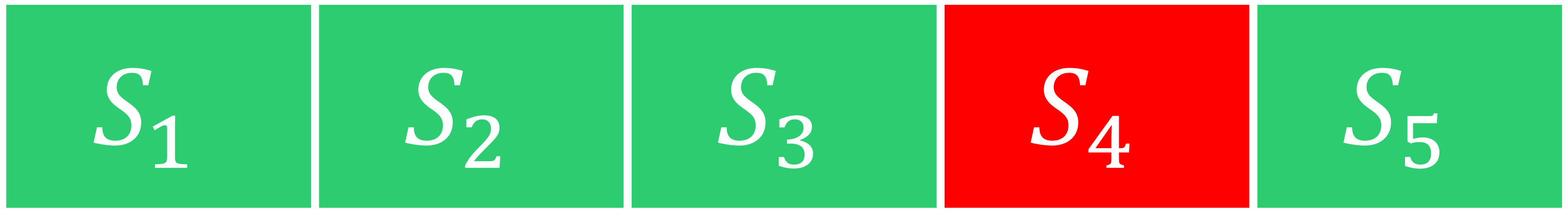
$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{\mathrm{j=1}}^{\mathrm{k}} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives "bad" advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation



Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

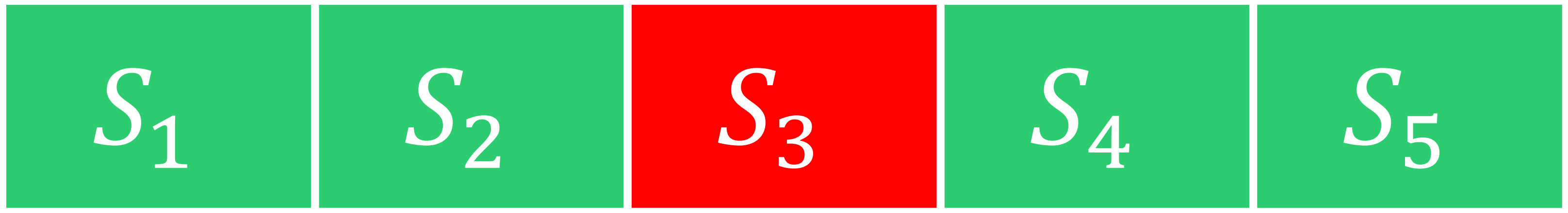
$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{\mathrm{j=1}}^{\mathrm{k}} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives “bad” advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation



Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

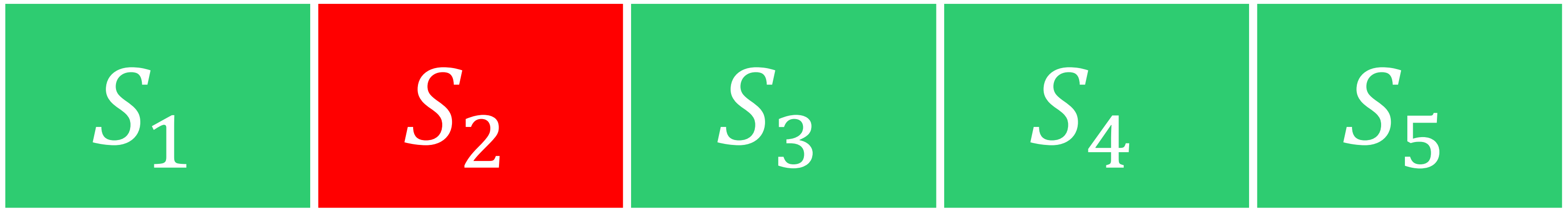
$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{j=1}^{k} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives "bad" advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation



Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

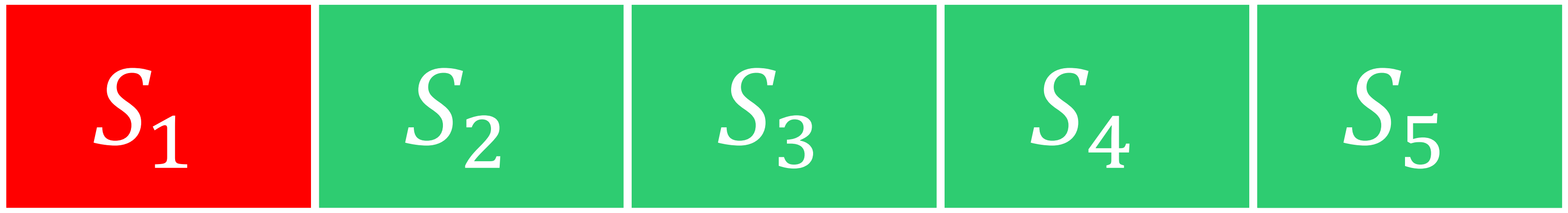
$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{\text{j=1}}^{\text{k}} \text{TEST}(m_i, \text{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives “bad” advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation



Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

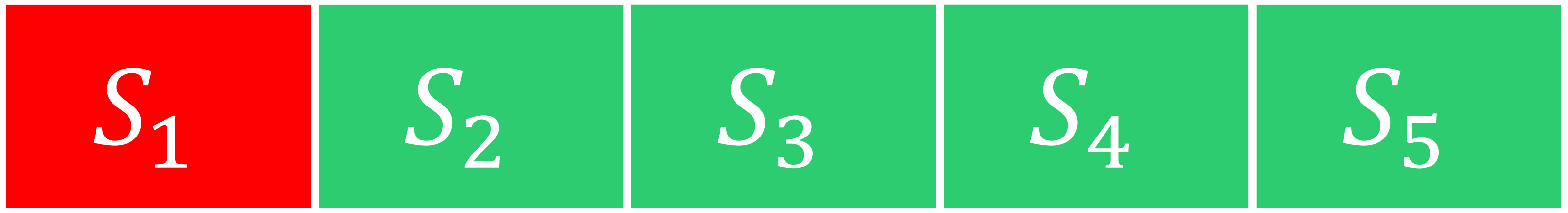
$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{j=1}^{k} \mathrm{TEST}(m_i, \mathrm{TRAIN}(m_i, S \backslash S_j), S_j)$$

More expensive but more reliable as well

Even if one part is unlucky and gives "bad" advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection 2: k-fold Cross Validation



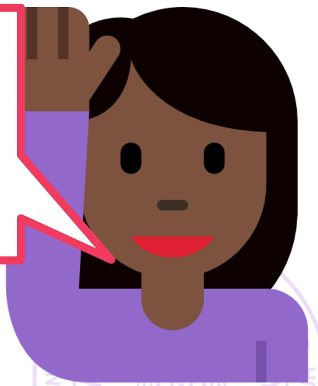
Split training set S into k parts $S_1, S_2, \ldots, S_k$ randomly ($k = 5$ popular)

Train each model on all but S_j, test on S_j. Repeat for all $j = 1, \ldots, k$

Choose model with best average performance

$$m^* = \arg\min_{m_i \in \mathcal{M}} \frac{1}{k} \sum_{j=1}^{k} \mathrm{TES}$$

LOO is popular for algorithms that require no "training" e.g. kNN since training n times super expensive!



More expensive but more reliable as well

Even if one part is unlucky and gives "bad" advice, there are other parts as well

Extreme variant LOO (leave-one-out) make every data point a part i.e. $k = |S|$

Model Selection: other techniques

Random $\boldsymbol{k}$-Fold: select k randomly chosen sets $S_1, \ldots, S_k$ of size, say $0.3n$,. Train on $S \backslash S_k$, test on S_k. Choose model with best avg. perf.

Note that folds may overlap with each other in this case

Bootstrap: select n data points randomly with replacement and use as training set. Use points never selected as a validation set

Note that the same point may repeat in the training set

Structural Risk Minimization (SRM): define a notion of complexity for each model $r(m_i)$ (e.g. # layers, clusters, magnitude of hyperparam)

Prefers models that are less "complex" (see Occam's razor if interested)

$$m^* = \arg\min_{m_i \in \mathcal{M}} \{\text{TEST}(m_i, \text{TRAIN}(m_i, S), S) + r(m_i)\}$$

Akaike/Bayesian info. criteria (AIC, BIC): designed for MAP, Bayesian methods. Similar to SRM (max likelihood instead of min test error)

Model Selection: other techniques

Bandit Optimization: useful when $\mathcal{M}$ is a model class i.e. $m \in \mathcal{M}$ given by different hyperparams. View model selection as an optim. problem

$$m^* = \arg\min_{m \in \mathcal{M}} f(\mathrm{m}) = \arg\min_{m \in \mathcal{M}} \mathrm{TEST}(m, \mathrm{TRAIN}(m, S), S)$$

However, getting "gradients" for the above objective function intractable

Hence cannot request for gradients or Hessians of f while optimizing it

Can only ask for $f(\cdot)$ values on specific models $m^1, m^2, \ldots$

Also known as zeroth-order optimization, derivative-free optimization

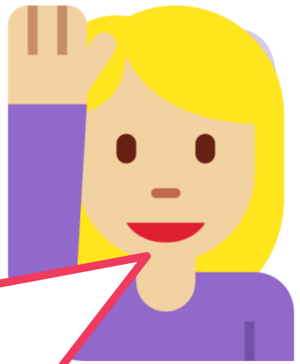
Bayesian optimization is an example of Bandit optimization

Bayesian Learning: cast model selection as a learning problem!

Establish a prior over the model class $\mathcal{M}$ and a likelihood $\mathbb{P}[S \mid m]$

Perform model learning jointly with parameter learning

Model Selection: other techniques



In fact, if tuning multiple hyperparameters (say A, B), can apply optimization tricks. Suppose A can take values $\{a_1, \dots, a_m\}$ and $B \in \{b_1, \dots, b_n\}$

Method 1: Try a few random combinations of A, B and choose the best one. Cheap but may miss best combination if unlucky not to have sampled it.

Method 2: Try all possible $m \cdot n$ combinations and see which works best – called *grid search*. Simple but can be expensive

Method 3: Try alternating optimization. Choose some a^0, say $\text{median}\{a_1, \dots, a_m\}$. Fix $A = a^0$ and find $b^0 = \text{BEST}_{j \in [n]}\left(a^0, b_j\right)$. Then fix $B = b^0$ and find a best value for A i.e. $a^1 = \text{BEST}_{i \in [m]}(a_i, b^0)$. Repeat till budget allows or convergence

oblem

$S)$

able

it

n

Bayesian optimization is an example of Bandit optimization

Bayesian Learning: cast model selection as a learning problem!

Establish a prior over the model class $\mathcal{M}$ and a likelihood $\mathbb{P}[S \mid m]$

Perform model learning jointly with parameter learning

Bias Variance Tradeoffs

Two main sources of bad test performance for ML algos

- ***Bias***: *model is too weak e.g. linear model for a very complex task*
- *Even the best trained linear model is pathetic*
- ***Variance***: *model is strong but you could not train it properly e.g. NN*
- *The best trained NN is NP-hard to learn*

Models with high variance usually are brittle as well

- *Changing training data even slightly changes the model parameters a lot*
- *Usually models that are weak are also easy to train very accurately*
- *In other words, they exhibit high bias, low variance*
- *Usually models that are strong are more difficult to train too*
- *In other words, they exhibit low-bias, high variance*

Need to balance bias and variance in practice

Bias Varia

Two main sourc

__Bias__: model is too weak e.g. linear model for a very complex task

Even the best traine

__Variance__: model is s

The best trained NN is NP-hard to learn

Models with high variance usually are brittle as well

Changing training data even slightly chan

Usually models that are weak are also eas

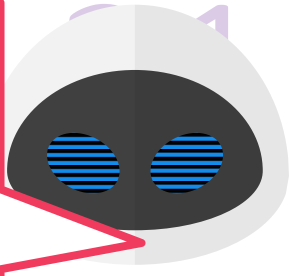
In other words, they exhibit high bias, low variance

Usually models that are strong are more difficult to train too

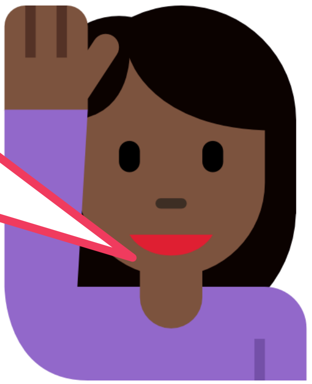
In other words, they exhibit low-bias, high variance

Need to balance bias and variance in practice

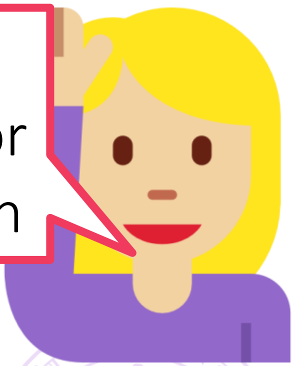
Models with low bias and low variance are golden but usually they exist only for specific domains (e.g. linear models may do very well in predicting income as a function of education). Expecting low variance and low bias in general is a pipe dream.



Models with high bias and high variance usually useless in the most spectacular way unless they offer other benefits like small model size or small prediction time



Variance of most models goes down with more training data or else more effective optimization



Bias Varian

Models with low bias and low variance are golden but usually they exist only for specific domains (e.g. linear models may do very well in predicting income as a function of education).

pipe dream.

Two main sourc

Bias: model is t

x task

Even the best t

Variance: mode

The best traine

ally useless

ffer other

iction time

Models with hig

Changing train

Usually models

In other words,

Usually models

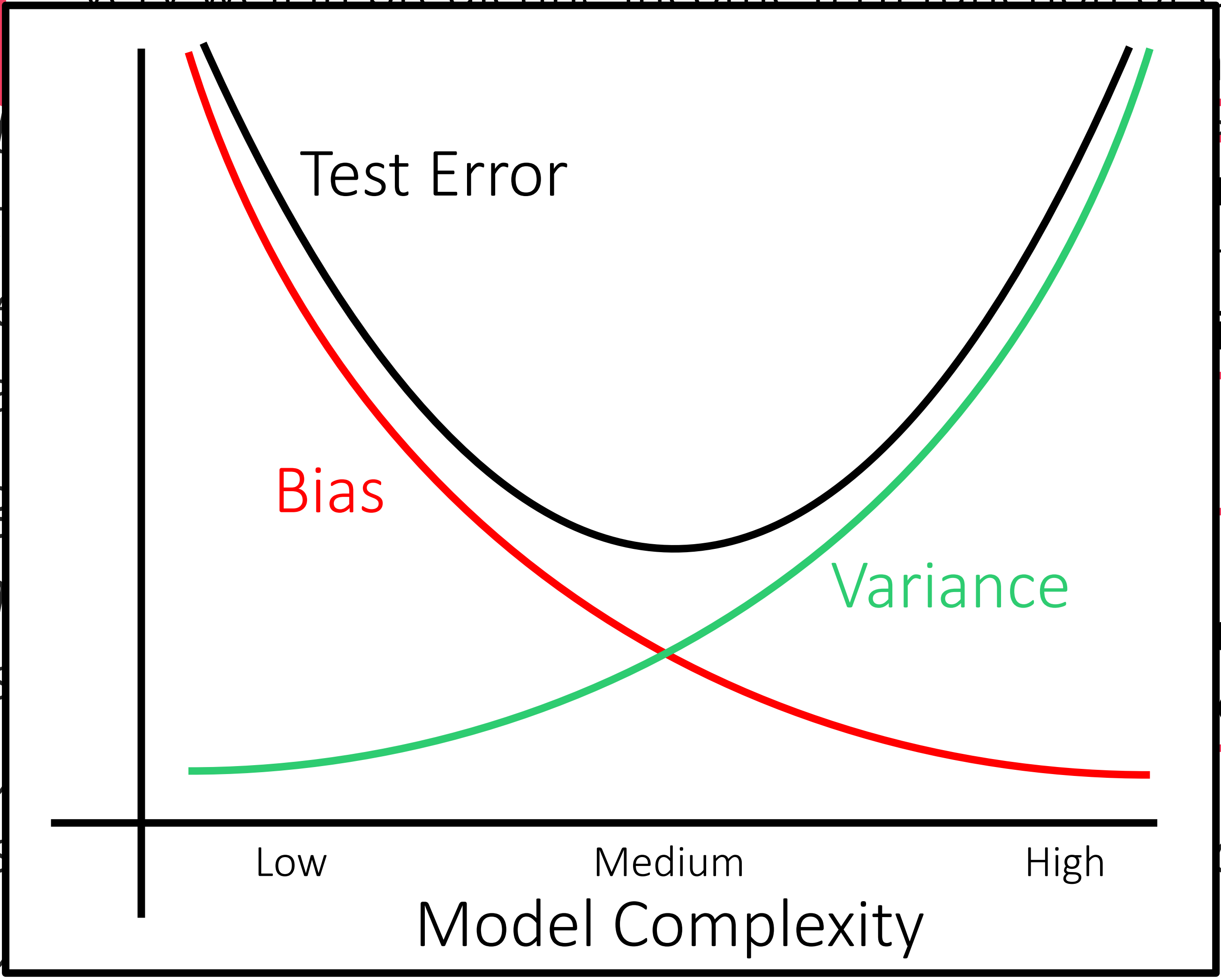
oo

In other words,

models goes

raining data or

e optimization



Need to balance bias and variance in practice

Bias Variance Mathematically

Suppose we have fixed a model m (including all its hyperparameters) and all that is left are learning the parameters of that model $\theta \in \Theta$

Suppose using n data points, we learn parameters $\theta_n \in \Theta$

Let $\mathcal{L}(\theta)$ denote the test error of any parameter $\theta \in \Theta$ and let θ^* denote the parameter with best possible test error i.e. $\mathcal{L}(\theta^*) = \min_{\theta \in \Theta} \mathcal{L}(\theta)$

Then we can write $\mathcal{L}(\theta_n) = \mathcal{L}(\theta^*) + \left(\mathcal{L}(\theta_n) - \mathcal{L}(\theta^*)\right)$ in other words

$$\mathcal{L}(\theta_n) = \left[\min_{\theta \in \Theta} \mathcal{L}(\theta)\right] + \left[\mathcal{L}(\theta_n) - \min_{\theta \in \Theta} \mathcal{L}(\theta)\right]$$

Thus, test error of our learnt model can be blamed on two factors

***Bias**: lowest error this model allowed (cant get better without changing model)*

To lower bias, change the model to make it more powerful (variance may go up)

Adding more (informative) features can also lower bias (but can also increase variance)

***Variance**: how well are we able to achieve the lowest error our model allows*

To lower variance, use more data or use a better algorithm to learn the parameters

Generalization Error

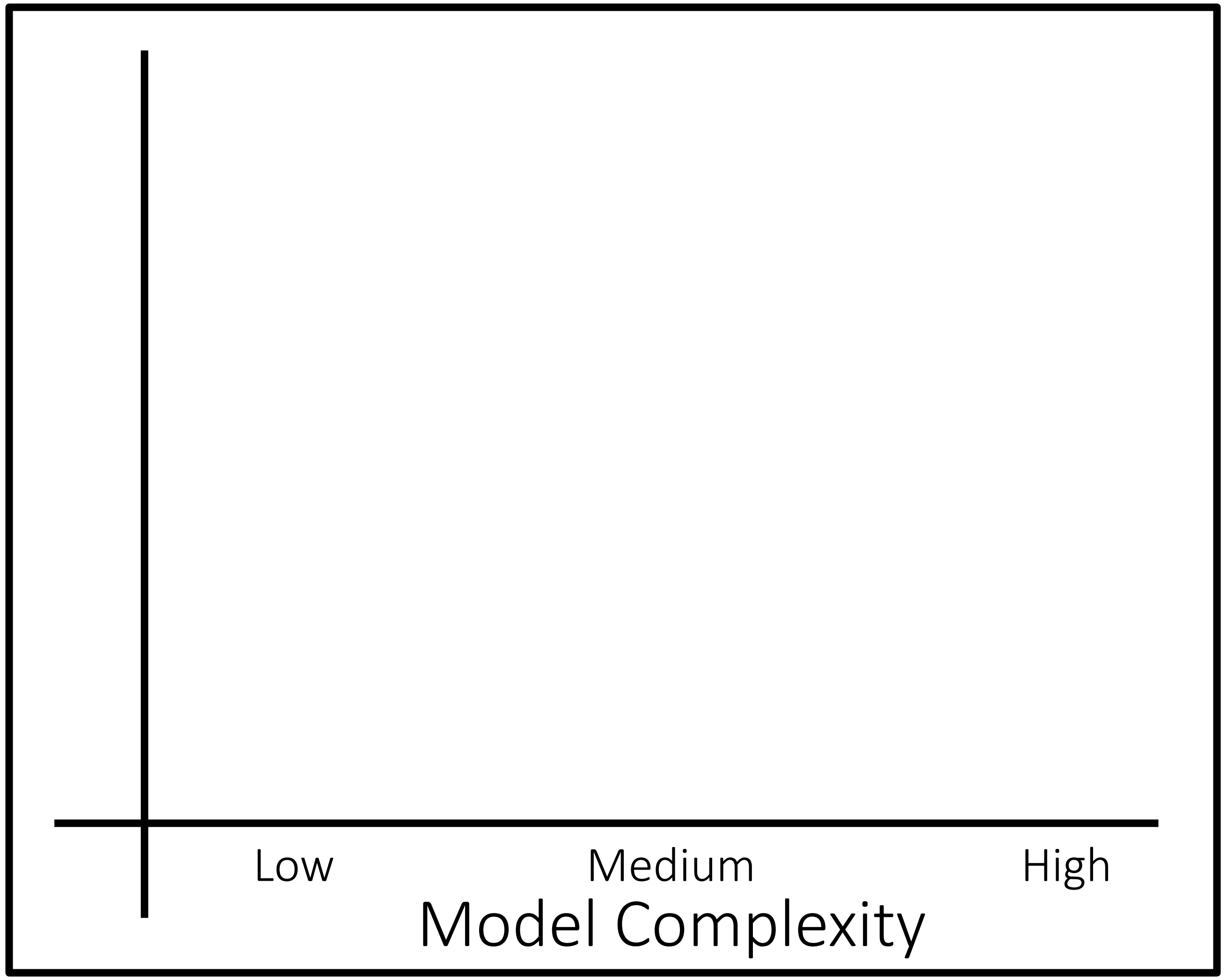
The gap between train and test error rates

- *Measures how well is the model+parameters able to "generalize" to unseen data*
- *Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)*

Note: a model with large bias may give very good gen error but high test error

- *Its test error will be close to train error but both will be very large*

Generalization Error



The gap between train and test error rates

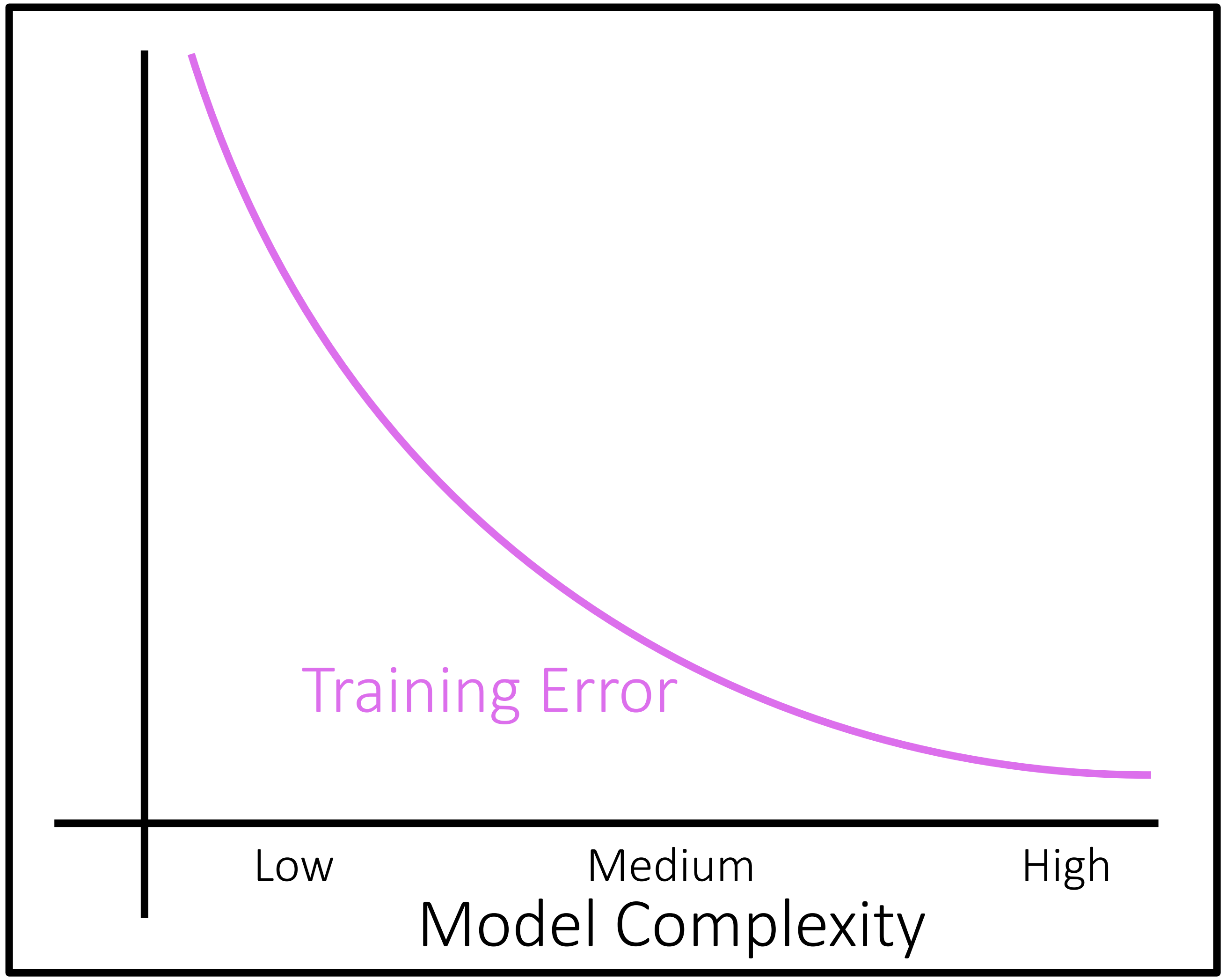
Measures how well is the model+parameters able to "generalize" to unseen data

Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Generalization Error



The gap between train and test error rates

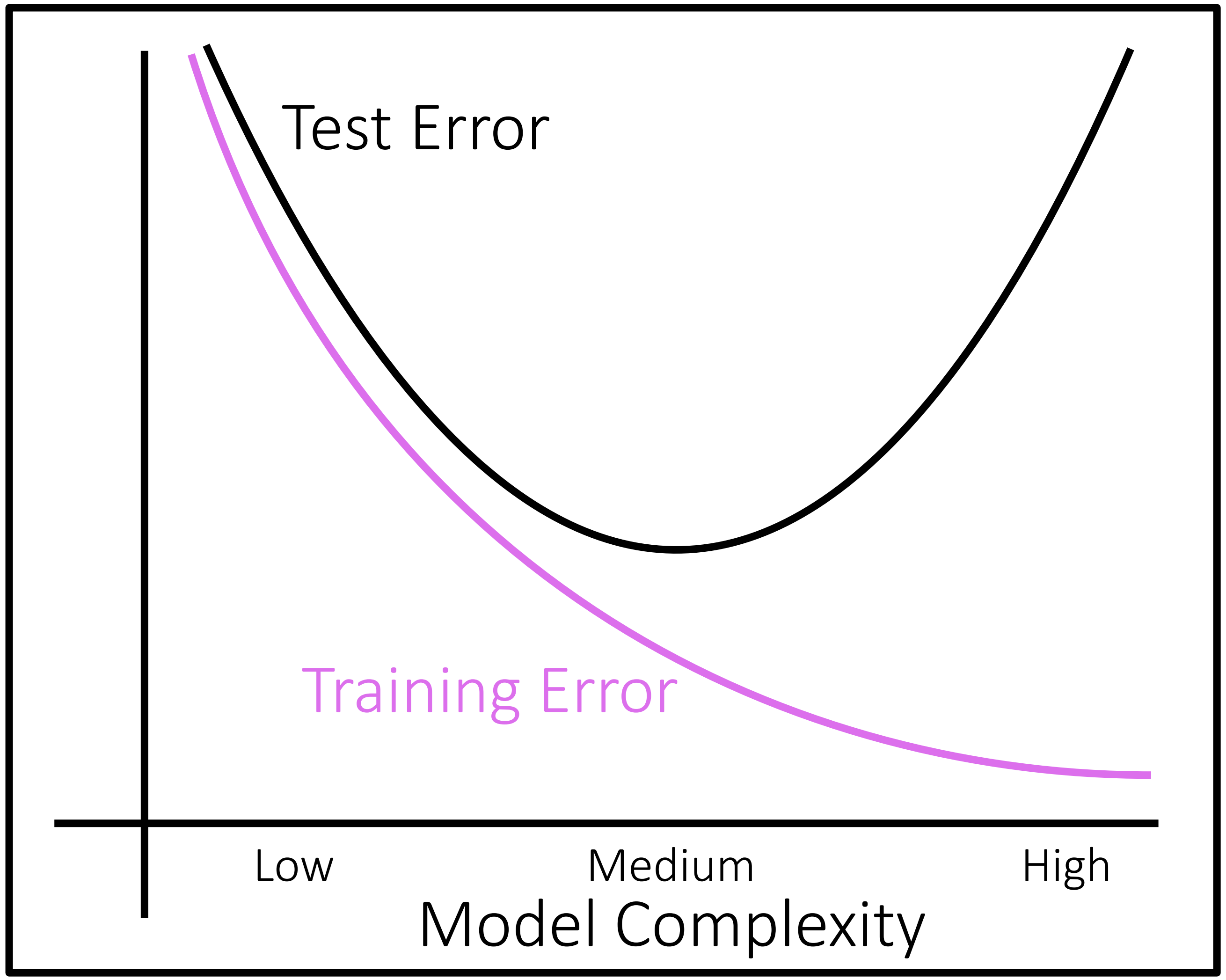
Measures how well is the model+parameters able to "generalize" to unseen data

Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Generalization Error



The gap between train and test error rates

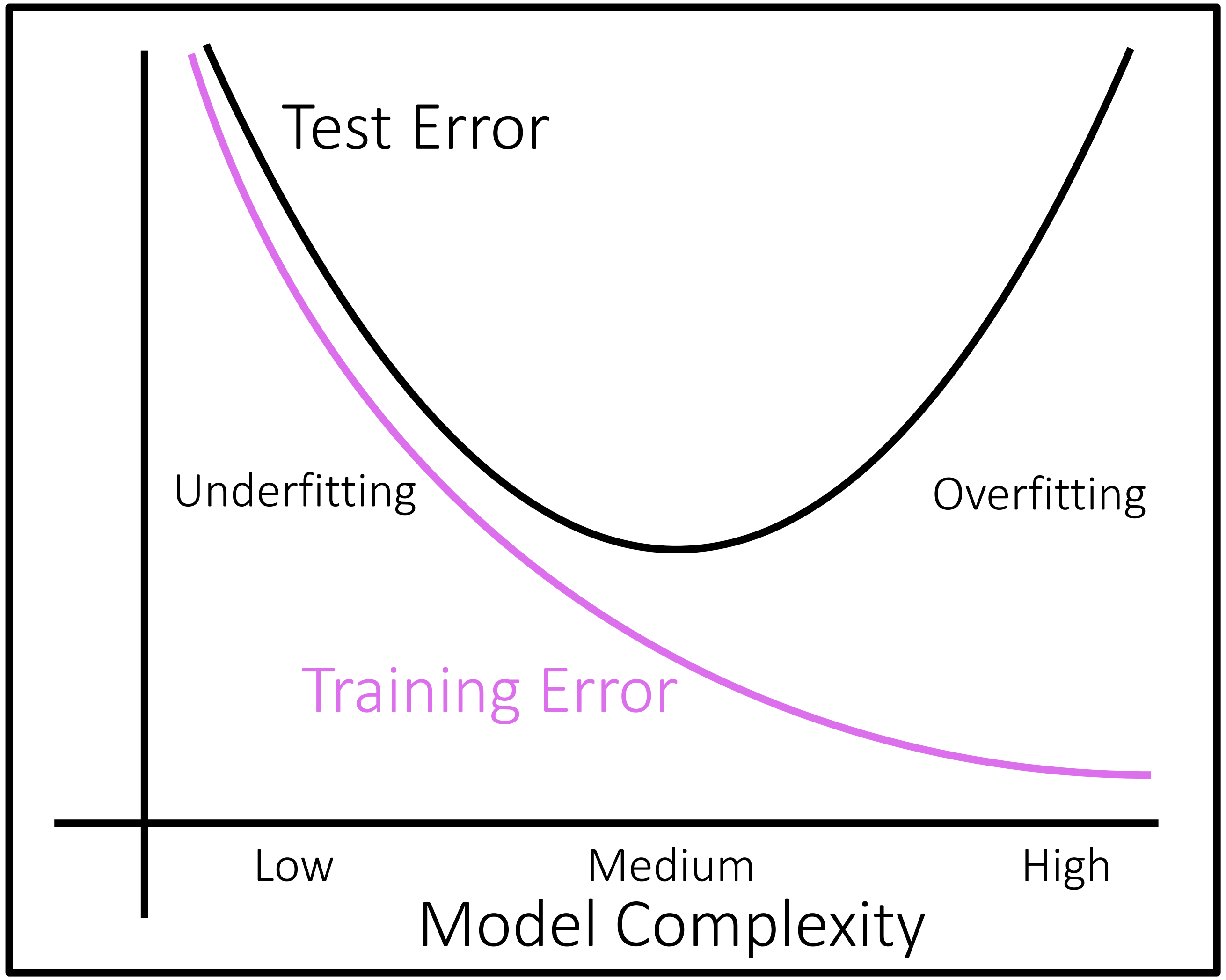
Measures how well is the model+parameters able to "generalize" to unseen data

Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Generalization Error



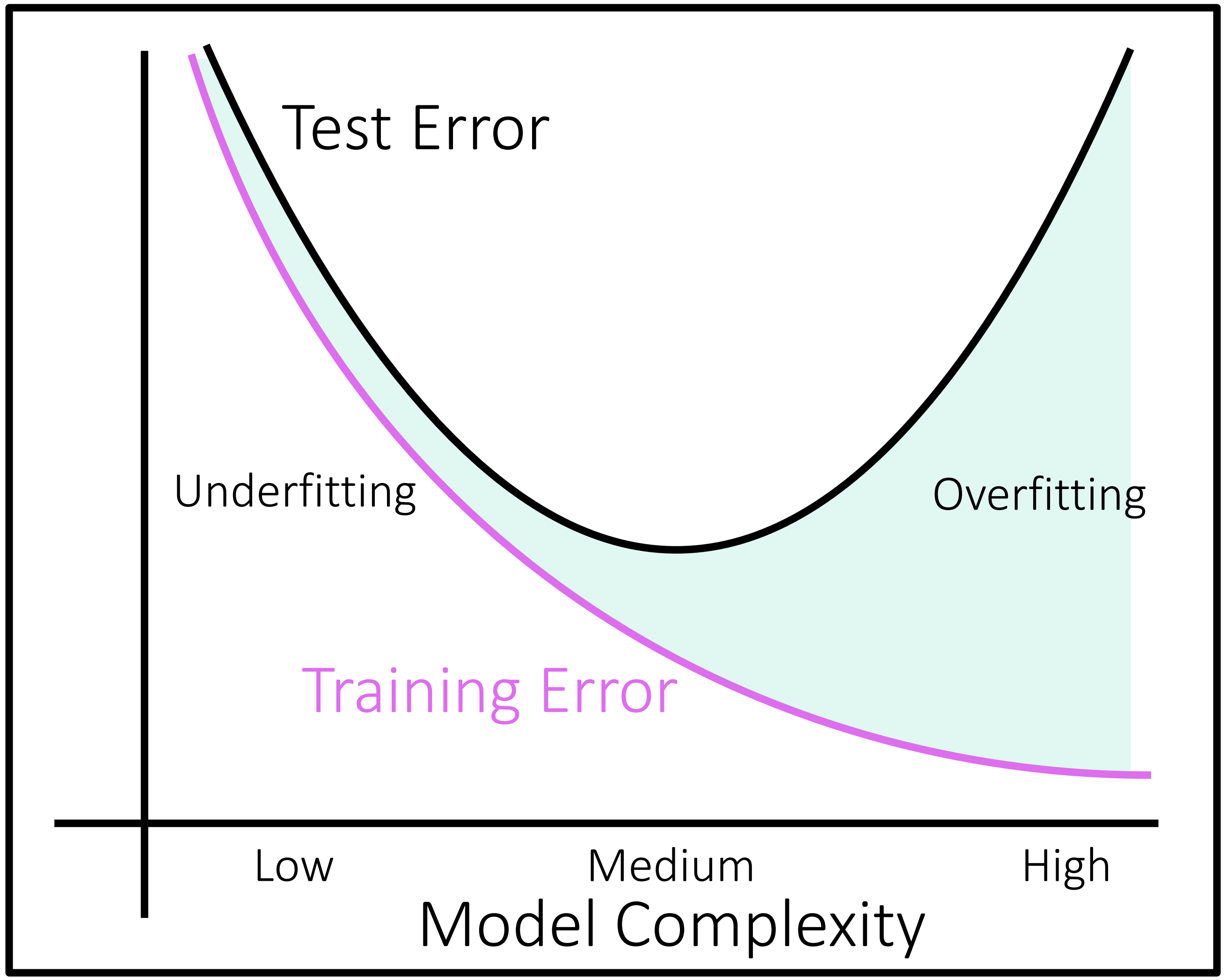
The gap between train and test error rates

- *Measures how well is the model+parameters able to "generalize" to unseen data*
- *Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)*

Note: a model with large bias may give very good gen error but high test error

- *Its test error will be close to train error but both will be very large*

Generalization Error



The gap between train and test error rates

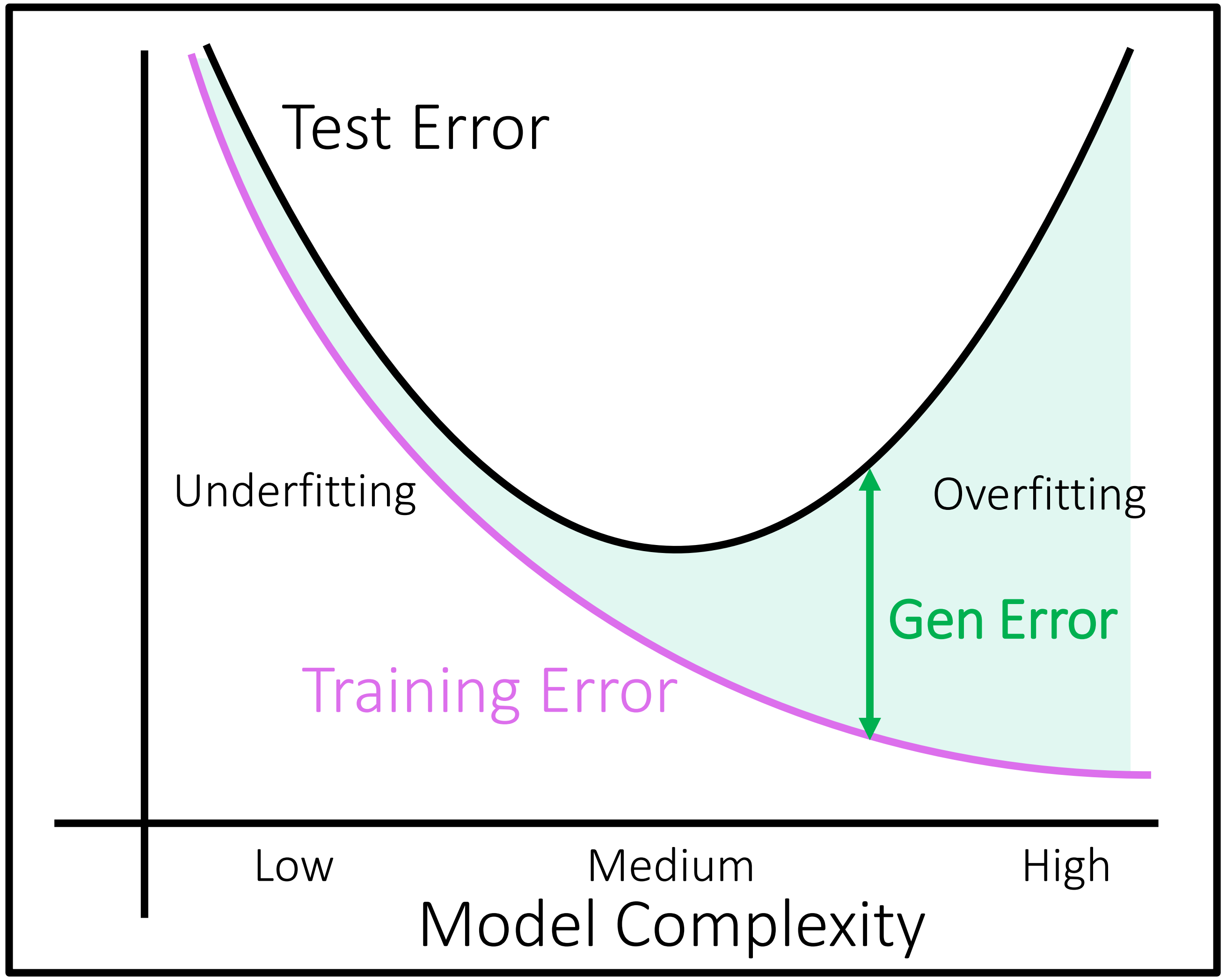
Measures how well is the model+parameters able to "generalize" to unseen data

Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Generalization Error



The gap between train and test error rates

Measures how well is the model+parameters able to "generalize" to unseen data

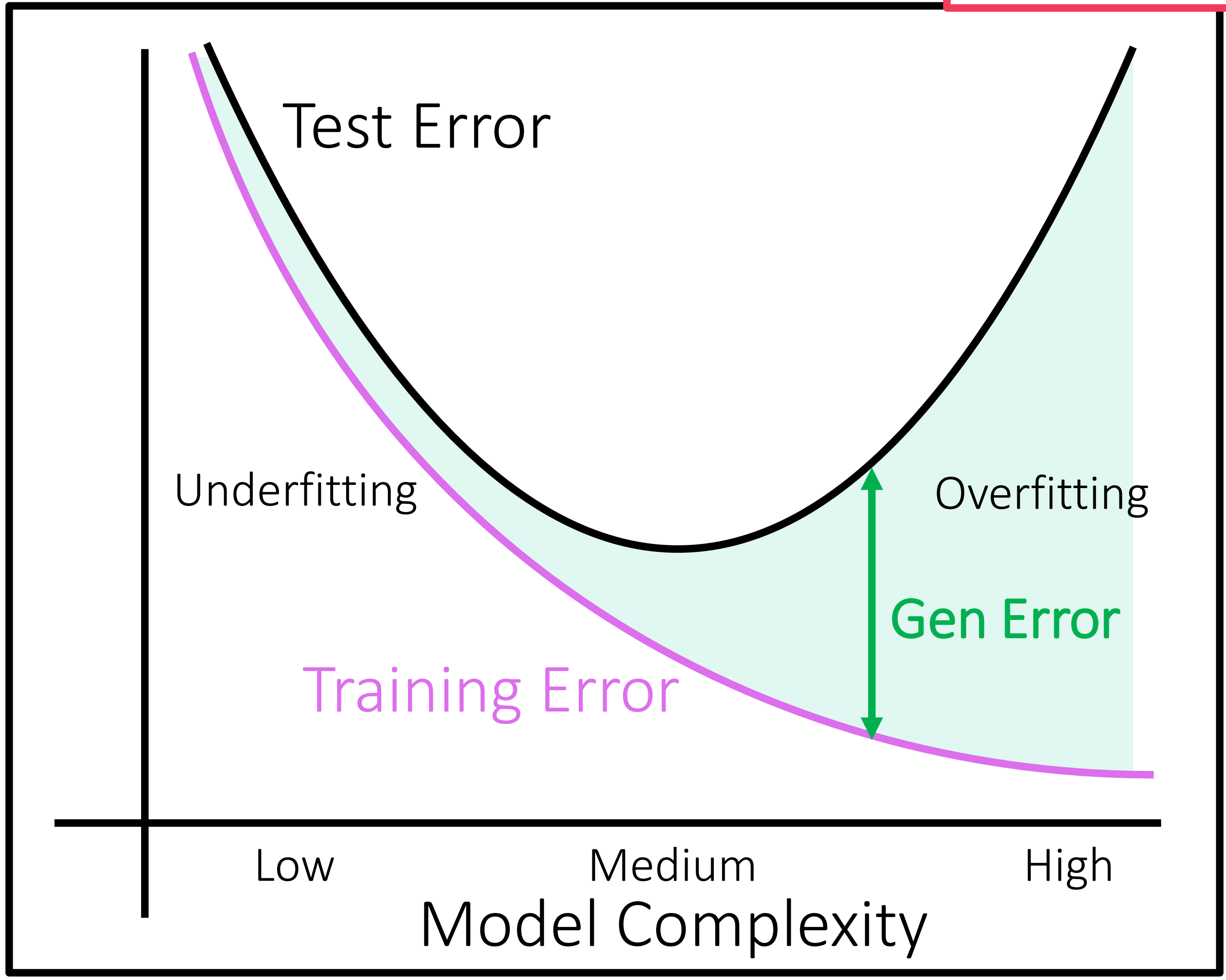
Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Generalization Err

Generalization error (just like variance) can usually be brought down by using more data points or choosing models that are simpler



The gap between train a
error rates

Measures how well is the model+parameters able to "generalize" to unseen data

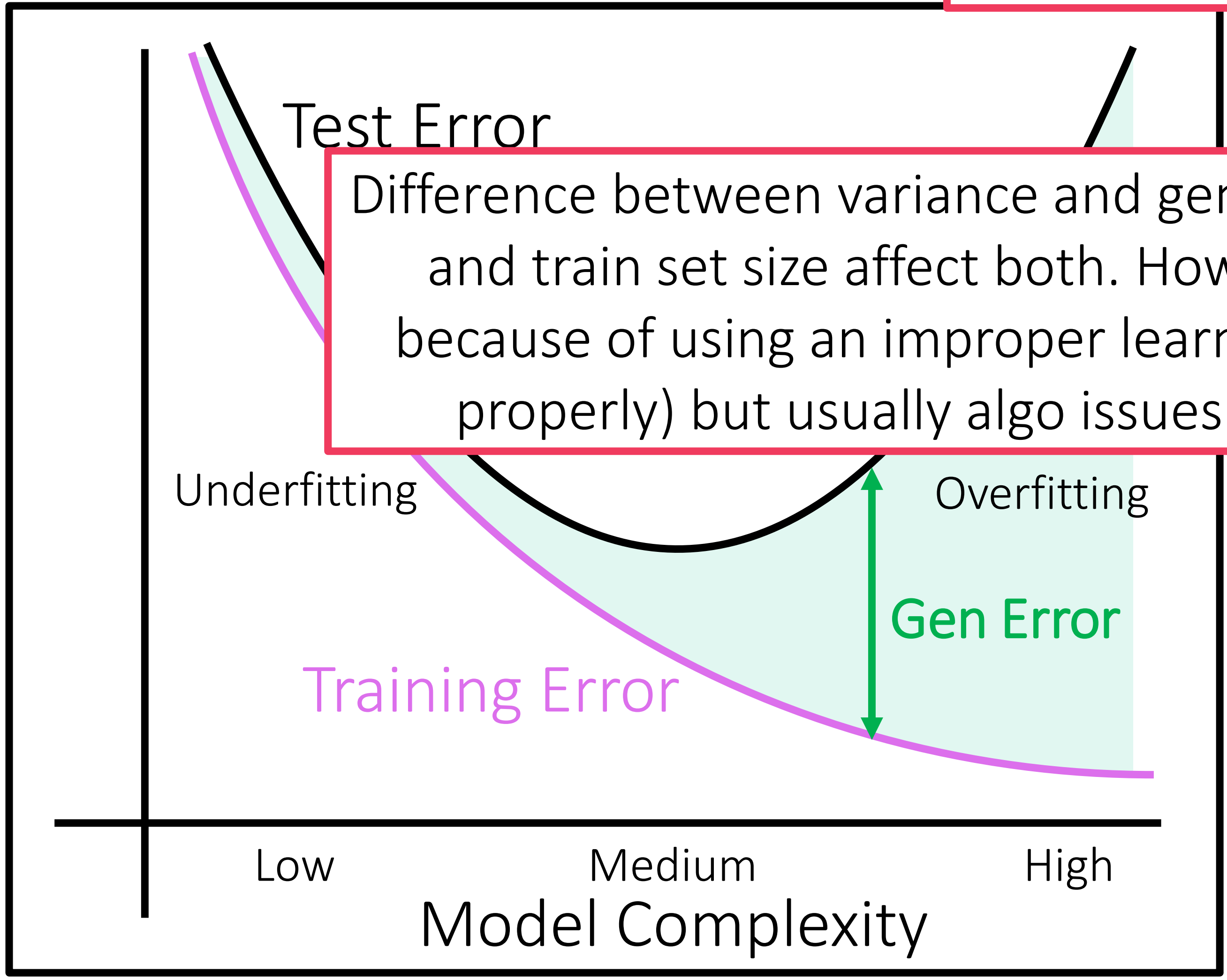
Gen error usually small for models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Generalization Err

Generalization error (just like variance) can usually be brought down by using more data points or choosing models that are simpler



The gap between train a
error rates

Difference between variance and gen error is subtle. Model complexity and train set size affect both. However, variance can also be high because of using an improper learning algorithm (or not optimizing properly) but usually algo issues do not affect gen error much.

or models with small complexity (small variance), high for models with high complexity (large variance)

Note: a model with large bias may give very good gen error but high test error

Its test error will be close to train error but both will be very large

Detecting Over/underfitting

Low training error but high test error??

You may have overfit – your model is simply memorizing training data

Your model is clearly powerful enough – does not seem to be a bias problem

Use more data/better optimizer/simpler model (or all) to decrease variance

High training error and high test error??

You may have underfit – your model is incapable of handling the learning task

Increase model class complexity, add better features, to decrease bias

Use more data, better ML algo to address any underlying variance issues

Low training error and low test error

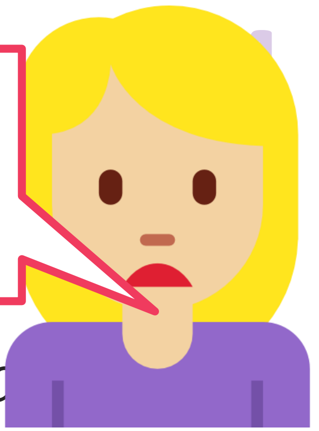
er … very good … moving on

High training error and low test error

Maybe you did early stopping which acted as a regularizer – lucky you!

Detecting Over/unde

Adding more data cannot decrease bias. The chosen model just sucks ☹ Adding more data can decrease variance though



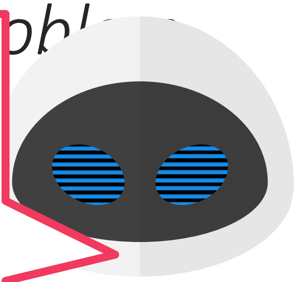
Low training error but high test

You may have overfit – your model is simply memorizing training dat

Your model is clearly power

Use more data/better optir

Sometimes may need to iterate through the above experiences (experience high bias, reduce it only to increase variance, then decrease variance etc) before reaching a sweet spot



High training error and hig

You may have underfit – your model is incapable of handling the learning task

Increase model class complexity, add better features, to decrease bias

Use more data, better ML algo to address any underlying variance issues

Low training error and low test error

er … very good … moving on

High training error and low test error

Maybe you did early stopping which acted as a regularizer – lucky you!

Ensemble ML Algorithms

Most real life systems that use ML use not one but several models

- *Known to be true of industrial models for recommendation, search, ranking*
- ***Ensemble**: a collection of several ML models working cooperatively*

Ensembles have several advantages

- *Reduce reliance on a single model which may fail at times*
- *Allow us to harness the strengths of a variety of models*
- *Offers users a smooth transition if ensemble needs modification*
  - E.g. if an outdated algo is removed from ensemble or a latest algo is added
  - If a single model had been used, changing that model could disrupt user experience
- *Can also be used to address bias-variance issues*
  - Some ensemble techniques can lower bias of weak models (make them more powerful)
  - Other techniques can lower variance of models (make them stable and less jittery)

Voting Ensemble

One of the simplest ensemble techniques – aka “learning with experts”

- *Works even when training is not in our hands or if models not from a single* $\mathcal{M}$
- *Suppose we have 5 sources to answer “Will it rain tomorrow?”*

Voting Ensemble

One of the simplest ensemble techniques – aka "learning with experts"

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer "Will it rain tomorrow?"

Gold Truth

Y

N

Y

Y

Y

N

Voting Ensemble

One of the simplest ensemble techniques – aka "learning with experts"

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer "Will it rain tomorrow?"

CNN BBC NBC NEWS दूरदर्शन ALJAZEERA

Gold Truth

Y
N
Y
Y
Y
N

Voting Ensemble

One of the simplest ensemble techniques – aka "learning with experts"

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer "Will it rain tomorrow?"

P

Correct prediction

Q

Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
					Y
					N
					Y
					Y
					Y
					N

Voting Ensemble

One of the simplest ensemble techniques – aka “learning with experts”

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensemble

One of the simplest ensemble techniques – aka “learning with experts”

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensemble

One of the simplest ensemble techniques – aka "learning with experts"

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer "Will it rain tomorrow?"

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensemble

One of the simplest ensemble techniques – aka “learning with experts”

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensemble

One of the simplest ensemble techniques – aka "learning with experts"

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer "Will it rain tomorrow?"

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensemble

One of the simplest ensemble techniques – aka “learning with experts”

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensemble

One of the simplest ensemble techniques – aka “learning with experts”

Works even when training is not in our hands or if models not from a single $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

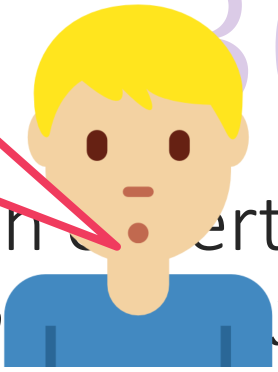
P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions



One of the simplest ensemble techniques – aka “learning with erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

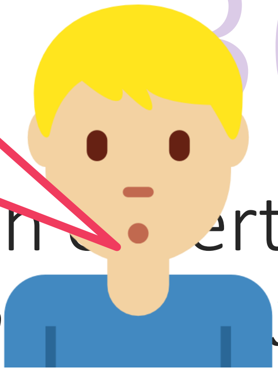
P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions



One of t erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

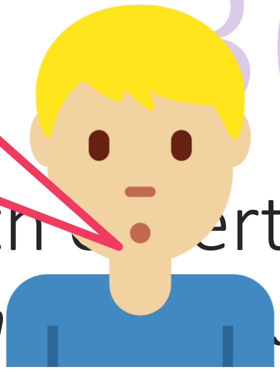
P
Correct prediction

Q
Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions



One of t erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions

One of t … erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

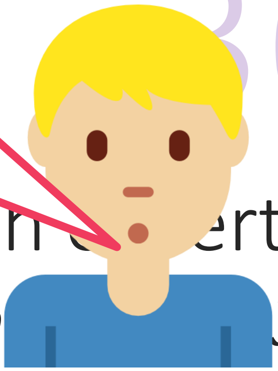
P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions



One of the simplest ensemble techniques – aka “learning with experts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

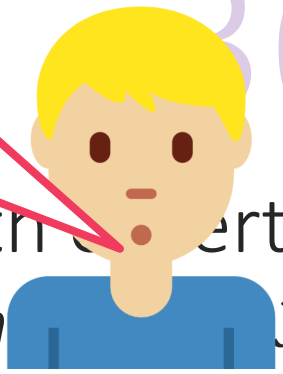
P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions



One of t erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

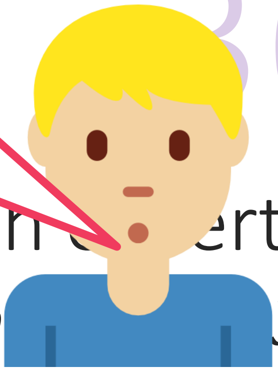
P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions



One of t [illegible] erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

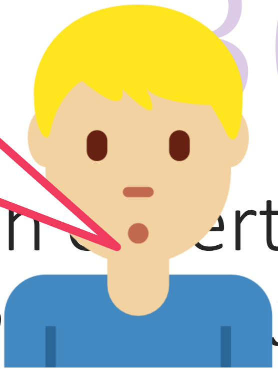
P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Votin

No individual news network gets more than 66% correct predictions but if we take a majority vote, we are 100% correct all the time. The same trick is also popularly used in psephology (“poll of polls”)



One of t erts”

Works even when training is not in our hands or if models not from $\mathcal{M}$

Suppose we have 5 sources to answer “Will it rain tomorrow?”

P Correct prediction

Q Incorrect prediction

CNN	BBC	NBC NEWS	दूरदर्शन	ALJAZEERA	Gold Truth
N	Y	N	Y	Y	Y
N	Y	N	N	Y	N
Y	Y	N	Y	Y	Y
N	Y	Y	N	Y	Y
Y	N	Y	Y	N	Y
Y	N	N	Y	N	N

Voting Ensembles

Receive K pre-trained classifiers $f_1, f_2, \ldots, f_K$ s.t. $f_i : \mathcal{X} \to \{-1, +1\}$

Construct a new classifier $\hat{f}_{\text{MAJ}}$ such that for any $x \in \mathcal{X}$

$$\hat{f}_{\text{MAJ}}(x) = \text{sign}\left(\sum_{k=1}^{K} f_k(x)\right)$$

Hope that mistakes of one classifier will be corrected by others

Stacking: interpret $[f_1(x), f_2(x), \ldots, f_K(x)]$ as a K-dimensional vector and learn a new classifier over these new "features"

This is not expected to do well in general. If the classifiers were not trained properly, they may synchronize their mistakes

Possible reason why "polls-of-polls" fail spectacularly – most polls are in unison

Fixing these issues leads to useful techniques called bagging and boosting