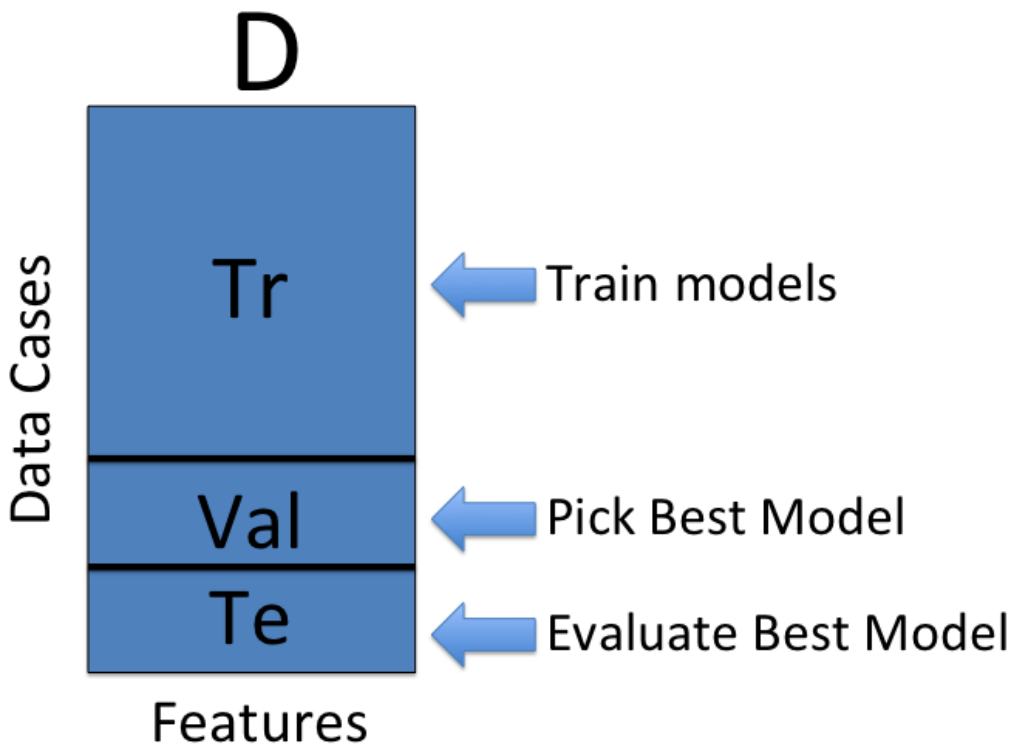
Model Selection and Evaluation

- We've identified the parameters that control the capacity of our models, we need a way to choose optimal values for these parameters.
- In addition, we will want an estimate of the generalization error that the selected parameters achieve.
- To obtain, valid results, we need to use appropriate methodology and construct learning experiments carefully.
- **Guiding Principle:** Data used to estimate generalization error can not be used for any other purpose (ie: model training, hyperparameter selection, feature selection, etc.) or the results of the evaluation will be **biased**.

Recipe 1: Train-Validation-Test

- Given a data set D, we randomly partition the data cases into a training set (Tr), a validation set (V), and a test set (Te). Typical splits are 60/20/20, 80/10/10, etc.
- Models M_i are learned on Tr for each choice of hyperparameters H_i
- The validation error Val_i of each model M_i is evaluated on V.
- The hyperparameters H_* with the lowest value of Val_i are selected and the classifier is re-trained using these hyperparameters on $Tr + V$, yielding a final model M_*
- Generalization performance is estimated by evaluating error/accuracy of M_* on the test data Te.

Example: Train-Validation-Test



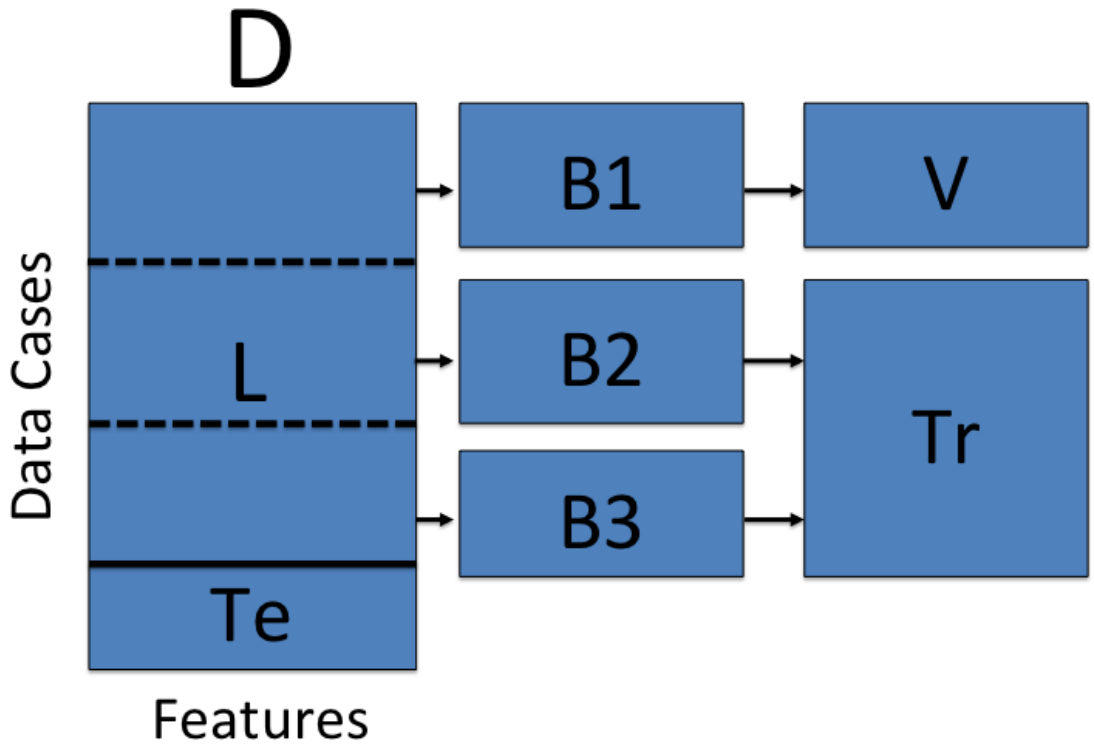
Note that the order of the data cases needs to be randomly shuffled before partitioning D.

Recipe 2: Crossvalidation-Test

- Randomly partition D into a learning set L and a test set Te (typically 50/50, 80/20, etc).
- We next randomly partition L into a set of K blocks $B_1, ..., B_K$.
- For each crossvalidation fold $k = 1, ..., K$:
  - Let $V = B_k$ and $Tr = L/B_k$ (the remaining $K-1$ blocks).
  - Learn M_{ik} on Tr for each choice of hyperparameters H_i.
  - Compute Val_{ik} of M_{ik} on V.
- Select hyperparameters H_* minimizing $\frac{1}{K}\sum_{k=1}^{K} Val_{ik}$ and re-train model on L using these hyperparameters, yielding final model M_*.
- Estimate generalization performance by evaluating error/accuracy of M_* on Te.

Example: 3-Fold Cross Validation and Test

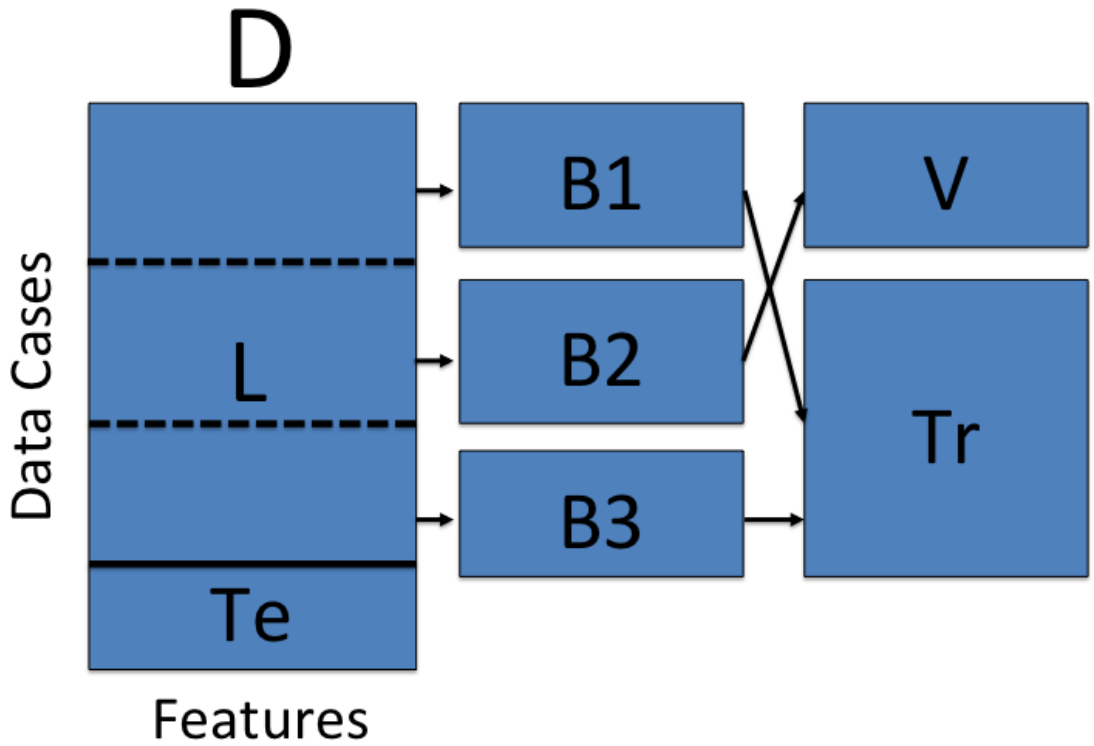
First Cross Validation Fold



Note that the order of the data cases needs to be randomly shuffled before partitioning D into L and Te.

Example: 3-Fold Cross Validation and Test

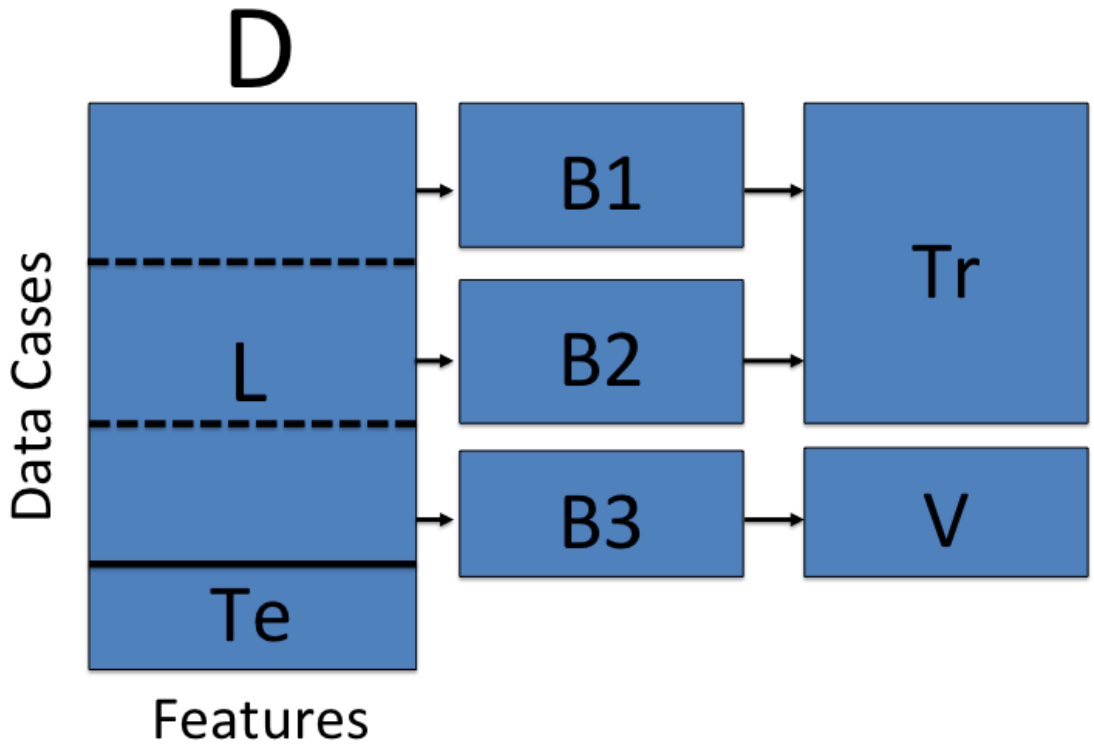
Second Cross Validation Fold



Note that the order of the data cases needs to be randomly shuffled before partitioning D into L and Te.

Example: 3-Fold Cross Validation and Test

Third Cross Validation Fold



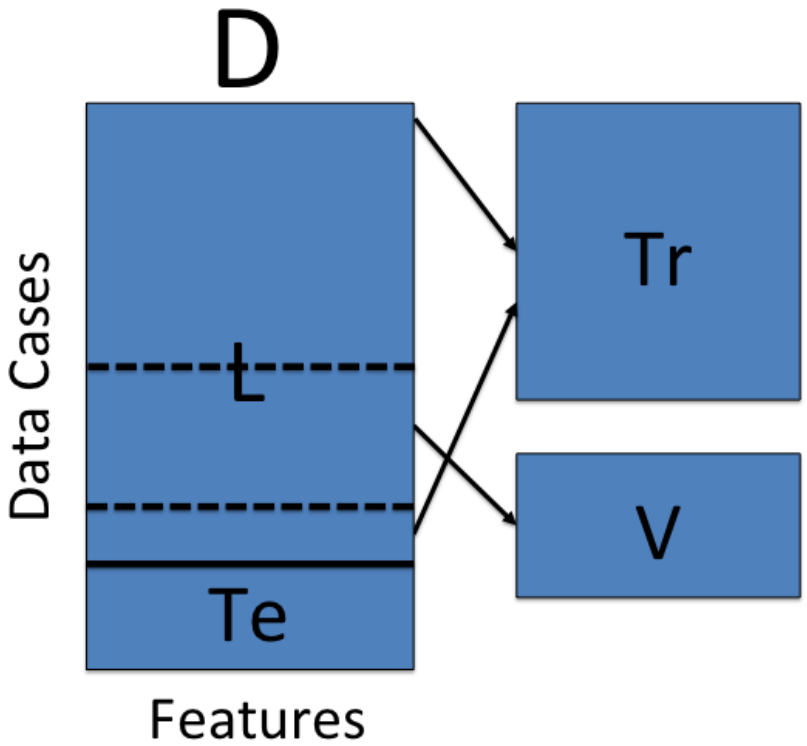
Note that the order of the data cases needs to be randomly shuffled before partitioning D into L and Te.

Recipe 3: Random Resampling Validation-Test

- Randomly partition the data cases into a learning set L and a test set Te (typically 50/50, 80/20, etc).
- For sample $s = 1, \ldots, S$:
  - Randomly partition L into Tr and V (again 50/50, 80/20, etc).
  - Learn M_{is} on Tr for each choice of hyperparameters H_i.
  - Compute Val_{is} of M_{is} on V.
- Select hyperparameters H_* minimizing $\frac{1}{S}\sum_{s=1}^{S} Val_{is}$ and re-train model on L using these hyperparameters, yielding final model M_*.
- Estimate generalization performance by evaluating error/accuracy of M_* on Te.

Example: 3-Sample Random Resampling and Test

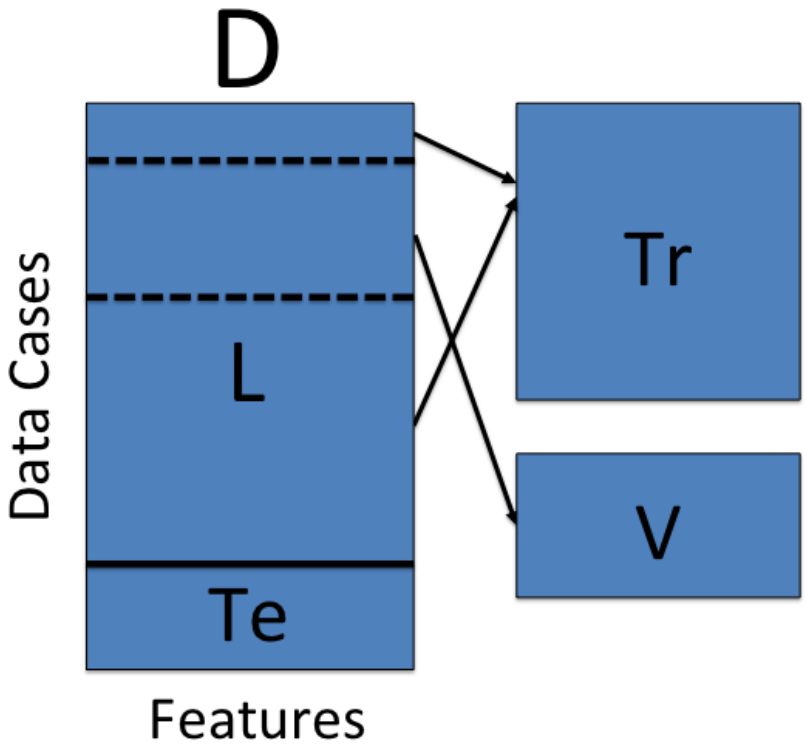
First Sample



Note that the order of the data cases needs to be randomly shuffled before partitioning D into L and Te.

Example: 3-Sample Random Resampling and Test

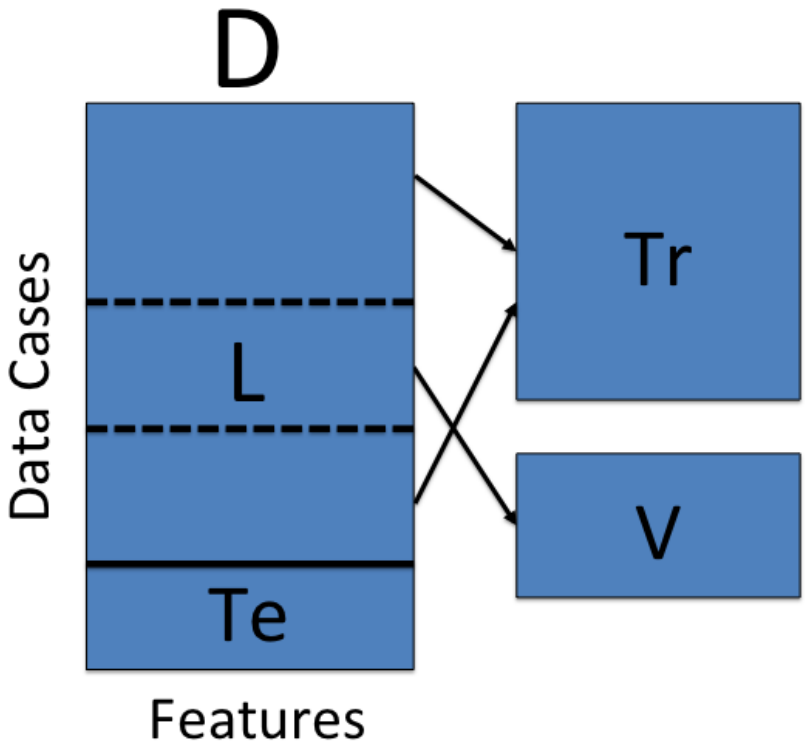
Second Sample



Note that the order of the data cases needs to be randomly shuffled before partitioning D into L and Te.

Example: 3-Sample Random Resampling and Test

Third Sample



Note that the order of the data cases needs to be randomly shuffled before partitioning D into L and Te.

Recipe 4: Crossvalidation-Crossvalidation

- Randomly partition data set D into a set of J blocks $C_1, ..., C_J$.
- For $j = 1, ..., J$:
  - Let $Te_j = C_j$ and $L_j = D/C_j$
  - Partition L_j into a set of K blocks $B_1, ..., B_K$.
  - For $k = 1, ..., K$:
    - Let $V = B_k$ and $Tr = L_j/B_k$.
    - Learn M_{ik} on Tr for each choice of hyperparameters H_i.
    - Compute error Val_{ik} of M_{ik} on V.
  - Select hyperparameters H_* minimizing $\frac{1}{K}\sum_{k=1}^{K} Val_{ik}$ and re-train model on L_j using these hyperparameters, yielding model M_{*j}.
  - Compute Err_j by evaluating M_{*j} on Te_j.
- Estimate generalization error using $\frac{1}{J}\sum_{j=1}^{J} Err_j$
- We can define a similar nested random resampling validation procedure.

Trade-Offs

- In cases where the data has a benchmark split into a training set and a test set, we can use Recipes 1-3 by preserving the given test set and splitting the given training set into train and validation sets as needed.
- In cases where there is relatively little data, using a single held out test set will have high bias. In these cases, Recipe 4 often provides a better estimate of generalization error, but has much higher computational cost.
- Choosing larger K in cross validation will reduce bias. Choosing larger S in random re-sampling validation will reduce variance and bias. However, both increase computational costs. $K = 3, 5, 10$ are common choices for cross validation. $K = N$, also known as Leave-one-out cross validation is also popular when feasible.