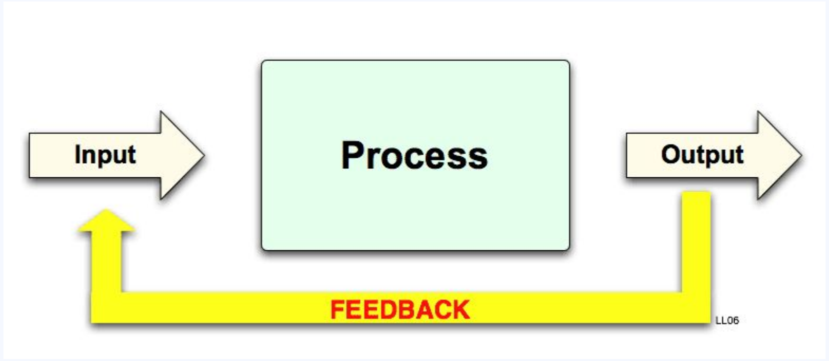
Basic Machine Learning

02

Machine Learning

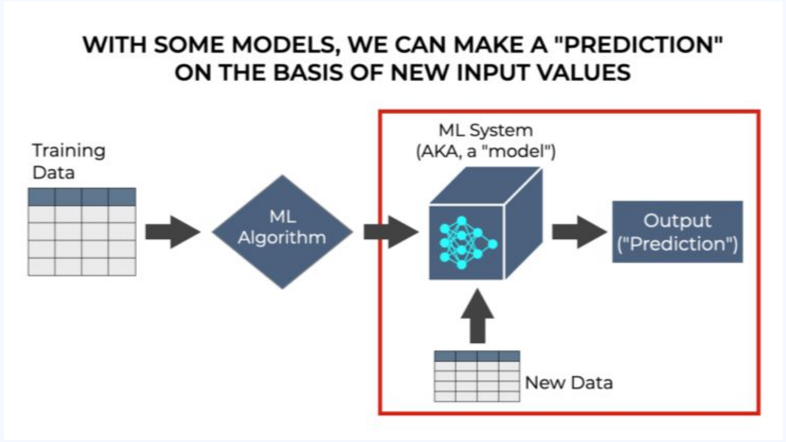
- Learn a model that can provide useful predictions about the data



A. Key Concepts in Training

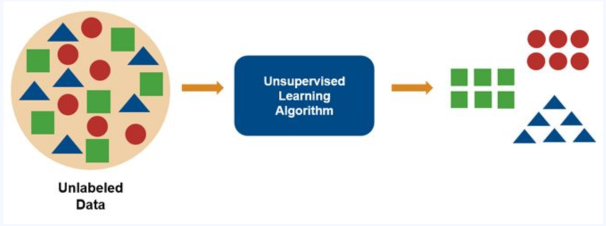
Key Concepts in Training

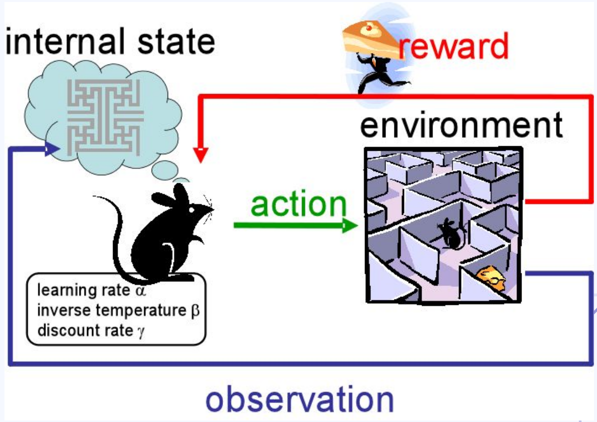
- A dataset is a collection of these (x, y) pairs
- x: independent variable(s)
- y: dependent variable
- A model (that models the relationship between x and y) inputs the data point x and outputs a prediction y_pred


WITH SOME MODELS, WE CAN MAKE A "PREDICTION" ON THE BASIS OF NEW INPUT VALUES
Training Data
ML Algorithm
ML System (AKA, a "model")
Output ("Prediction")
New Data

Key Concepts in Training

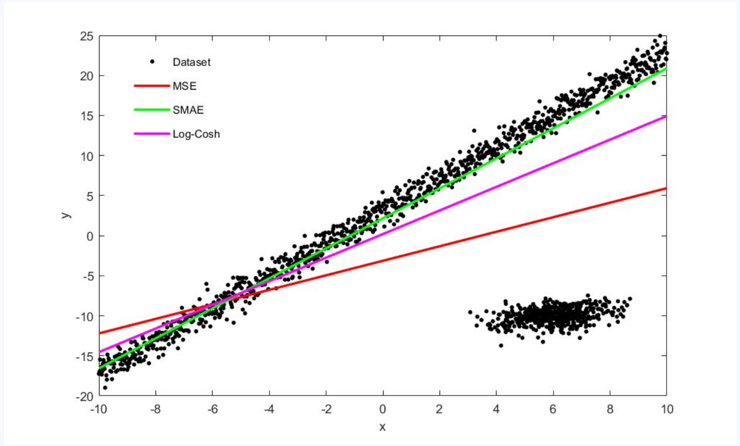
- Supervised learning: learn from (x, y) pairs
- Unsupervised learning: learn from data distribution itself
- Reinforcement learning: learn from “trial and error” using reward signals





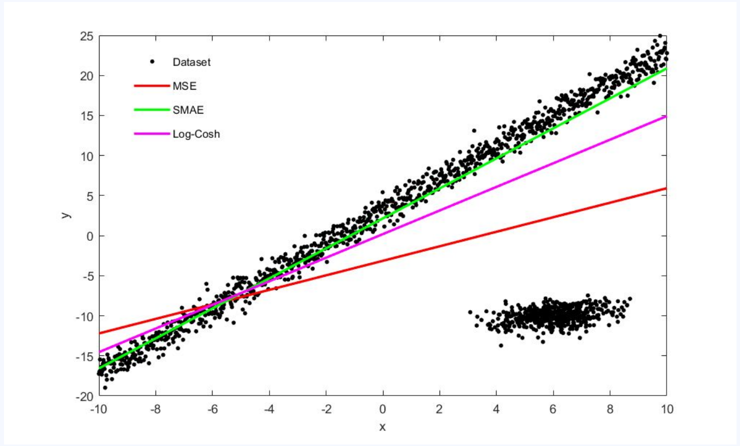
Key Concepts in Training

- y_pred and y are compared through loss functions
- The more similar the prediction (y_pred) is to the ground truth (y), the lower the loss function



Key Concepts in Training

- In supervised learning, we train a model from data so that the output is as close to the ground truth as possible (minimize the loss function)
- Supervised learning requires data-label pairs



Key Concepts in Training

- The more similar the prediction (y_pred) is to the ground truth (y), the lower the loss function
- A common example (in regression, when predicting continuous values) is the mean squared error, which is the average of (y_pred - y)^2 across the entire dataset

$$\mathrm{MSE} = \frac{1}{n} \sum (Y_i - \hat{Y}_i)^2$$

Key Concepts in Training

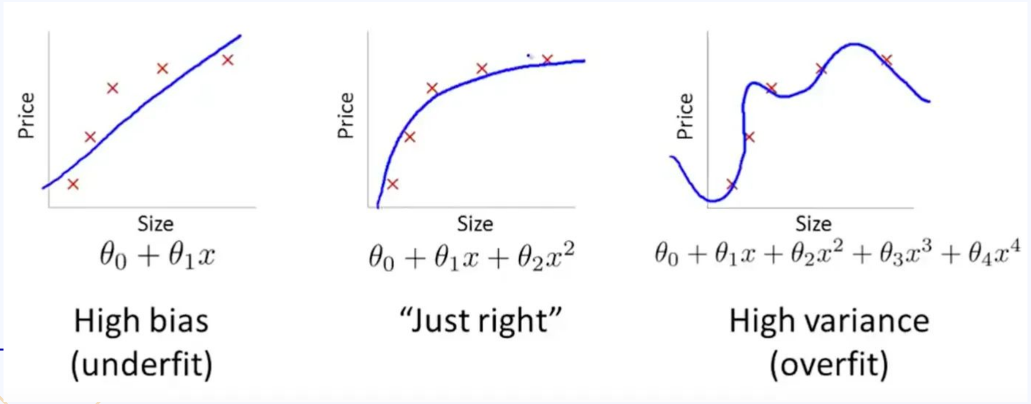
- In classification, the cross entropy loss is used (compare the cross entropy between the predicted probability distribution between classes and the ground truth one-hot distribution)
- The higher the confidence in the correct class, the lower the loss value

Key Concepts in Training

- Good performance on train set does NOT mean good performance on unseen data
- When you train a model, overfitting can occur: the models are tuned so heavily to the training set that performance on unseen data suffers
- We need to split the data into a train set and a test set to detect this

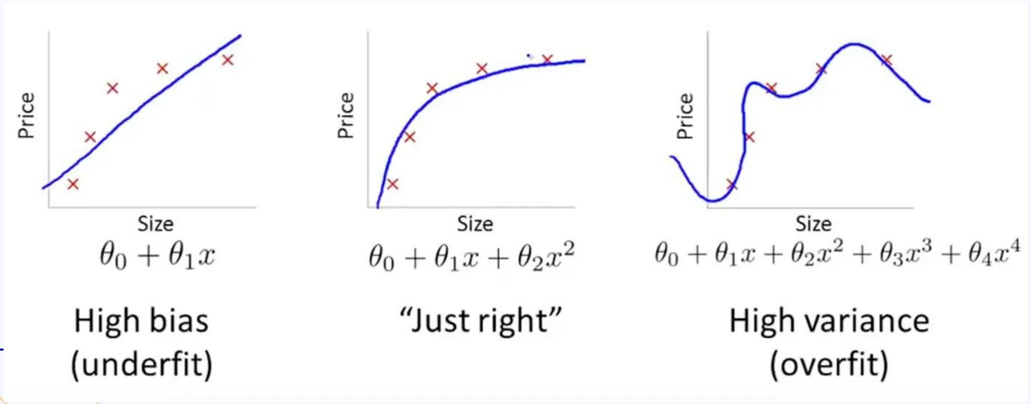
Key Concepts in Training

- Train the model on the train set, and then evaluate on the validation set from time to time
- If your model does poorly on both of these sets, it’s underfitting. The model is too simple to model the necessary dependencies (e.g., using linear regression for nonlinear data).



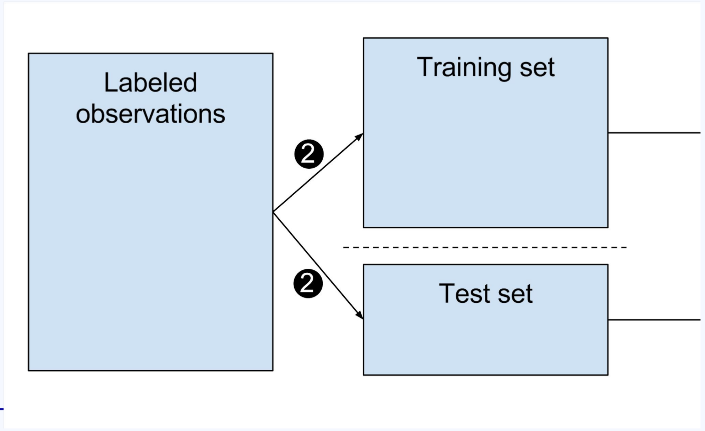
Key Concepts in Training

- If your model does significantly better on the train set than the validation set, the model is overfitting. The model is so complex, or you have trained it for so long, that the model starts capturing the noise in the data instead of the actual underlying patterns.



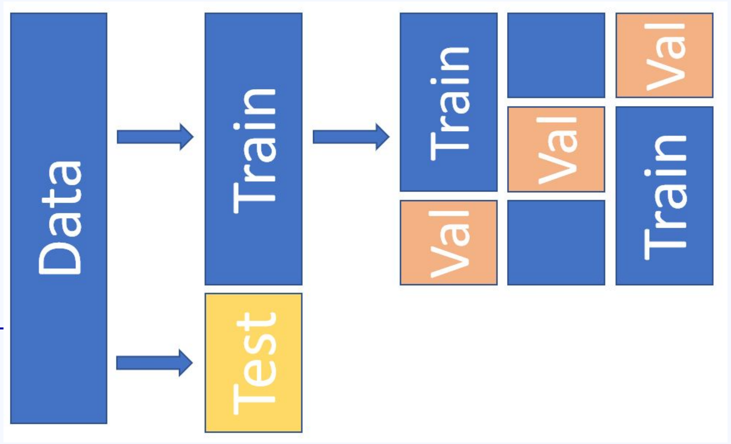
Data Splitting

- We need to split the data into a train set and a validation set to detect overfitting or underfitting
- Generally, we use the "validation" set to assess generalization performance of a model to select models
-



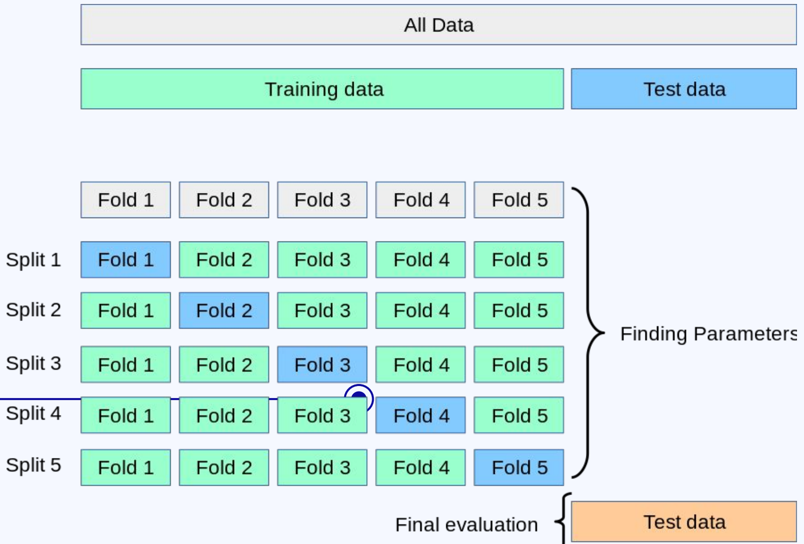
Data Splitting

- To offer one more layer of protection, it is also a common practice to split data further into a 3-way train/validation/test split
- So that some data is held out and used as few times as possible in order to prevent unreliable performance indication from “overfitting” to the test set



Cross Validation

- Another useful way to split data is to use cross validation
- Split the data into k equally sized folds
- For each of the k training runs, hold out one of the folds for validation
- Evaluate for each training run and aggregate metrics



Cross Validation

- Advantage: enables a large portion of the dataset to be used for evaluation, making it more reliable when assessing performance of a training configuration (e.g., hyperparameters, models)
- Disadvantage: takes a longer time to train and evaluate (lengthens the development cycle)

Data Splitting

- For kaggle competition purposes, it's fine to do just train/validation as the "test" sets are hidden from view until you submit; but for real-life applications, the data-splitting pipeline might have to be more rigorous.

Try It Out!

- Experience the data splitting pipeline on https://www.kaggle.com/code/lailaicoding/predict-salary-1

B.

Metrics

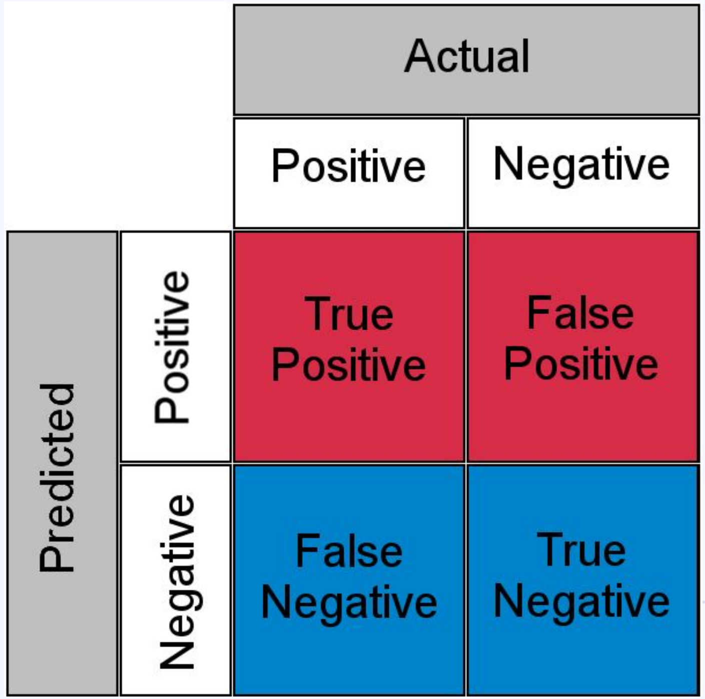
How to evaluate a model

Metrics

- Apart from loss functions, there are other metrics that can be used to measure model performance
- Metrics in regression are similar to loss functions (e.g., MSE, RMSE)
- Metrics in classification are not cross entropy losses

Metrics in Classification

- Accuracy (corrects / total)
- Ratios involving TP (true positive), FP (false positive), TN (true negative), FN (false negative)



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Metrics in Classification

$$\text{Precision} = \frac{TP}{TP + FP}$$

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

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- If either precision or recall is low, F1 score is low as well
- F1 score demands balance between precision and recall

When to Use Which Metric?

- When choosing or designing a metric, you look at which kinds of behavior are penalized
- e.g., compare the costs of missing an actual positive sample and detecting a false positive
- In kaggle competition scenarios this is usually not your job (just follow the metric provided), but in actual applications it's very important to account for that

When to Use Which Metric?

- False negatives are serious (e.g., in disease detection) -> lean towards recall
- False positives are serious -> lean towards precision
- Balance between precision and recall: F1 score (requires good precision AND recall to get good score)
- If dataset is class-balanced (each class contains roughly the same number of elements): Accuracy
- (though in safety-critical applications precision / recall based evaluations are still recommended)

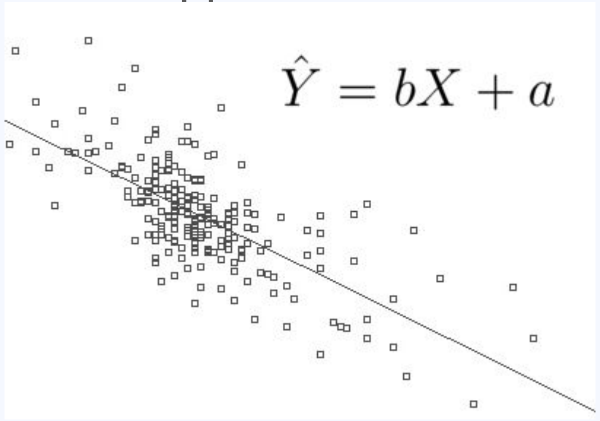
C.

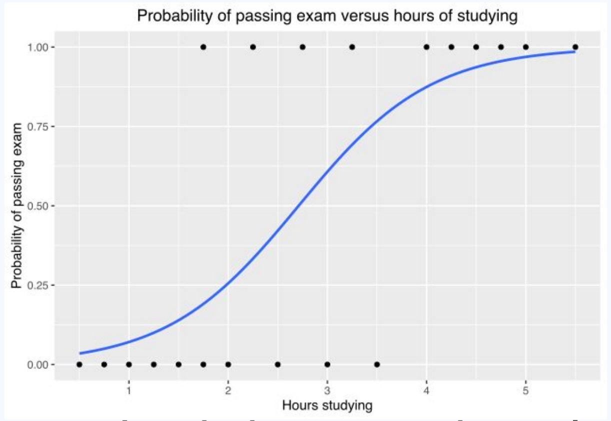
Models

Choose the best model for your projects

Linear Models

- The simplest of machine learning models are linear models.
- It applies a linear transform onto the data to generate predictions.





- Linear regression: the output is a weighted sum of the input features (plus some bias term)

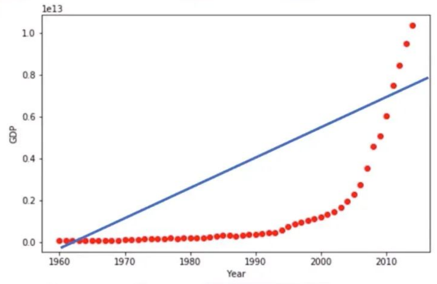
- Logistic regression: classifying a target variable using the value of the weighted sum of the input variables

Linear Models

- Advantages: very simple and highly interpretable (can look into how the model predicts)
- Disadvantages: cannot model nonlinear relationships

Should we use linear regression?

	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10
...	.....	....



COGNITIVE CLASS

2

Image source: https://medium.com/@toprak.mhmt/non-linear-regression-4de80afca347

-

Decision Trees

- Using a tree-like structure, decide the output prediction based on the input variables

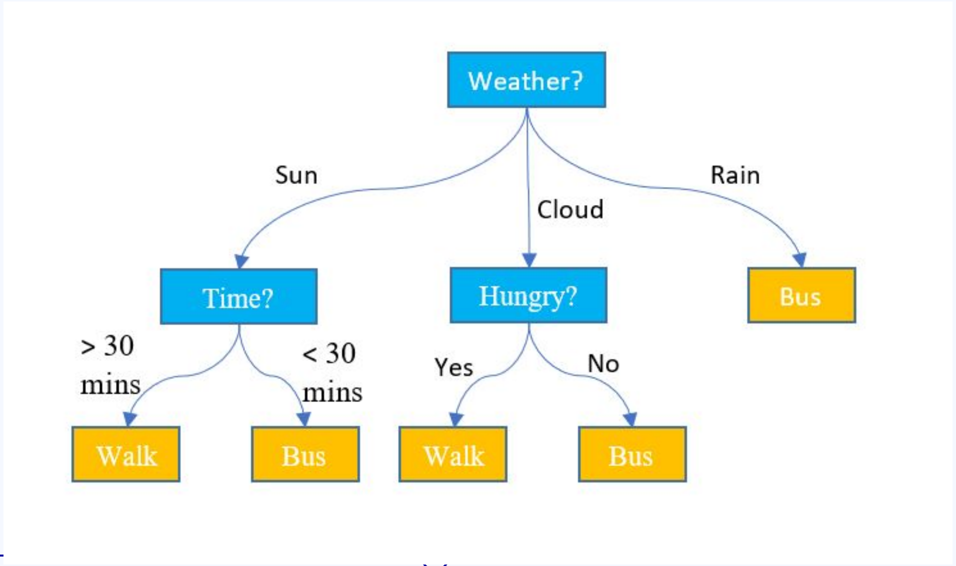


Image source: https://www.displayr.com/what-is-a-decision-tree/

-

Decision Trees

- Random forest: many decision trees ensembled with each other
- Their predictions are averaged or voted

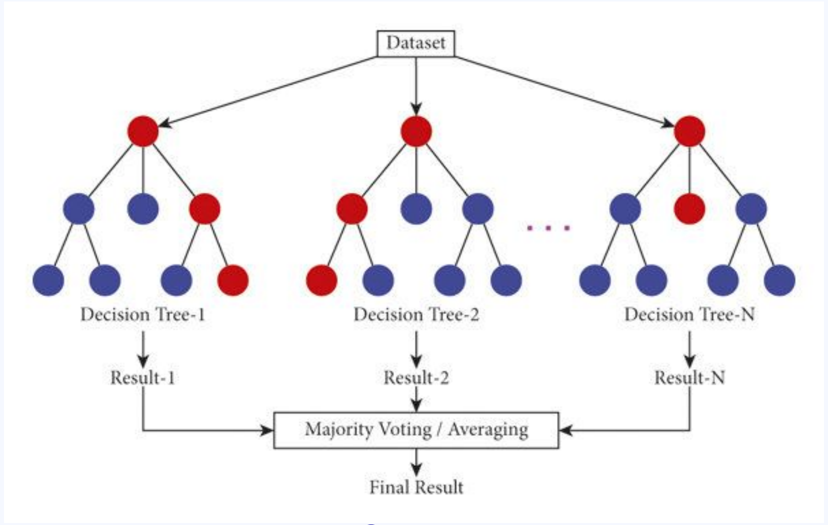


Image source: https://www.researchgate.net/figure/Illustration-of-random-forest-trees_fig4_354354484

-

Decision Trees

- Advantages: highly interpretable (especially for small number of decision trees), extensive regularization settings
- Disadvantages: can overfit data easily under default configurations

- Boosted decision trees: decision trees can be boosted on top of each other
- Each decision tree in the boosting setup predicts the residual (prediction - ground truth) of the upstream trees.

Try It Out!

- In the notebook, implement linear models and decision trees using the scikit-learn library
- https://www.kaggle.com/code/lailaicoding/predict-salary-1

slidesgo

Instructions for use

Machine Learning

- Learn a model that can provide useful predictions about the data
- (input -> process -> output)

Key Concepts in Training

- A dataset is a collection of these (x, y) pairs
- A neural network (or a model in general) inputs the data point x and outputs a prediction y_pred

Key Concepts in Training

- y_pred and y are compared through loss functions
- The more similar the prediction (y_pred) is to the ground truth (y), the lower the loss function

Key Concepts in Training

- In supervised learning, we train a model from data so that the output is as close to the ground truth as possible (minimize the loss function)
- Supervised learning requires data-label pairs

Key Concepts in Training

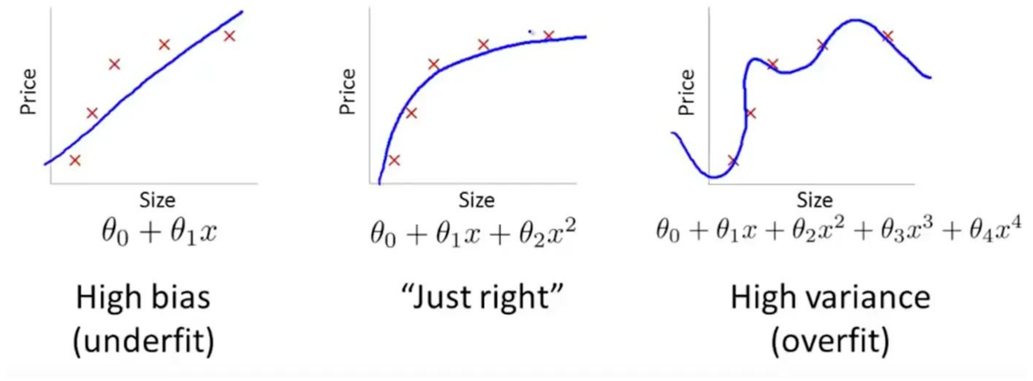
- The more similar the prediction (y_pred) is to the ground truth (y), the lower the loss function
- A common example (in regression, when predicting continuous values) is the mean squared error, which is the average of (y_pred - y)^2 across the entire dataset
- In classification, the cross entropy loss is used (compare the cross entropy between the predicted probability distribution between classes and the ground truth one-hot distribution)

Key Concepts in Training

- Good performance on train set != good performance on test set
- When you train a model, overfitting can occur: the models are tuned so heavily to the training set that performance on the test set suffers
- We need to split the data into a train set and an test set to detect this

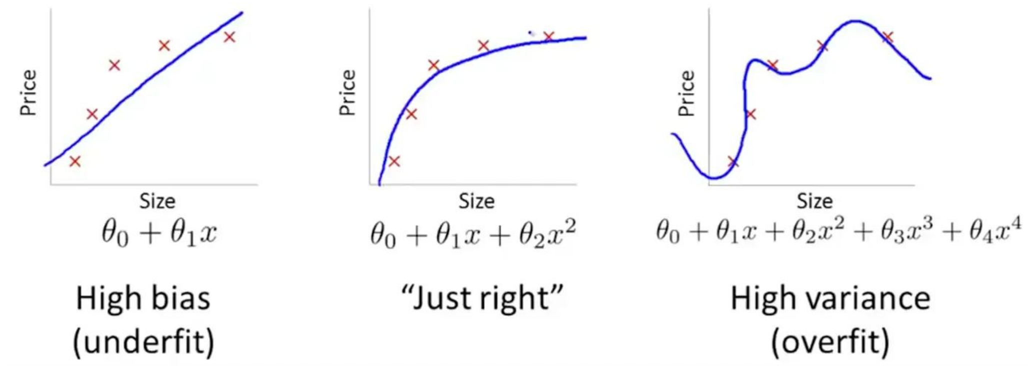
Key Concepts in Training

- Train the model on the train set, and then evaluate on the test set from time to time
- If your model does poorly on both of these sets, it's underfitting. The model architecture is too simple to model the necessary dependencies (e.g., using linear regression for nonlinear data).



Key Concepts in Training

- Train the model on the train set, and then evaluate on the test set from time to time
- If your model does significantly better on the train set than the test set, the model is overfitting. The model architecture is so complex, and you have trained it for so long, that the model starts capturing the noise in the data instead of the actual underlying patterns.



Key Concepts in Training

- We need to split the data into a train set and a test set to detect this
- “Generally”, we use the test set to assess generalization performance of a model to select models

Key Concepts in Training

- To offer one more layer of protection, it is also a common practice to split the train set further into the train and validation set to make it a 3-way train/val/test split
- So that some data is held out and used as few times as possible in order to prevent unreliable performance indication from “overfitting” to the test set

Key Concepts in Training

- For competition purposes, it’s fine to do just train/val as the “test” sets are hidden from view until you submit; but for real-life applications, the data-splitting pipeline might have to be more rigorous.

Try It Out!

- Experience the data splitting pipeline on https://colab.research.google.com/drive/1PSDxQRr4eyO7HHenijMIXCM_kAk22BYa#scrollTo=anRQ6hLzyPnu

Metrics

- Apart from loss functions, there are other metrics that can be used to measure model performance
- Metrics in regression are similar to loss functions (e.g., MSE, RMSE)
- Metrics in classification are not cross entropy losses

Metrics in Classification

- Accuracy (corrects / total)
- Ratios involving TP (true positive), FP (false positive), TN (true negative), FN (false negative) (include diagraam)

Metrics in Classification

- Precision (TP / (TP + FP))
- Recall (TP / (TP + FN))
- F1 score (2 / (1 / Precision + 1 / Recall))
- If either precision or recall is low, F1 score is low as well
- F1 score demands balance between precision and recall

When to Use Which Metric?

- When choosing or designing a metric, you look at which kinds of behavior are penalized
- e.g., compare the costs of missing an actual positive sample and detecting a false positive
- In competition scenarios this is usually not your job, but in actual applications it’s very important to account for that

When to Use Which Metric?

- False negatives are serious (e.g., in disease detection) -> lean towards recall
- False positives are serious (e.g.,) -> lean towards precision
- Balance between precision and recall: F1 score (requires good precision AND recall to get good score)
- If dataset is class-balanced (each class contains roughly the same number of elements): Accuracy
- (though in safety-critical applications precision / recall based evaluations are still recommended)

Linear Models

- The simplest of machine learning models are linear models.
- It applies a linear transform onto the data to generate predictions.

(maybe put these slides onward into that colab notebook? I’m not sure)

Linear Models

- Linear regression: the output is a weighted sum of the input features

(write down a latex formula)

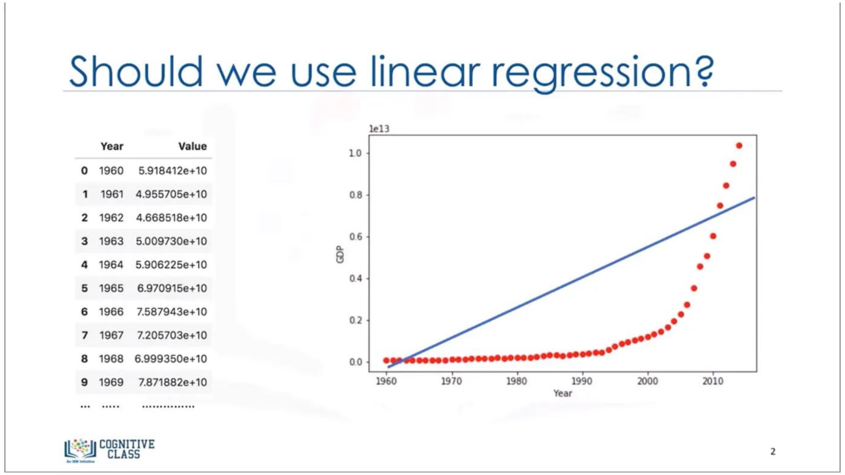
Linear Models

- Logistic regression: classifying a target variable using the value of the weighted sum of the input variables

(write down a latex formula)

Linear Models

- Advantages: very simple and highly interpretable (can look into how the model predicts)
- Disadvantages: cannot model nonlinear relationships



Should we use linear regression?

	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10
5	1965	6.970915e+10
6	1966	7.587943e+10
7	1967	7.205703e+10
8	1968	6.999350e+10
9	1969	7.871882e+10
...	.....	....

2

Image source: https://medium.com/@toprak.mhmt/non-linear-regression-4de80afca347

Decision Trees

- Using a tree-like structure, decide the output prediction based on the input variables

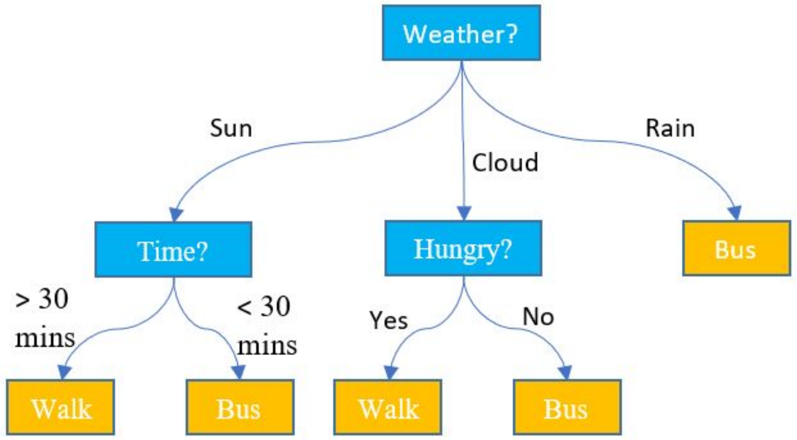


Image source: https://www.displayr.com/what-is-a-decision-tree/

Decision Trees

- Random forest: many decision trees ensembled with each other
- Their predictions are averaged or voted

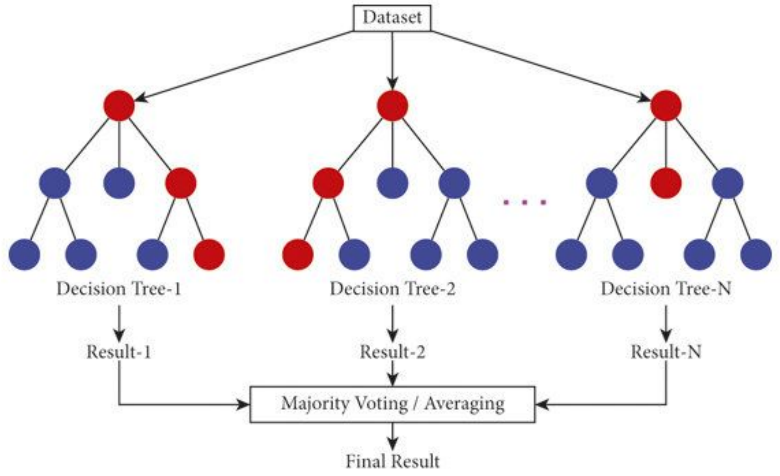


Image source: https://www.researchgate.net/figure/Illustration-of-random-forest-trees_fig4_354354484

Decision Trees

- Advantages: highly interpretable (especially for small number of decision trees), extensive regularization settings
- Disadvantages: can overfit data easily under default configurations

Decision Trees

- Boosted decision trees: decision trees can be boosted on top of each other
- Each decision tree in the boosting setup predicts the residual (prediction - ground truth) of the upstream trees.

Try It Out

- In the notebook, implement linear models and decision trees using the scikit-learn library https://colab.research.google.com/drive/1PSDxQRr4eyO7HHenijMIXCM_kAk22BYa#scrollTo=dGJHzTqIxGkf