Week 2: Building TensorFlow Applications

Unit 1: Nuts and Bolts of Deep Learning

Run Simple

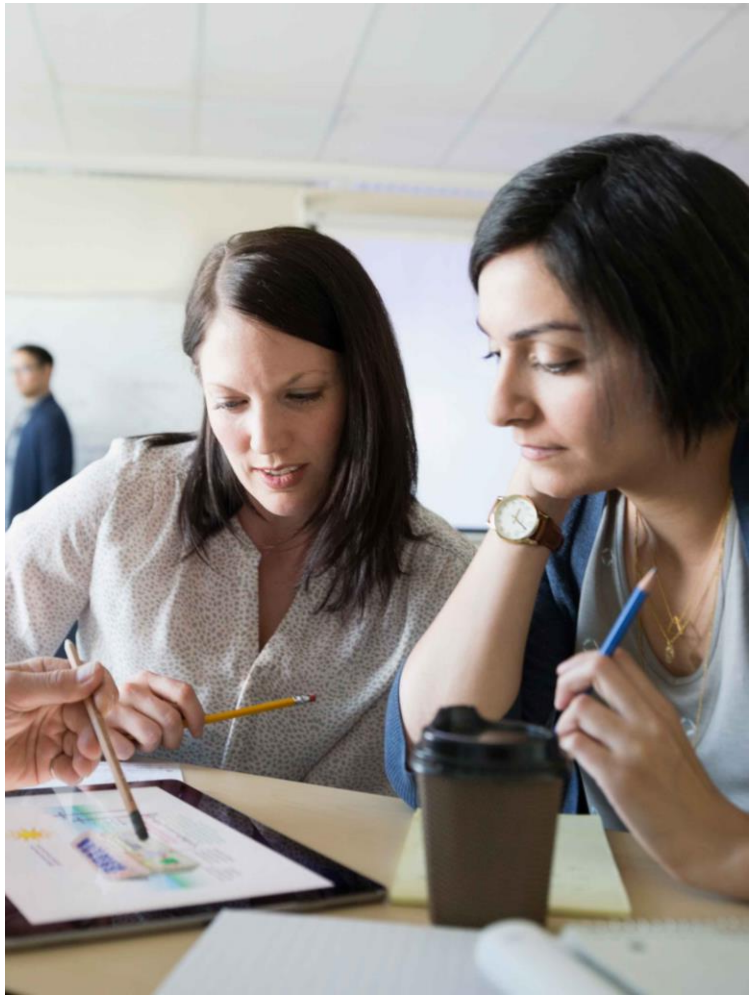
open.sap.com

Nuts and Bolts of Deep Learning

What we covered in the last week

Getting Started with Deep Learning

- From neural networks to deep learning
- Jupyter notebooks
- First neural network
- Introduction to TensorFlow
- When to use deep learning



Nuts and Bolts of Deep Learning

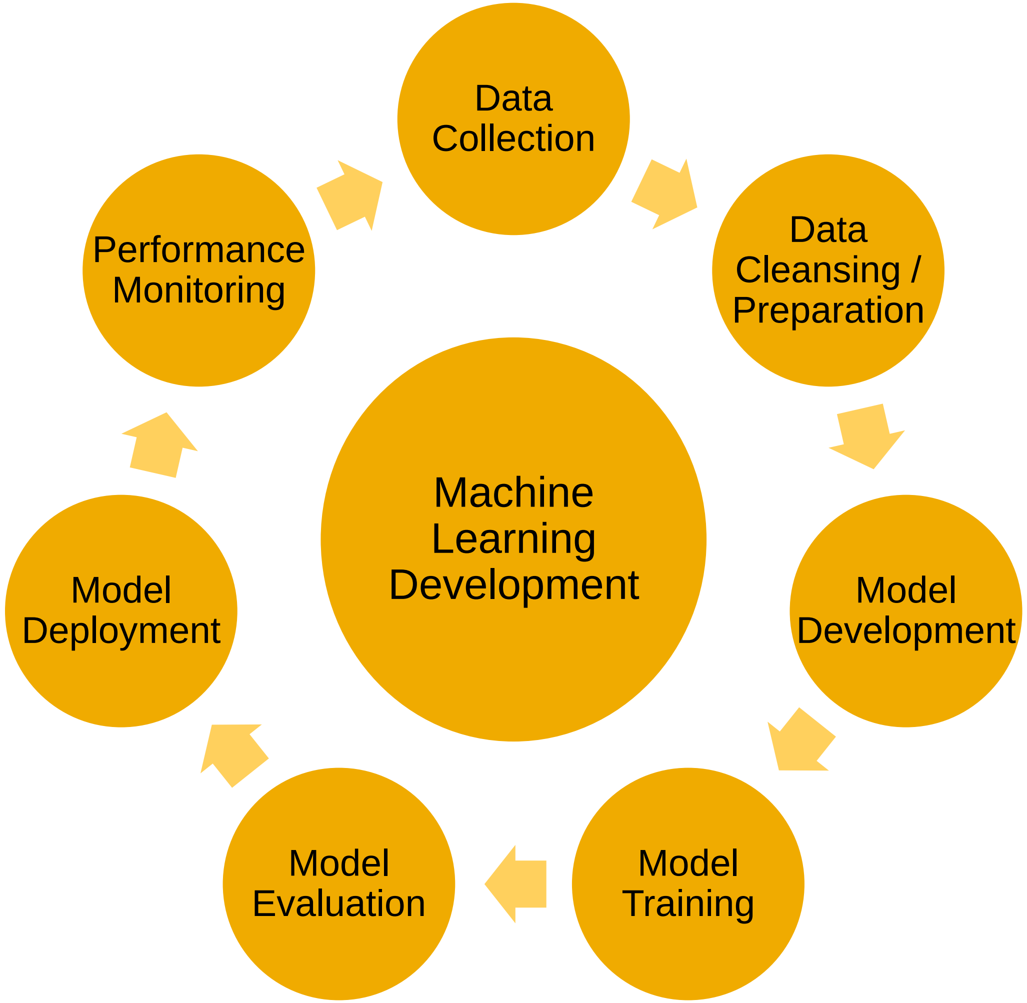
General machine learning process

Software development

- Well-defined requirements for features
- Outcomes are dependent on engineering capabilities

Machine learning development

- Model development involves trial and error and experimentation
- Outcomes are dependent on data, model parameters, and other parameters



Nuts and Bolts of Deep Learning

The train, validate, and test routine

Training

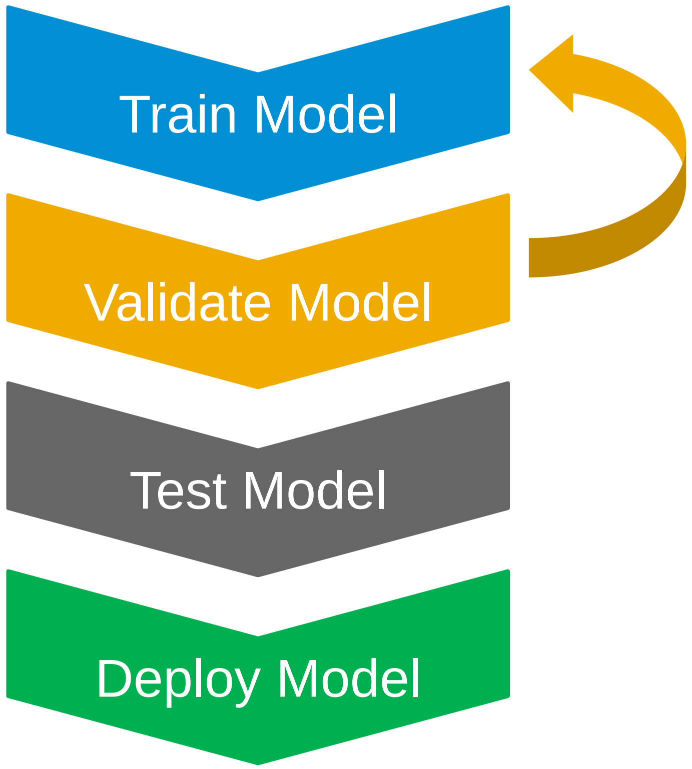
- Optimize model parameters on training data

Validation

- Tune hyperparameters on dev data

Test

- Approximate production performance on test data



Nuts and Bolts of Deep Learning

The train, dev, and test set

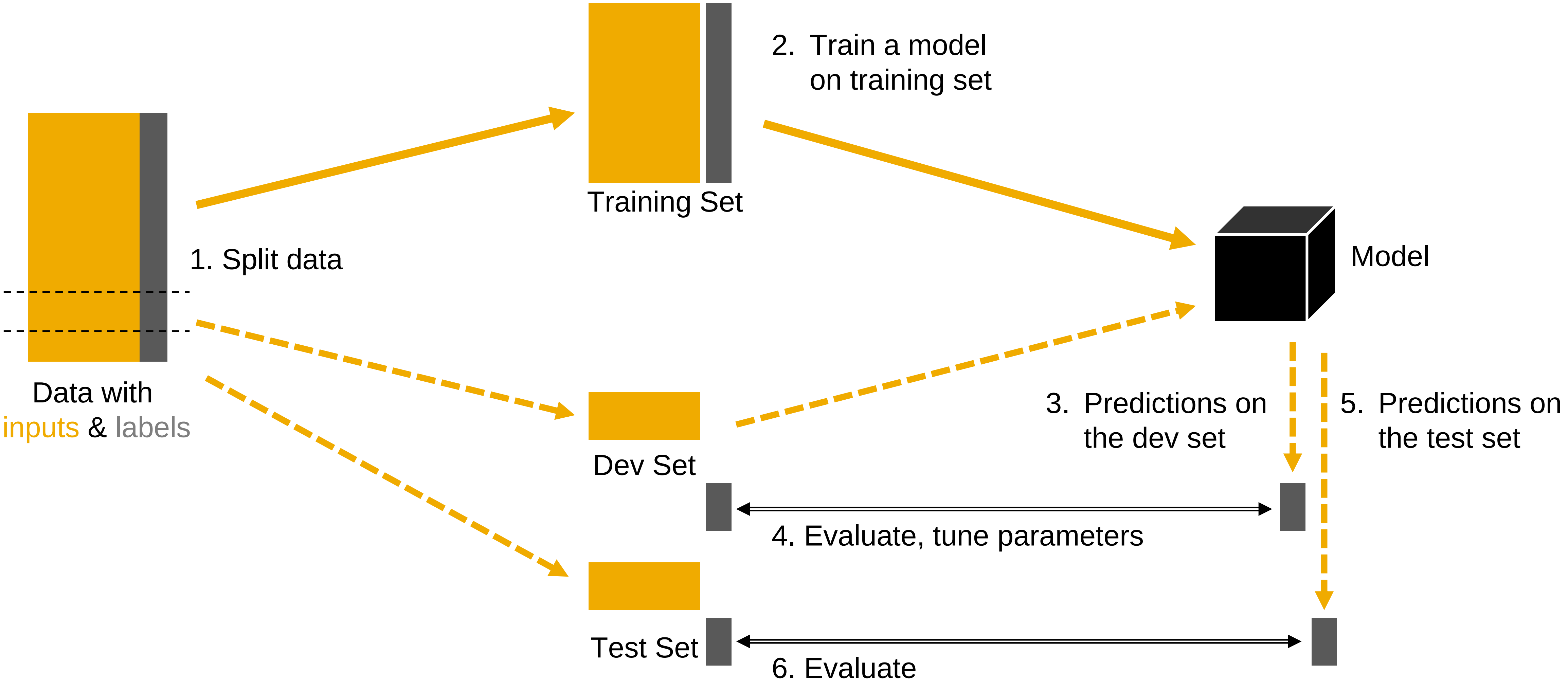
- Split data into train, dev, and test set
- Ensures model generalizes well on unseen data
- Dev and test set should be from the same distribution as training set
- Dev and test set should be similar and proxy for production
- Dev and test set large enough for robust evaluation

Classic split 60:20:20

Split for deep learning

Nuts and Bolts of Deep Learning

The train, validate, and test routine

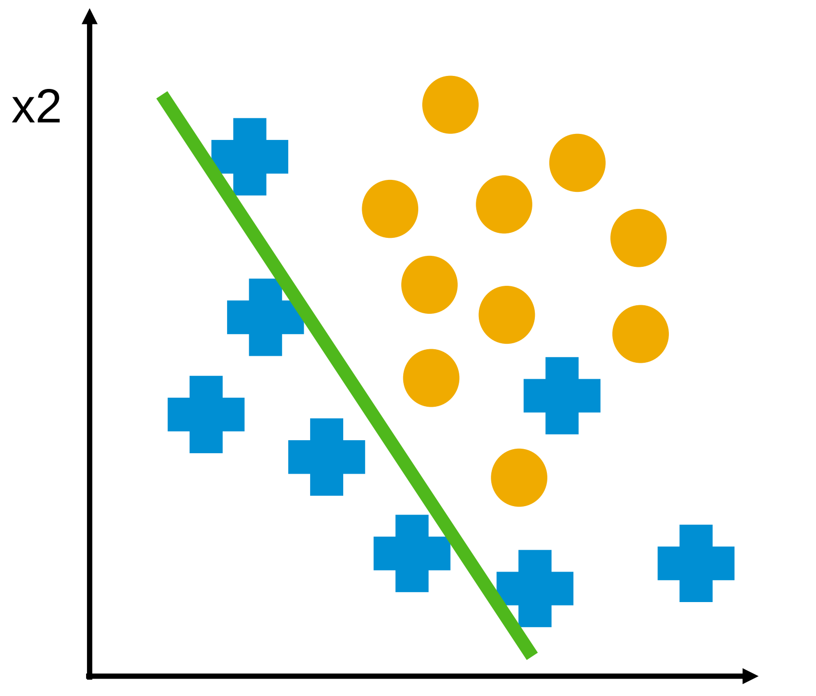


Nuts and Bolts of Deep Learning

Generalization and regularization

Generalization

- Ability of a learning algorithm to perform well on unseen examples
- Underfit: network cannot model the data



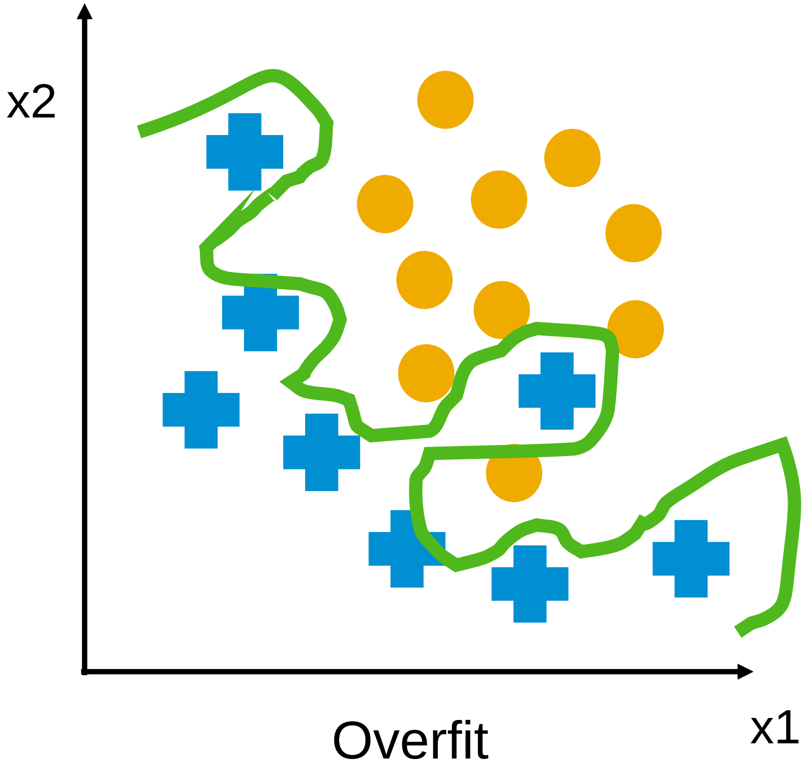
Underfit

Nuts and Bolts of Deep Learning

Generalization and regularization

Generalization

- Ability of a learning algorithm to perform well on unseen examples
- Overfit: network captures idiosyncratic noise, does not generalize



Overfit

Nuts and Bolts of Deep Learning

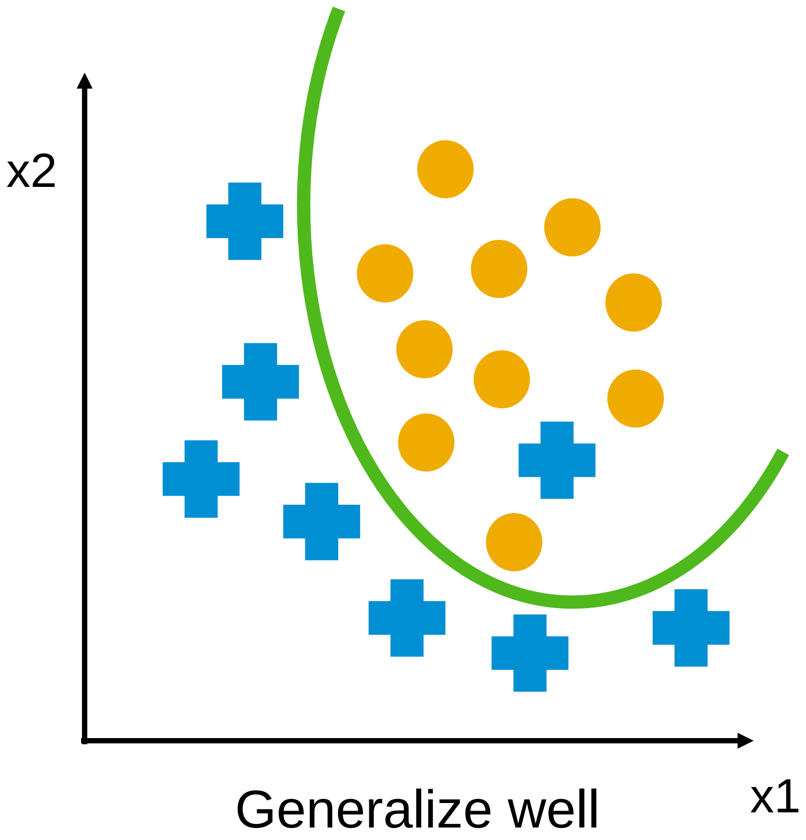
Generalization and regularization

Generalization

- Ability of a learning algorithm to perform well on unseen examples
- Generalize well: fit training data well but also perform well on new data, model underlying “true” pattern

Regularization

- Techniques to reduce overfitting
- Improves ability of the model to generalize
- L1/L2 regularization, dropout



Nuts and Bolts of Deep Learning

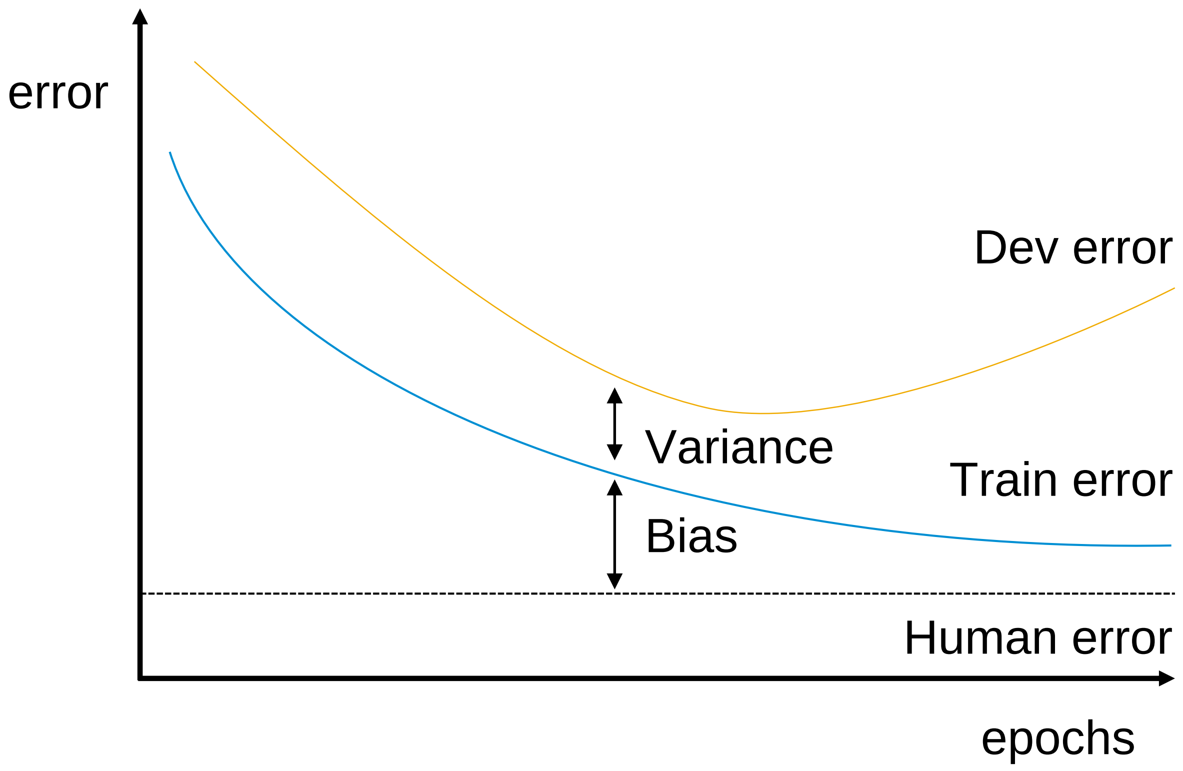
Bias-variance tradeoff

Bias

- High-bias models fail to capture patterns
- Leads to underfitting

Variance

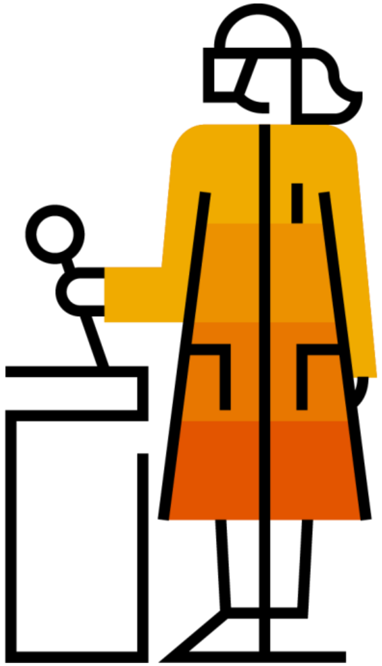
- High-variance models represent training data very well
- Leads to overfitting



Nuts and Bolts of Deep Learning

Deep learning recipe

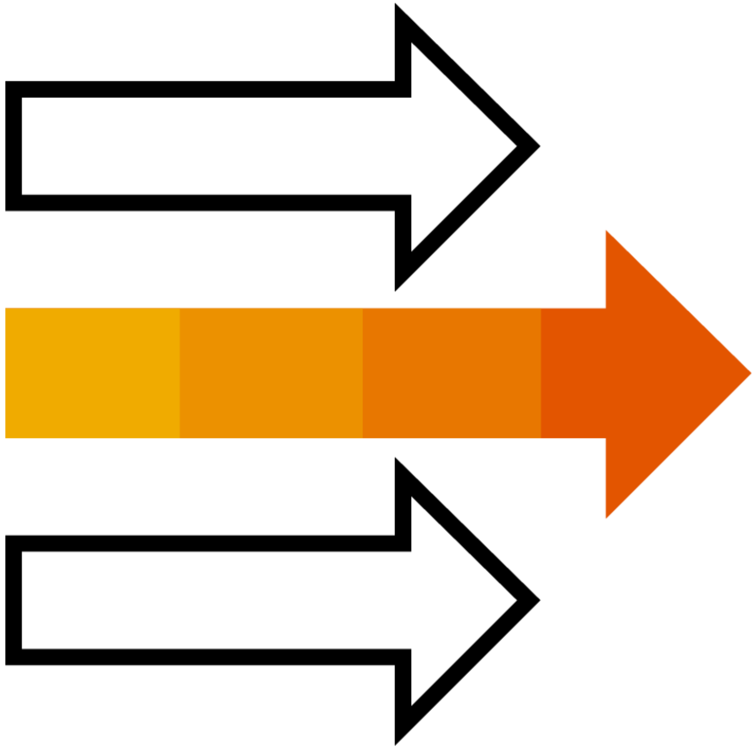
1. **Reduce train error**
   - larger network, train longer, learning rate, batch size, momentum
2. **Reduce dev error**
   - Regularization, larger training set
3. **Reduce test error**
   - Increase dev set, ensure dev and test set similar



Nuts and Bolts of Deep Learning

Coming up next

Visualizing and Preparing Data



Thank you.

Contact information:

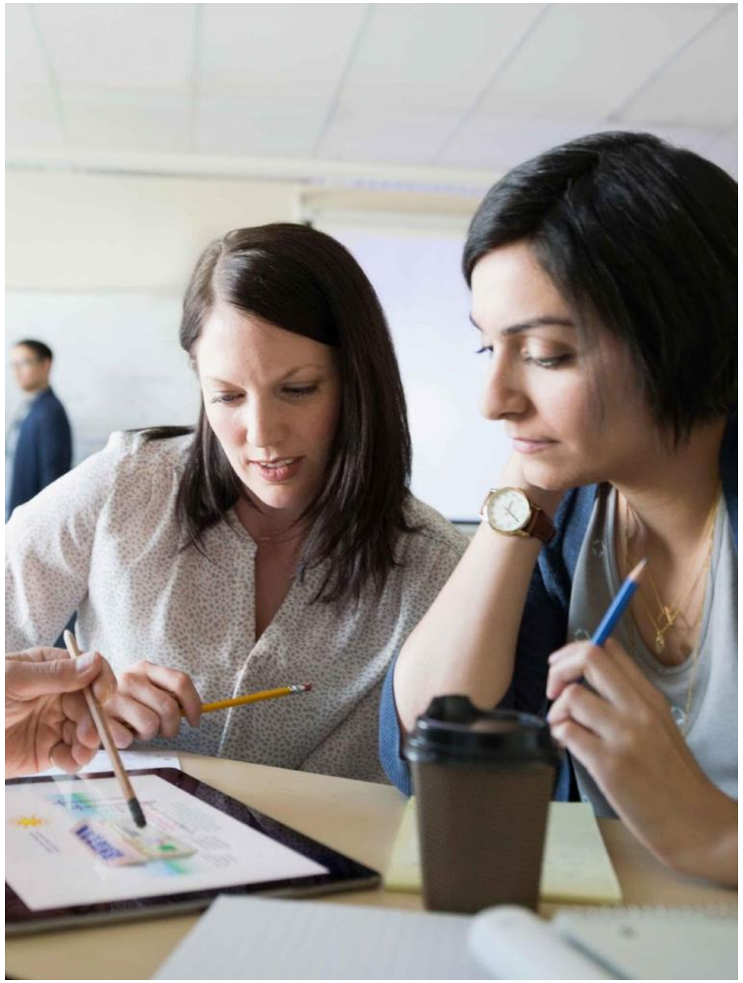
open@sap.com

Week 2: Building TensorFlow Applications

Unit 2: Intro to Classifying Structure Data with TensorFlow

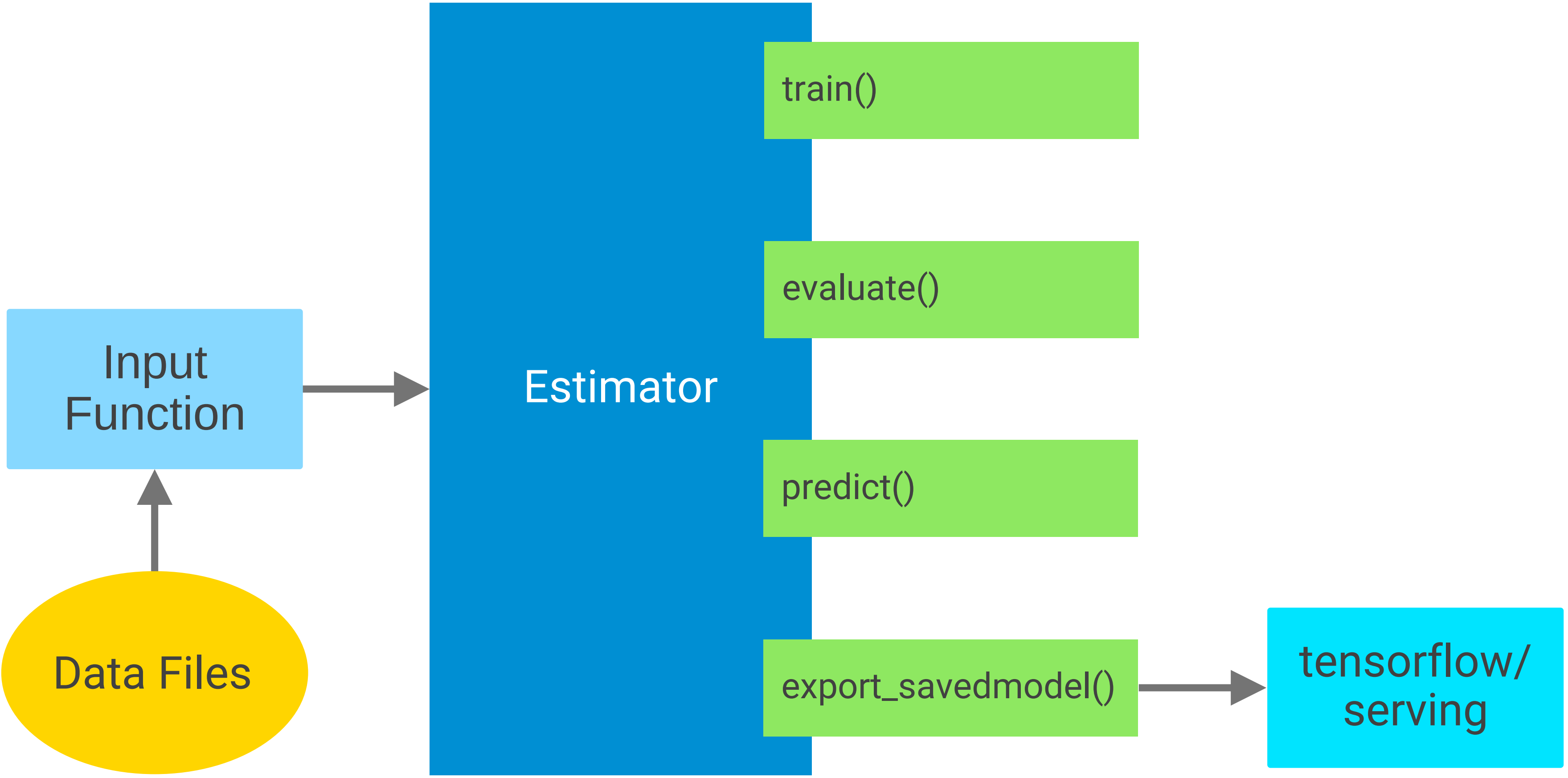
Intro to Classifying Structure Data with TensorFlow
Outline

- Visualize a dataset with facets
- Train a model using a canned estimator
- Feature engineering



Intro to Classifying Structure Data with TensorFlow

Estimators

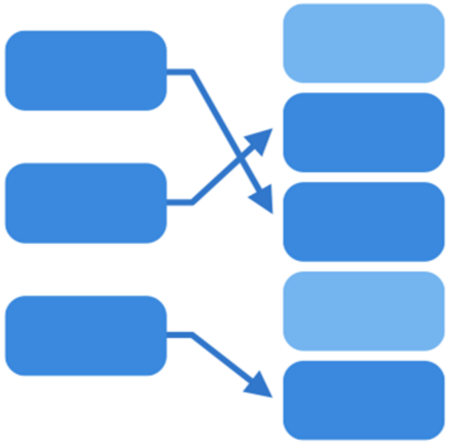


Intro to Classifying Structure Data with TensorFlow

Feature engineering

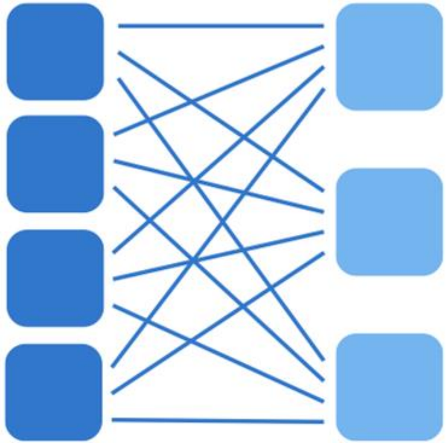
Bucketing

Hashing



Embedding

Crossing



Intro to Classifying Structure Data with TensorFlow

Train/Test Skew

- Over time, the data you use in production may no longer resemble the data you used to train your model
- Example:
  - Income model trained on 1990 census data
  - Because of inflation, more people make >= $50,000 overtime
  - Model becomes inaccurate

Intro to Classifying Structure Data with TensorFlow

Code at this Link

Code in the description, and also at this link
https://goo.gl/R1i7Bw

Thank you.

Contact information:

open@sap.com

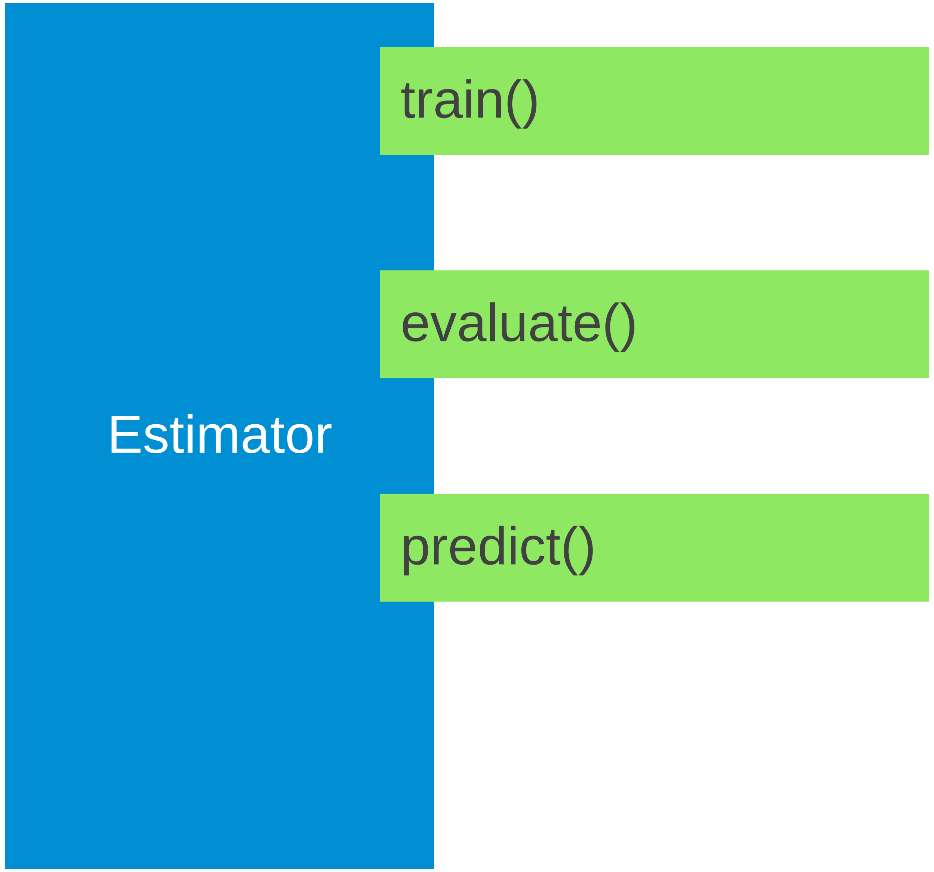
Week 2: Building TensorFlow Applications

Unit 3: Canned Estimators

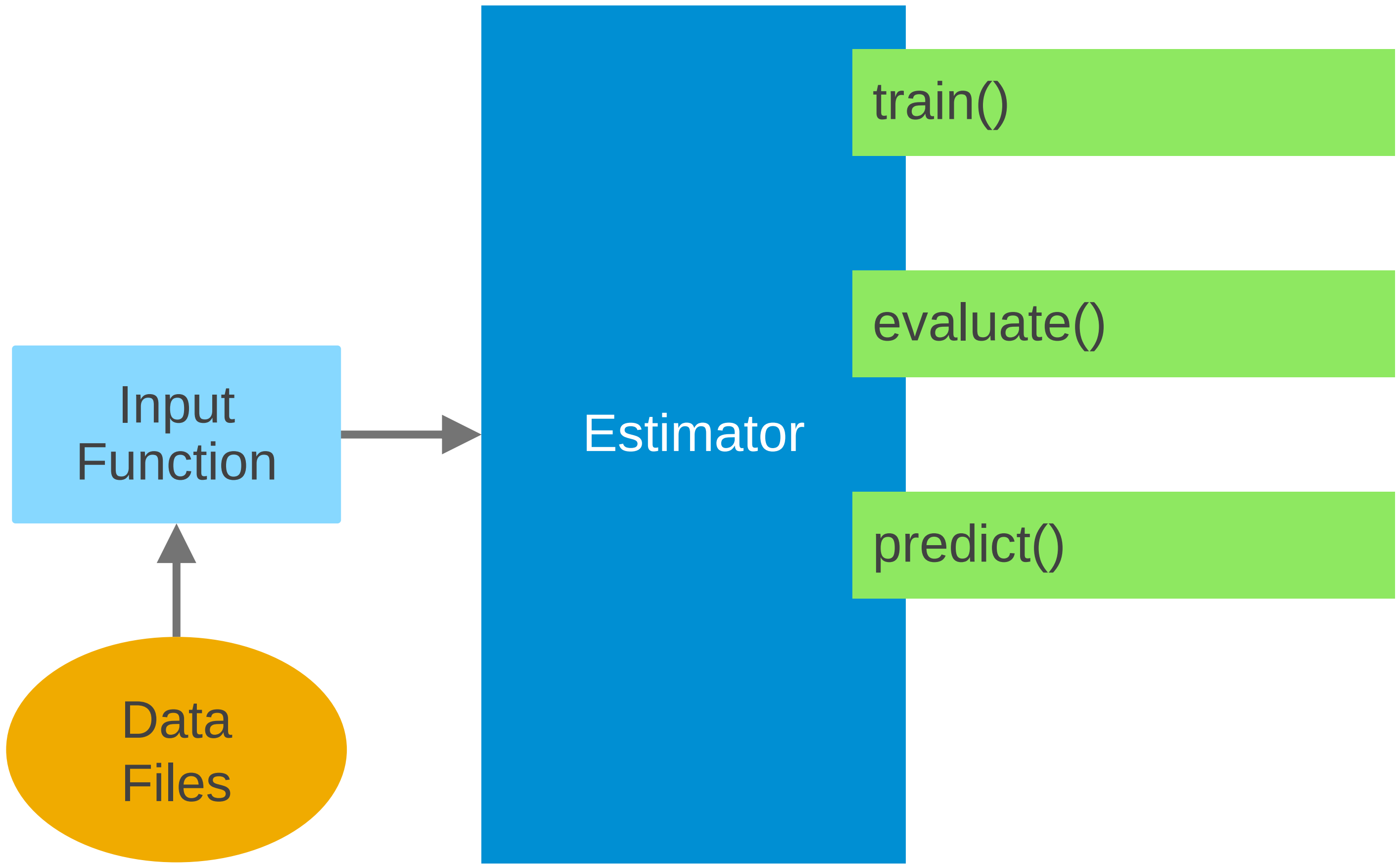
SAP Run Simple

openSAP
open.sap.com

Canned Estimators
About



Canned Estimators
Input functions

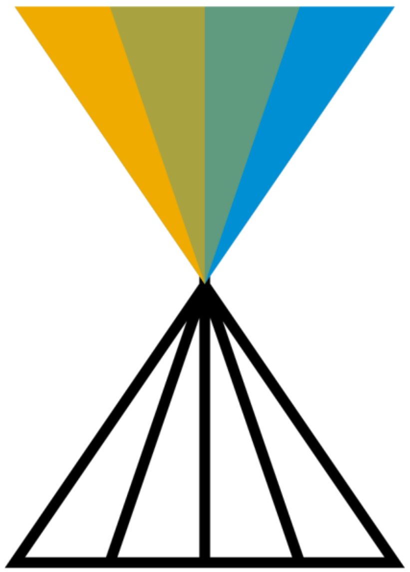


Canned Estimators
Input functions

https://www.tensorflow.org/get_started/input_fn

```
import numpy as np
# numpy input_fn.
my_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": np.array(x_data)},
    y=np.array(y_data),
    ...)
```

```
import pandas as pd
# pandas input_fn.
my_input_fn = tf.estimator.inputs.pandas_input_fn(
    x=pd.DataFrame({"x": x_data}),
    y=pd.Series(y_data),
    ...)
```



Canned Estimators
Datasets API

Simple and high-performance way to design custom input functions

https://www.tensorflow.org/programmers_guide/datasets

https://developers.googleblog.com/2017/09/introducing-tensorflow-datasets.html

Thank you.

Contact information:

open@sap.com

Week 2: Building TensorFlow Applications

Unit 4: Feature Engineering

Run Simple

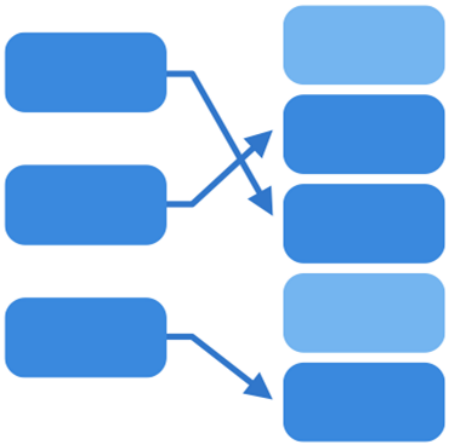
openSAP
open.sap.com

Feature Engineering

Feature engineering

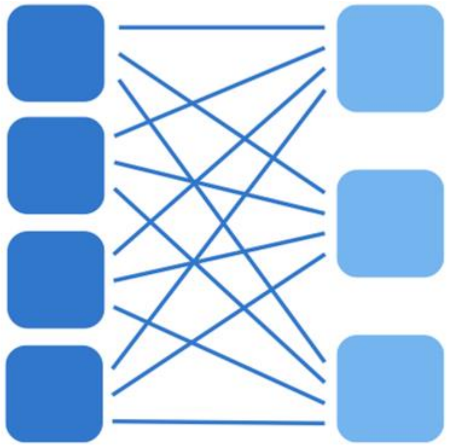
Bucketing

Hashing



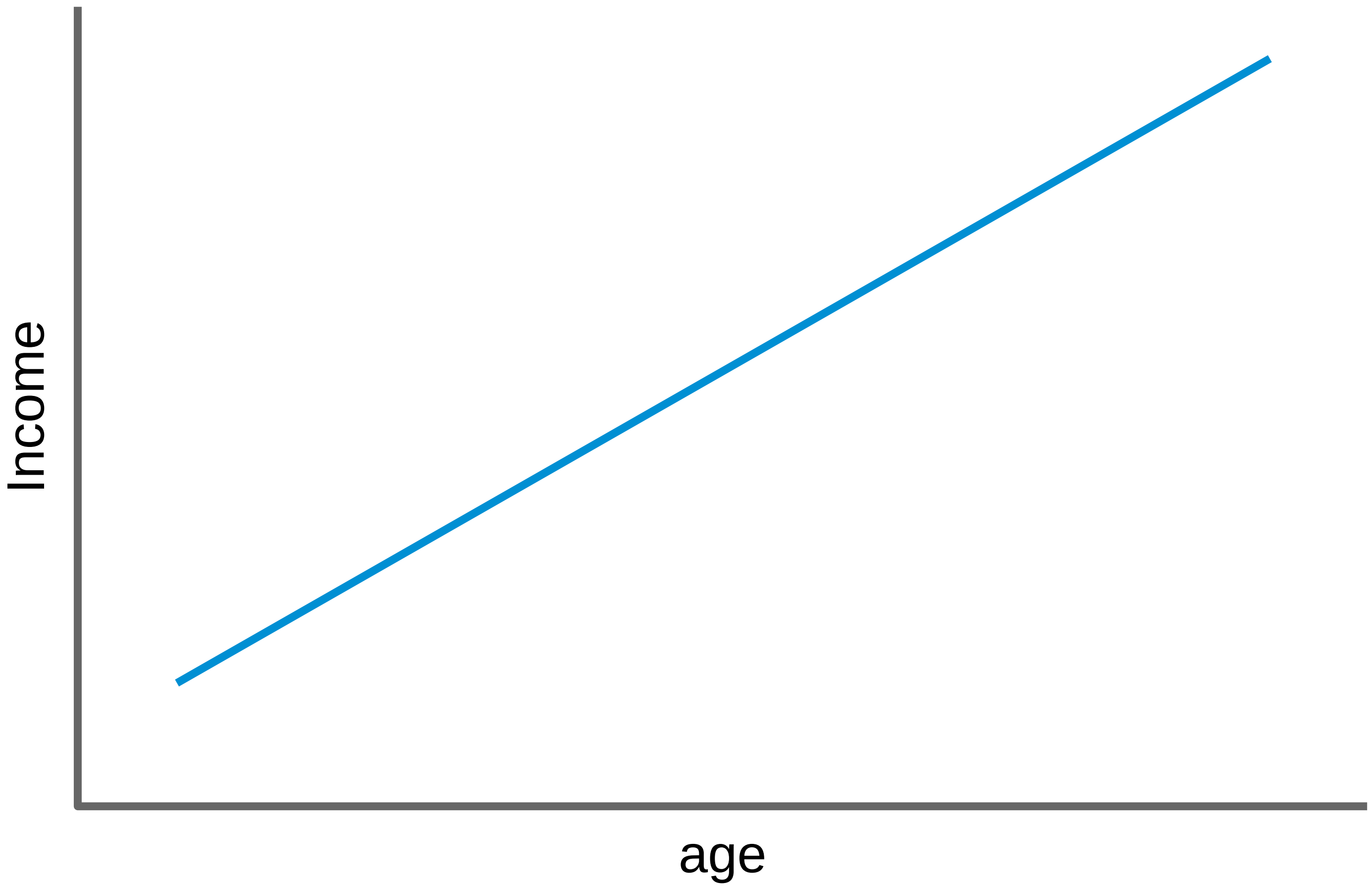
Embedding

Crossing



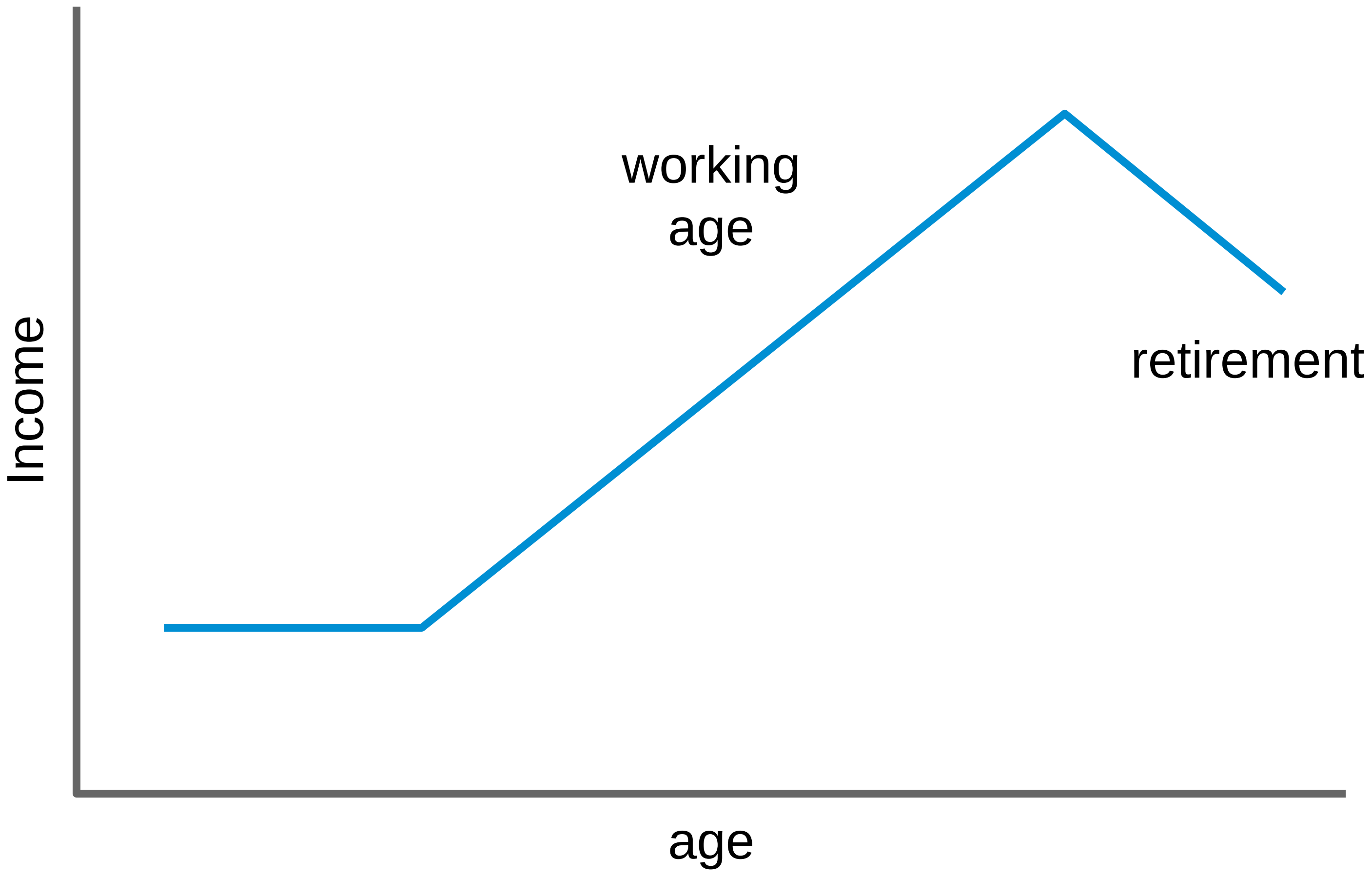
Feature Engineering

What can go wrong with this approach?



Feature Engineering

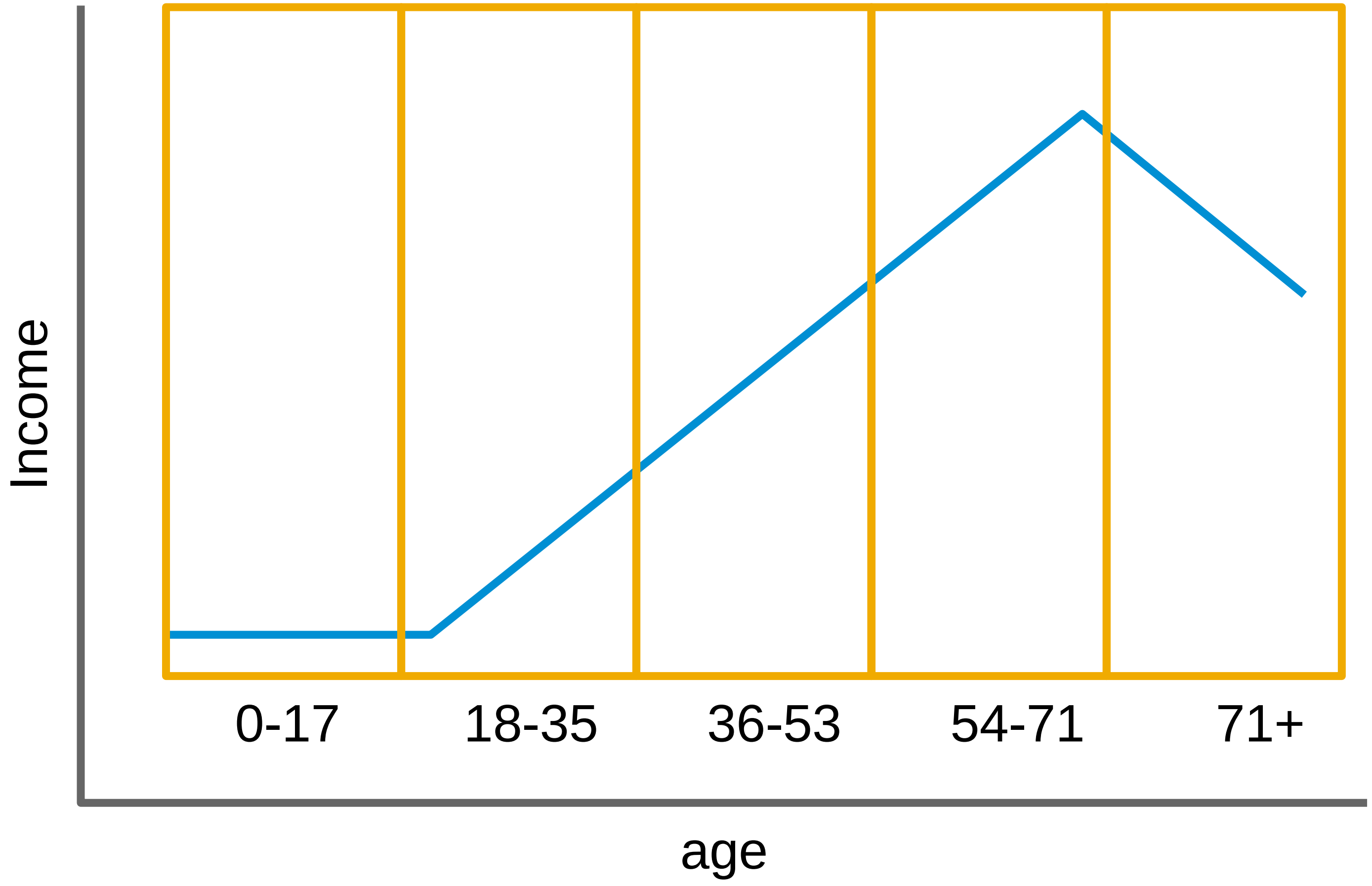
Relationship is not linear



Feature Engineering

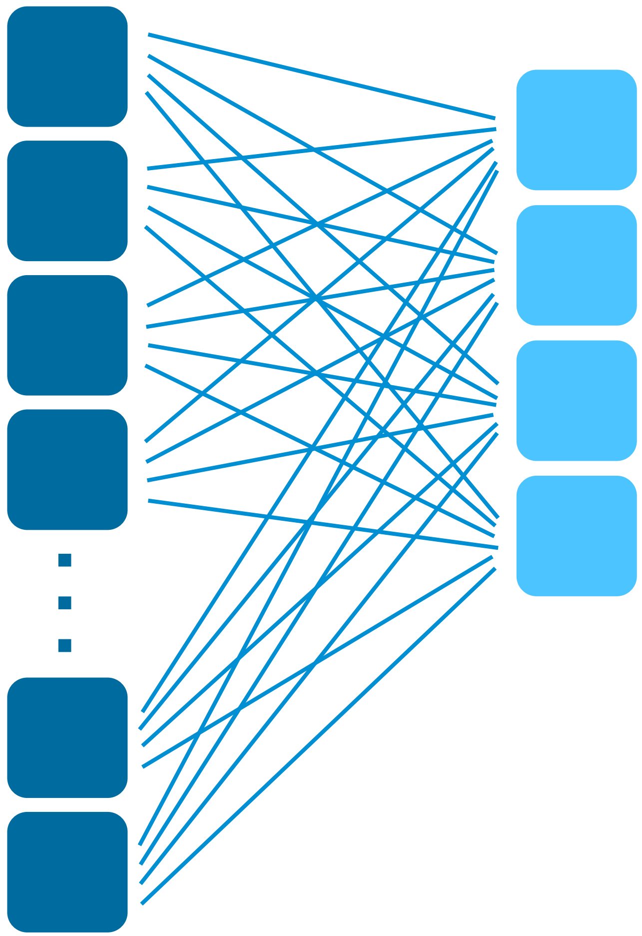
Bucketing

- One categorical feature is created for each bucket



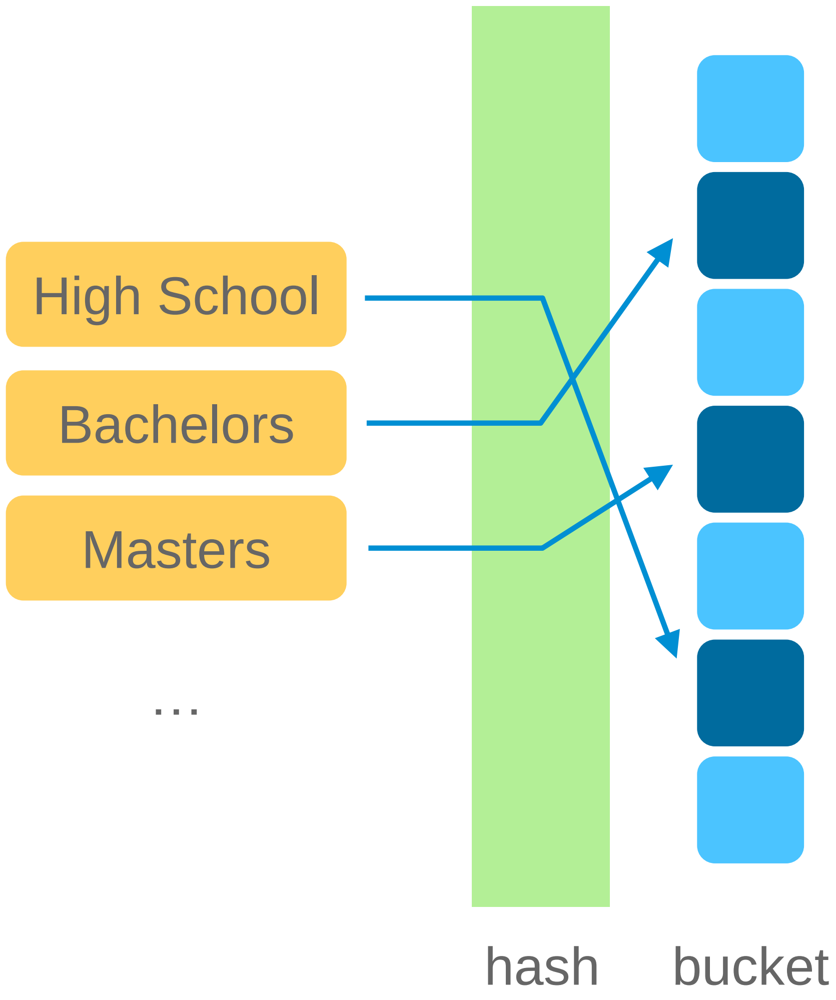
Feature Engineering

To save you time, but also memory with large vocabs



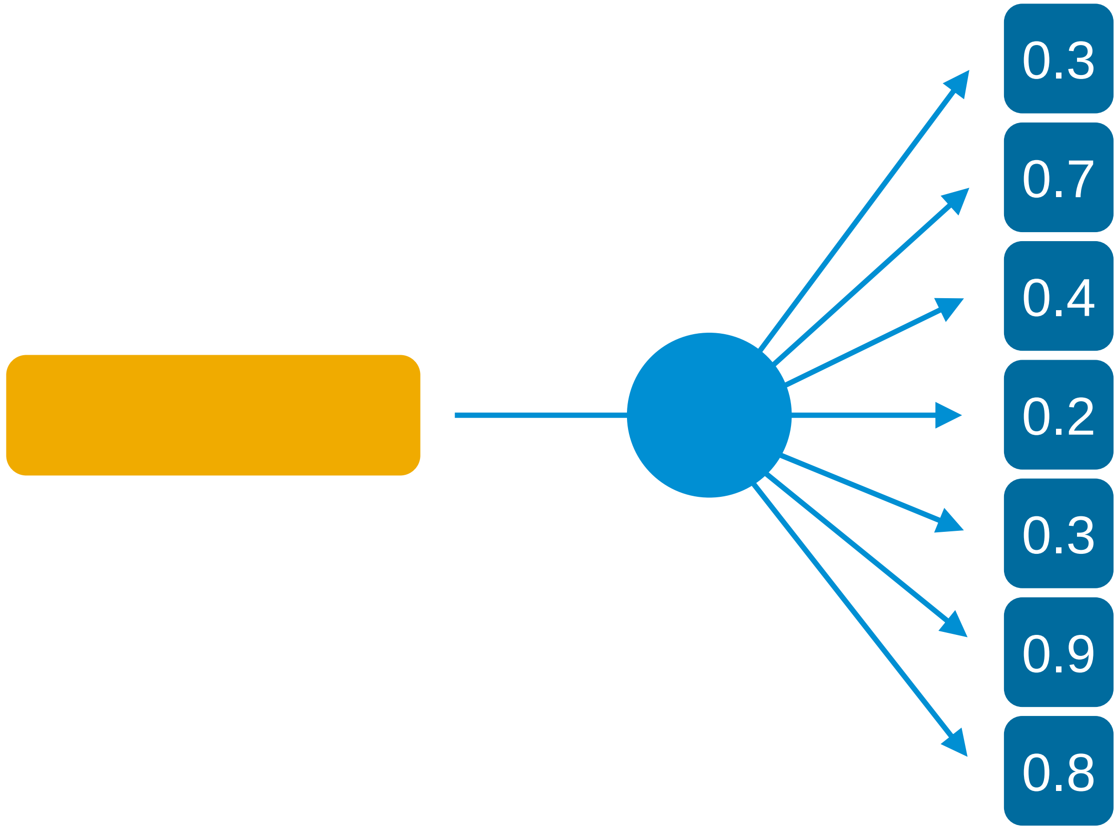
Feature Engineering

Hashing



Feature Engineering

Embeddings



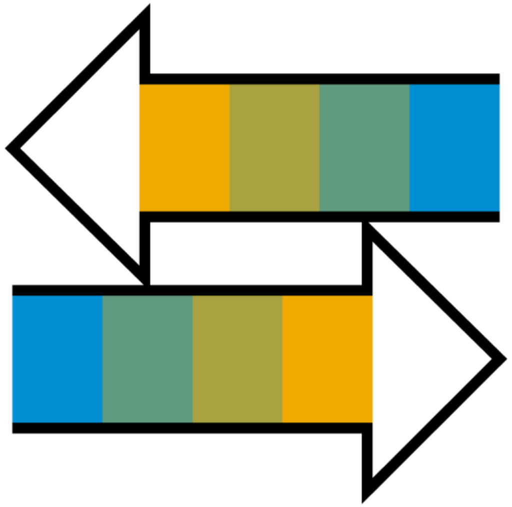
Feature Engineering
Pros and cons

Pros

- Can be memory-efficient with large vocabs
- Can lead to higher accuracy

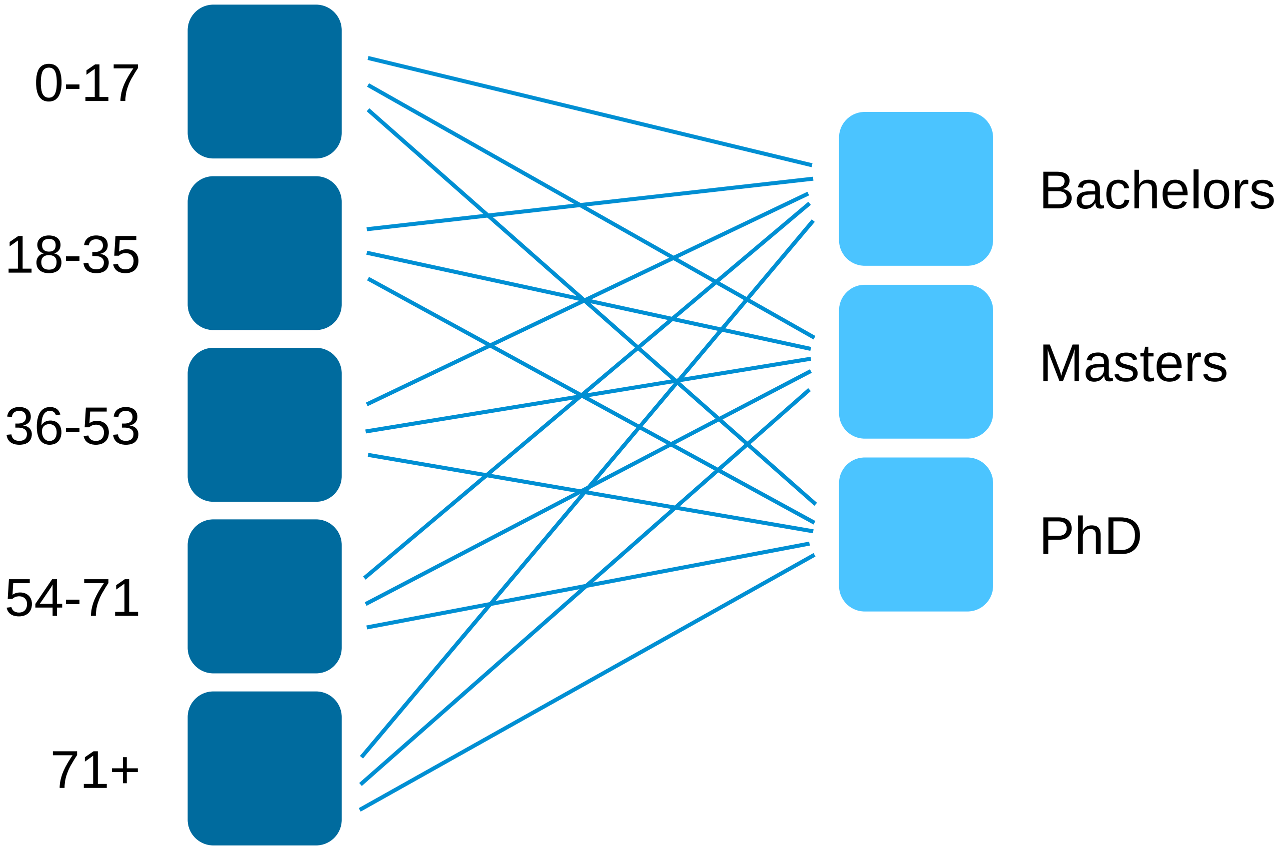
Cons

- More parameters to tune
- Lose interpretability vs one-hot encodings



Feature Engineering

Feature crosses

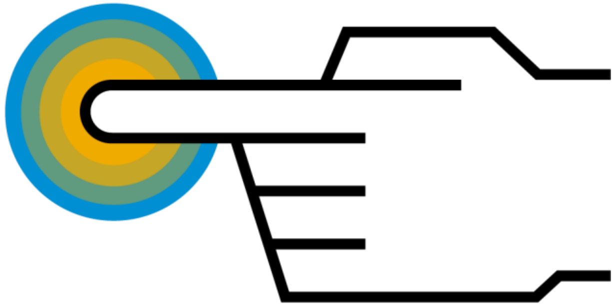


For each cross we create a new true/false feature

Feature Engineering

New features created

- 0-17-Bachelors (true/false)
- 0-17-Masters (true/false)
- 0-17-PhD (true/false)
- ...

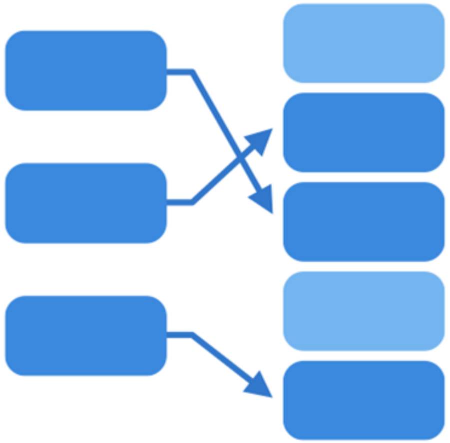


Feature Engineering

Feature engineering

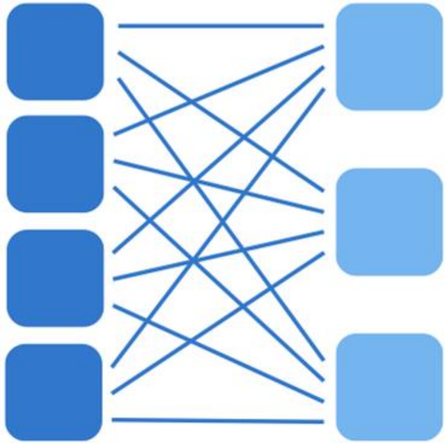
Bucketing

Hashing



Embedding

Crossing



Thank you.

Contact information: Josh Gordon

Twitter: @random_forests

Week 2: Building TensorFlow Applications

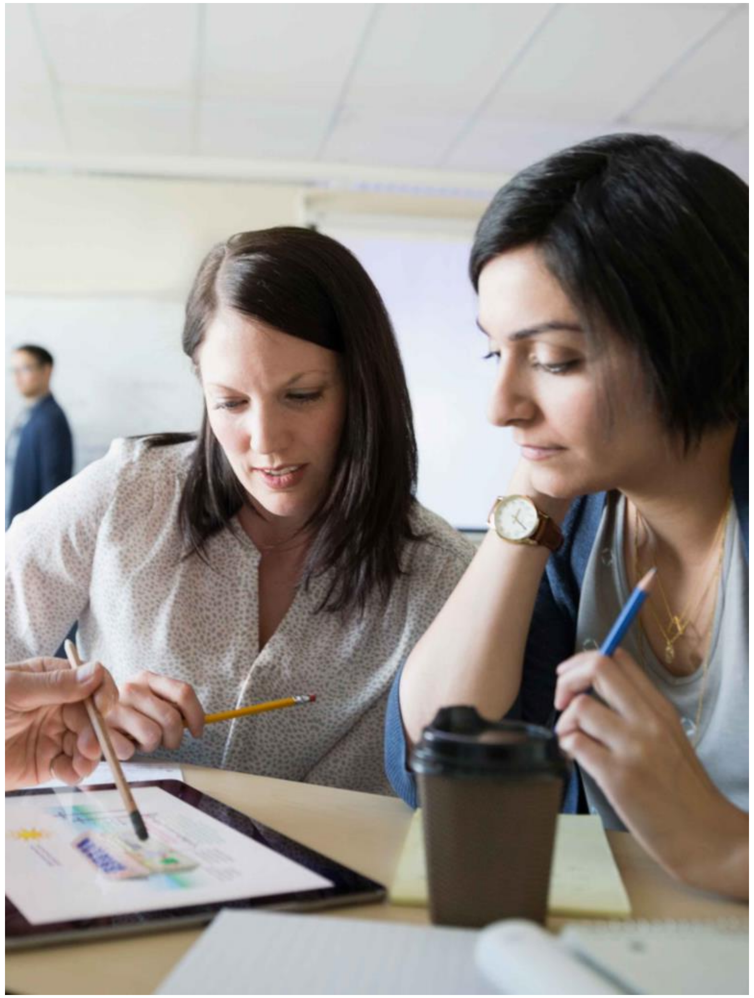
Unit 5: Architectures for Deep Learning

Architectures for Deep Learning

What we covered in the last units

Classifying Structure Data with TensorFlow

- Visualizing a dataset
- Canned estimators
- Feature engineering



Architectures for Deep Learning

Network architectures

Input-output networks

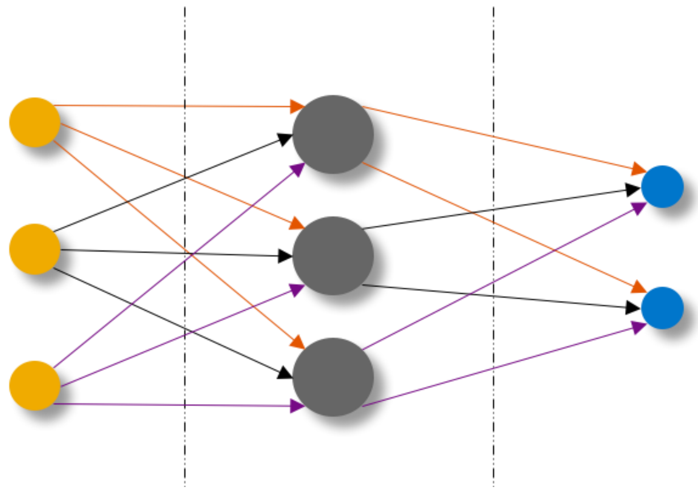
- Fixed-length input and output
- Example: feed-forward networks

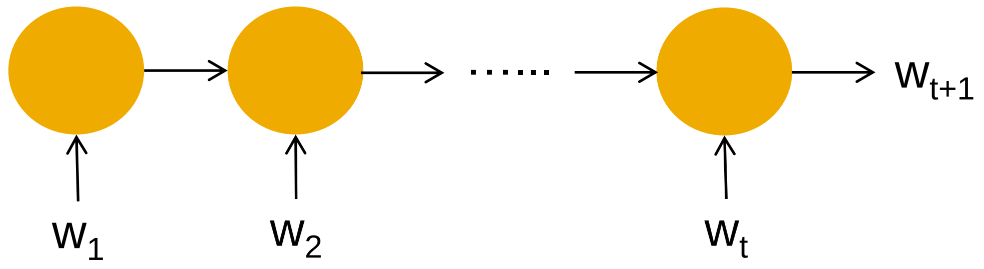
Sequence models

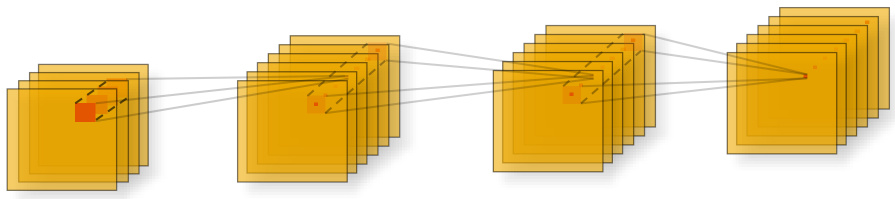
- Variable-length input
- Example: language models

Convolutional networks

- Local filters
- Example: computer vision







Architectures for Deep Learning

TensorFlow architecture

Model lifecycle

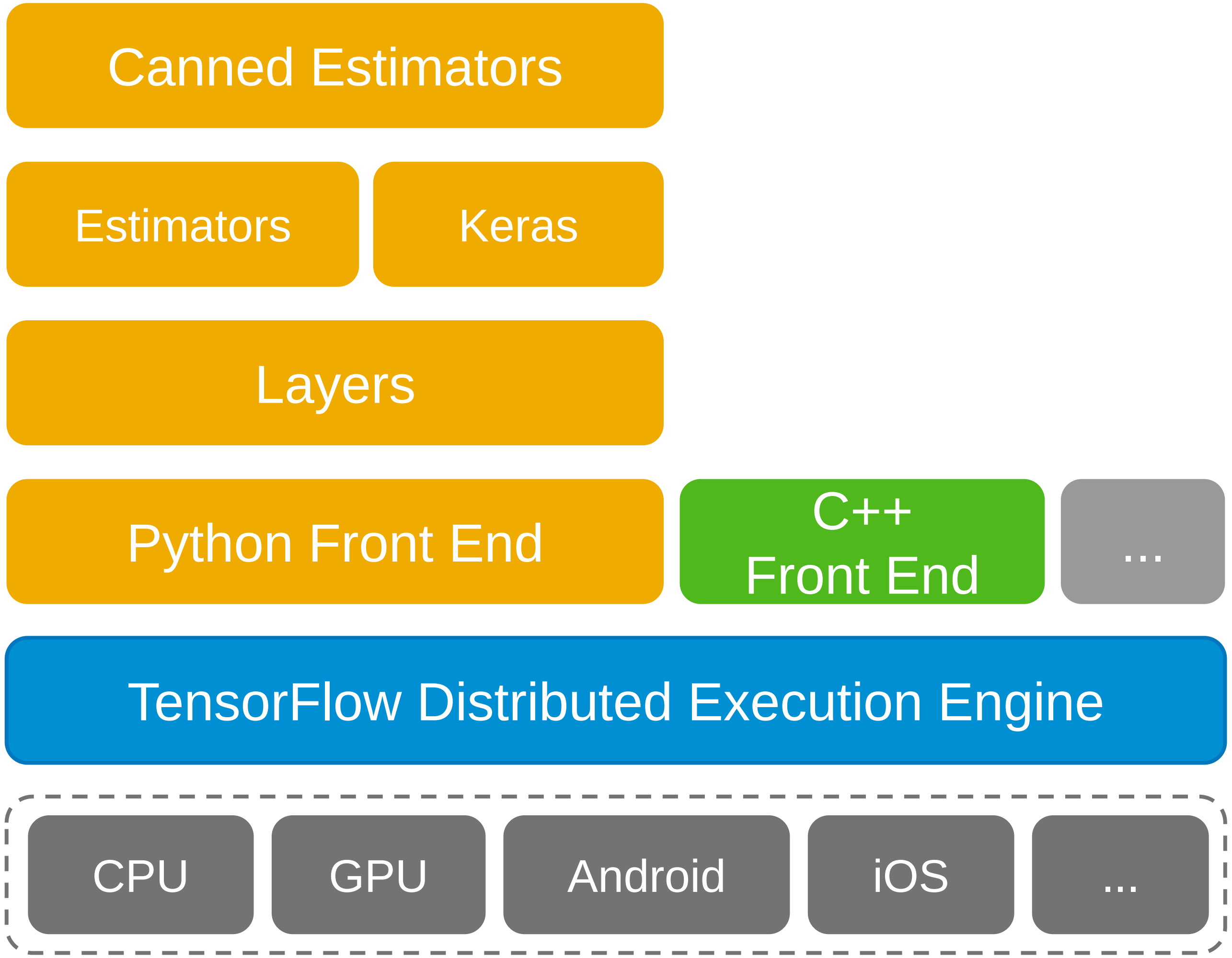
- Train models vs. inference

Hardware

- Graphics processing units (GPUs)
- Tensor processing units (TPUs)

TensorFlow

- Device-agnostic execution framework
- Multiple language front ends and higher-level APIs for application developers



Architectures for Deep Learning

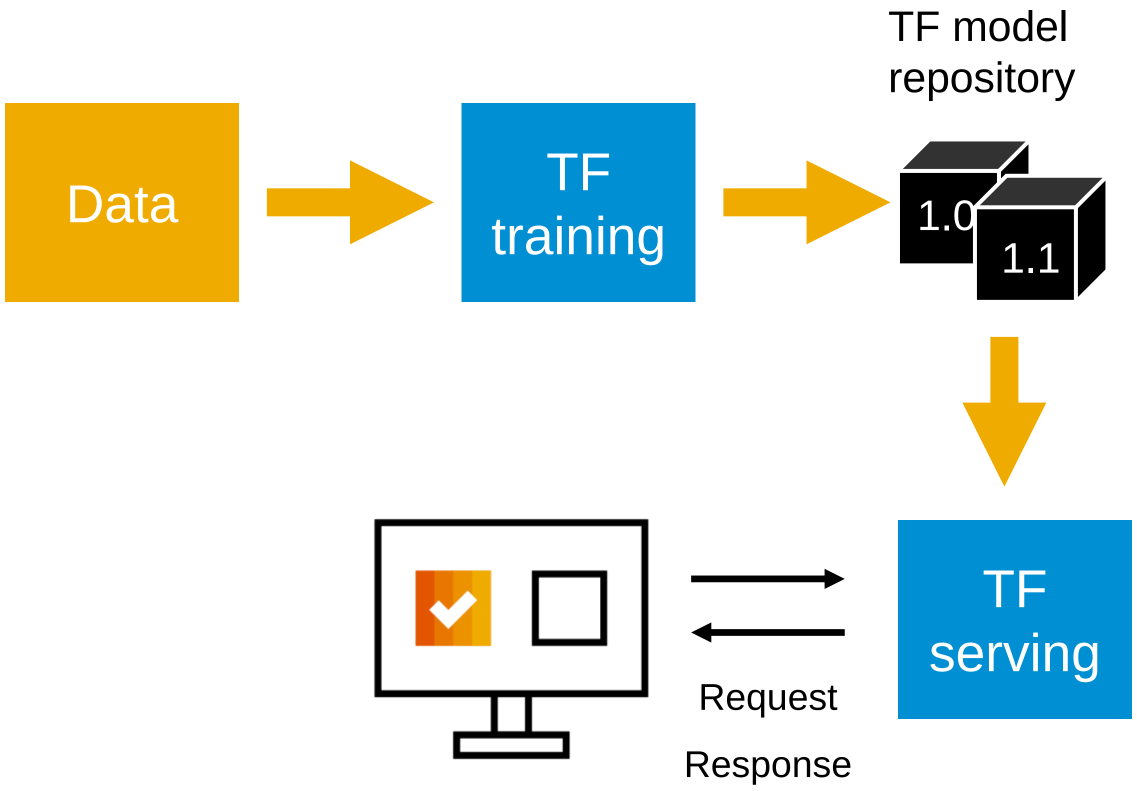
TensorFlow serving

What is TensorFlow serving?

- Flexible, high-performance serving system for machine learning models

Why TensorFlow serving?

- Manage model lifecycle (multiple models, multiple versions)
- Serve inference requests



Architectures for Deep Learning

SAP Leonardo Machine Learning Foundation

Applications

SAP Leonardo Machine Learning Foundation

Inference
- Ready-to-use services
- Bring your own model

Training
- Customize model
- Create training

SAP Cloud Platform

Use machine learning services

- Consume prediction from pre-trained models
- No training

Customize model

- Re-train existing models with own data

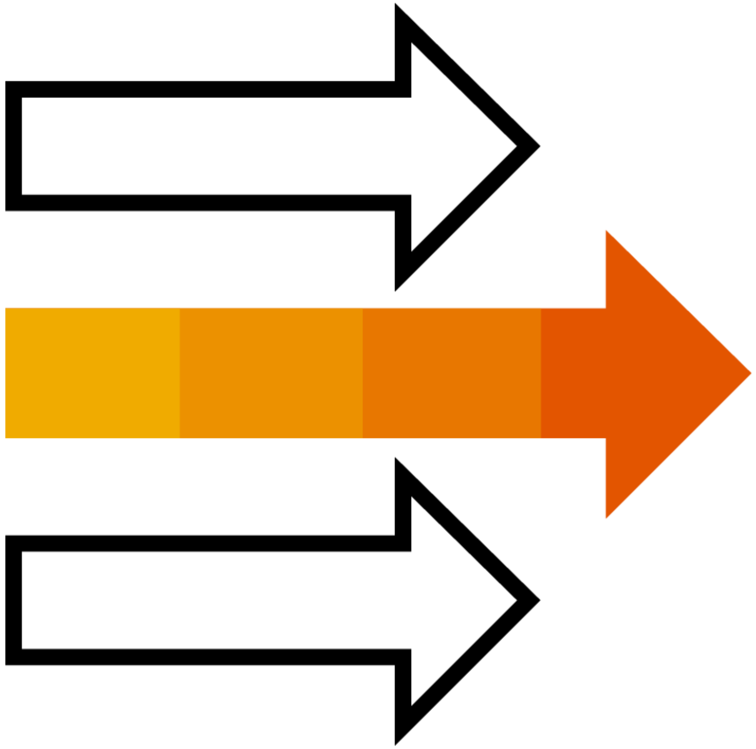
Build your own model

- Train offline and deploy to cloud services
- Train and deploy on cloud

Architectures for Deep Learning

Coming up next

Deep Networks and Sequence Models



Thank you.

Contact information:

open@sap.com