Deep Learning Tutorial

Luiz Gustavo Hafemann
LIVIA
École de Technologie Supérieure – Montréal

Agenda for the day 1

- **Introduction to Machine Learning**
- **Simbolic Math with Theano**

Introduction to Machine Learning

Introduction to Machine Learning

Learning functions from data

Introduction to Machine Learning

Learning functions from data

- **Supervised Learning**
  - Classification: SPAM classification, object recognition
    - Categorical output: $y = f(\mathbf{x})$ $y \in \mathcal{Y}$

Introduction to Machine Learning

Learning functions from data

- **Supervised Learning**
  - Classification: SPAM classification, object recognition
    - Categorical output: $y = f(\mathbf{x})$ $y \in \mathcal{Y}$
  - Regression: House pricing, forecasting
    - Real output: $y = f(\mathbf{x})$ $y \in \mathbb{R}$

Introduction to Machine Learning

Learning functions from data

- **Supervised Learning**
  - Classification: SPAM classification, object recognition
    - Categorical output: $y = f(\mathbf{x})$ $y \in \mathcal{Y}$
  - Regression: House pricing, forecasting
    - Real output: $y = f(\mathbf{x})$ $y \in \mathbb{R}$

- **Unsupervised Learning**
  - Density estimation, clustering, anomaly detection

Supervised learning - classification

Supervised learning - classification

Problem formulation:

Supervised learning - classification

Problem formulation:

Given a set of examples $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$, where

Supervised learning - classification

Problem formulation:

Given a set of examples $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$**, where**

$\mathbf{x} = \{x_1, x_2, ..x_n\}$ **are measurements of the input**

Supervised learning - classification

Problem formulation:

Given a set of examples $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$, where

$\mathbf{x} = \{x_1, x_2, ..x_n\}$ **are measurements of the input**

y **is the correct class**

Supervised learning - classification

Problem formulation:

Given a set of examples $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$, where

$\mathbf{x} = \{x_1, x_2, ..x_n\}$ **are measurements of the input**

y **is the correct class**

The objective is to learn a mapping (function) from x to y:

$$y_{\text{pred}} = f(x)$$

That generalizes to unseen examples

Supervised learning - classification

Toy example:

2-class problem: classify fish between bass and salmon

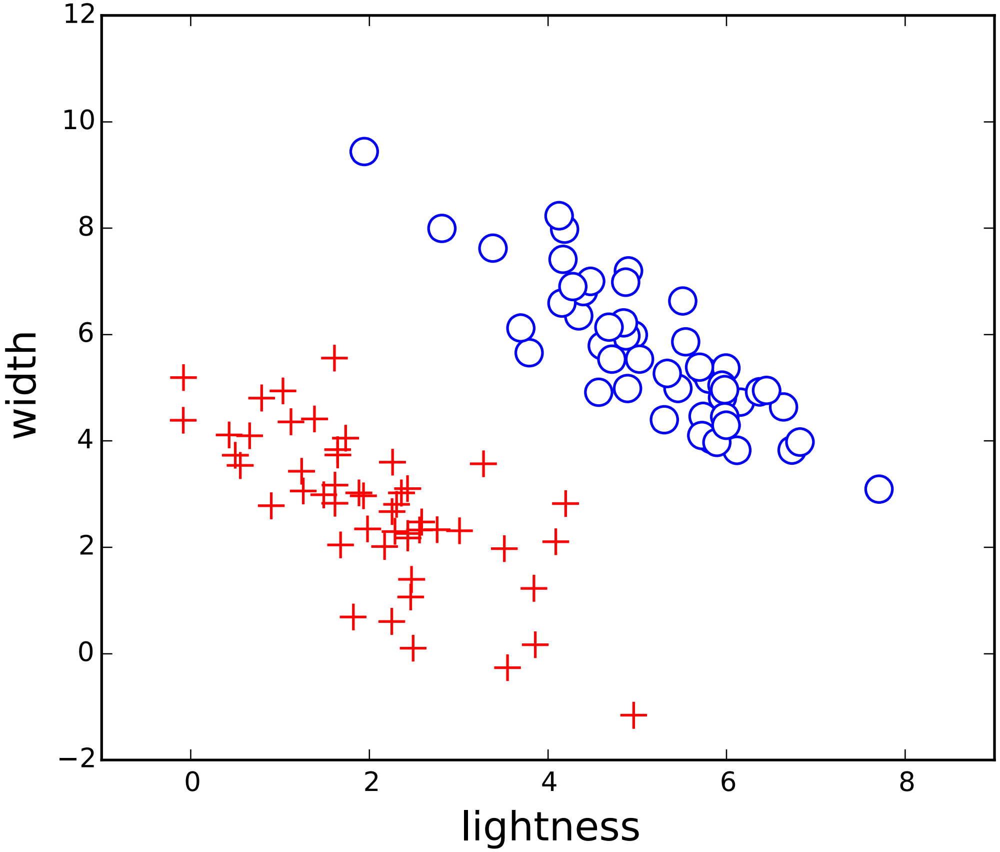
$$y \in \{\text{bass}, \text{salmon}\}$$

Two measurements: width and lightness

$$\mathbf{x} = \{x_1, x_2\}$$

Supervised Learning – toy example

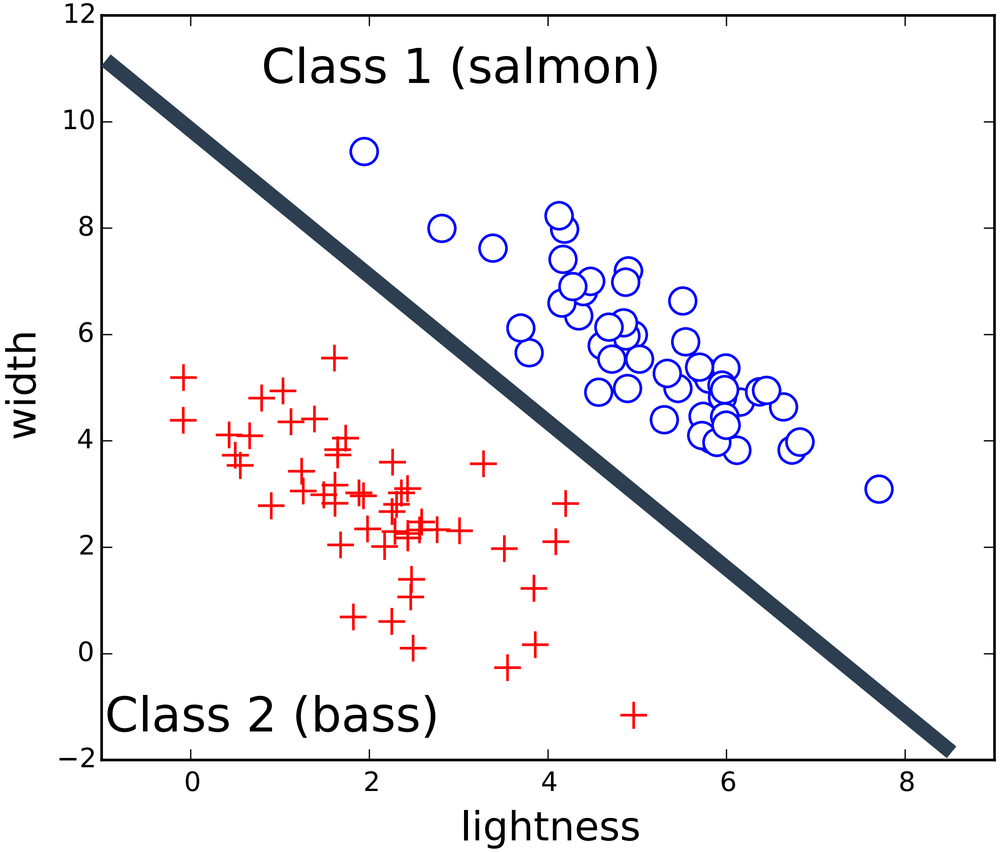
Acquired 50 examples from each class:



Supervised Learning – toy example

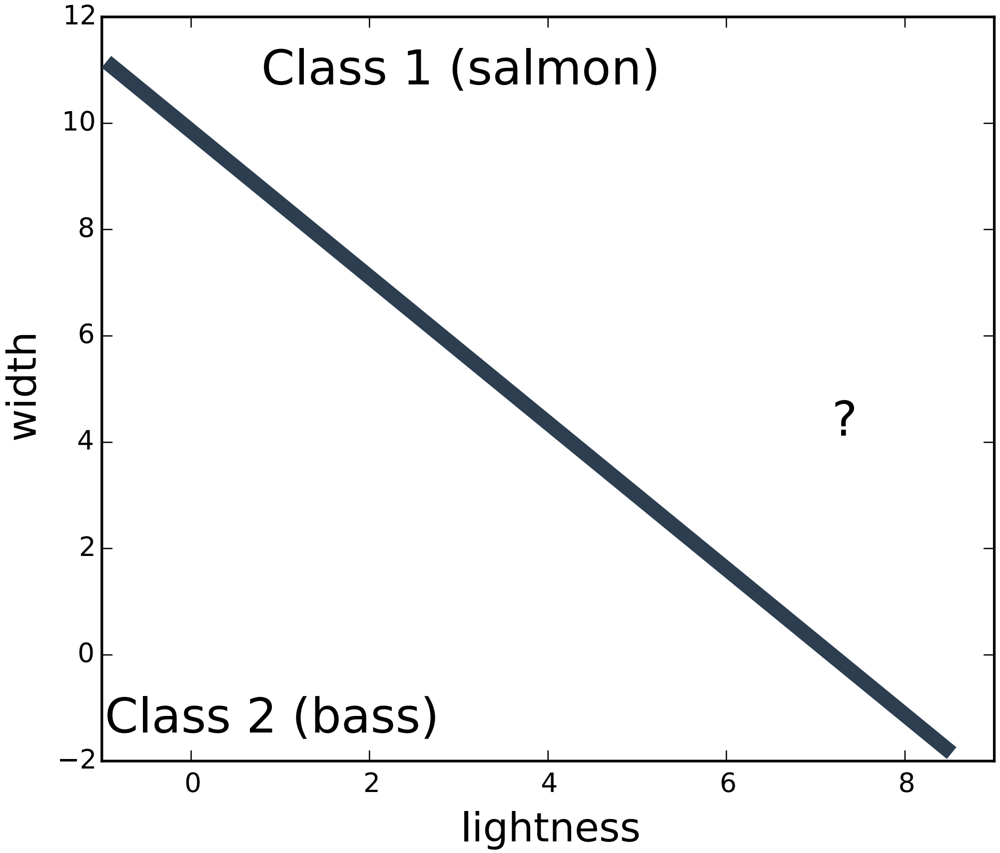
Learn a model

(in this case: a parametric linear model)

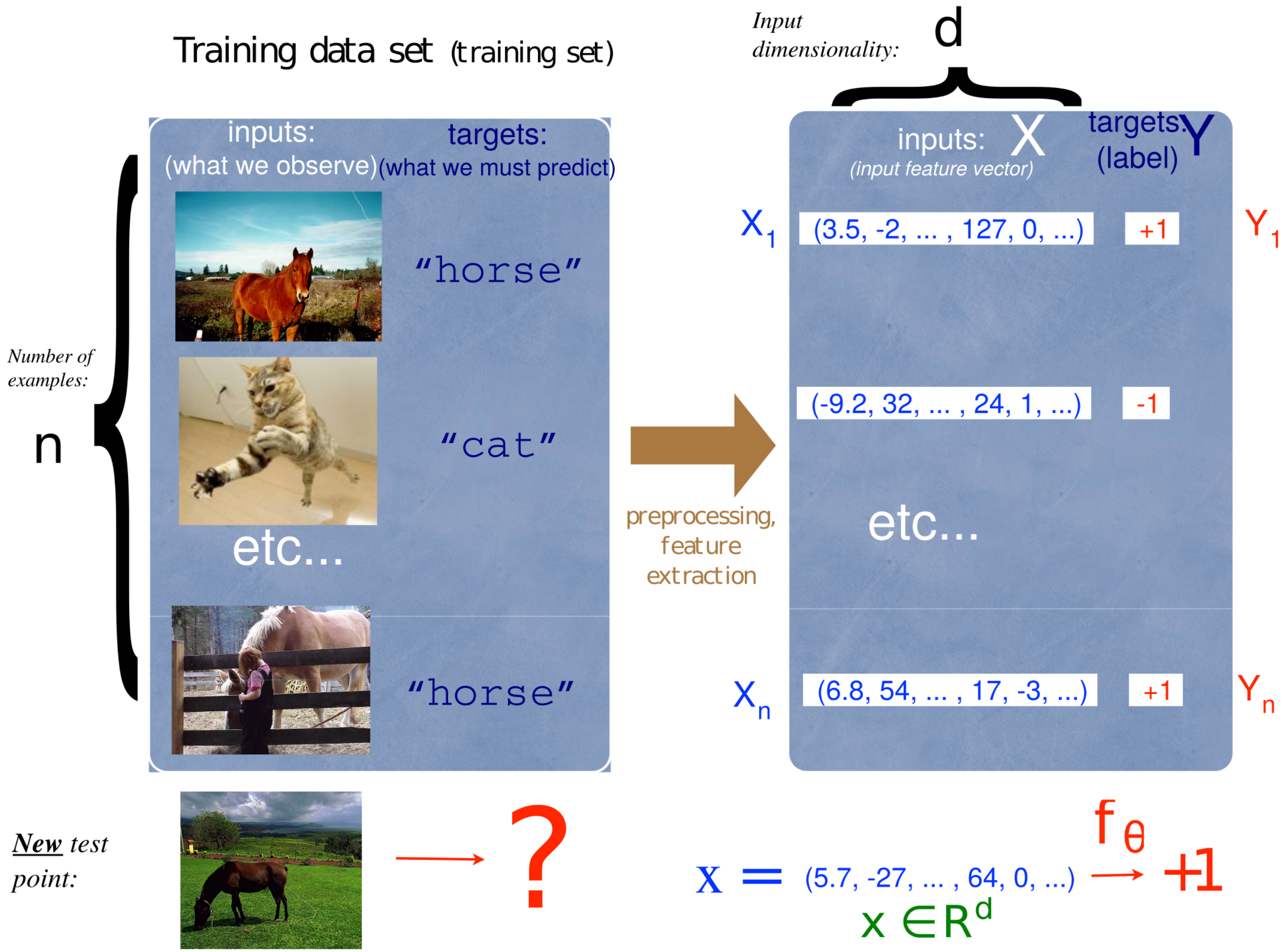


Supervised Learning – toy example

Generalize to new examples



Supervised Learning



Slide from
Pascal Vincent

3 basic elements for Machine Learning

3 basic elements for Machine Learning

Choose a function family

Usually a parametric family (e.g. “Logistic Regression”)

3 basic elements for Machine Learning

Choose a function family

Usually a parametric family (e.g. “Logistic Regression”)

A way to evaluate the quality of f

Loss function → the lower, the better is the model

3 basic elements for Machine Learning

Choose a function family

Usually a parametric family (e.g. “Logistic Regression”)

A way to evaluate the quality of f

Loss function → the lower, the better is the model

A way to search for the best f

Optimization procedure → how to change the parameters to get a lower loss

Logistic Regression

Logistic Regression

Function family:

linear: $w_1 x_1 + w_2 x_2 ... w_m x_m$

Use a non-linear function to get results between [0,1]

$$P(y|x) = \sigma(\mathbf{w}^\intercal \mathbf{x})$$

Logistic Regression

Function family:

linear: $w_1 x_1 + w_2 x_2 ... w_m x_m$

Use a non-linear function to get results between [0,1]

$$P(y|x) = \sigma(\mathbf{w}^\intercal \mathbf{x})$$

Objective:

Maximize $P(y|x)$ for the examples in the dataset.

Equivalent to minimize: $-\sum \log P(y|x)$

Logistic Regression

Function family:

linear: $w_1 x_1 + w_2 x_2 ... w_m x_m$

Use a non-linear function to get results between [0,1]

$$P(y|x) = \sigma(\mathbf{w}^\intercal \mathbf{x})$$

Objective:

Maximize $P(y|x)$ for the examples in the dataset.

Equivalent to minimize: $-\sum \log P(y|x)$

Optimization:

Gradient descent: Start with random w, do small steps that reduce the loss

Logistic Regression

Logistic Regression

Loss function:

$$L = -\frac{1}{N}\sum \log P(y|x)$$

$$L = -\tfrac{1}{N}\sum_{i=1}^{N} y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})$$

Logistic Regression

Loss function:

$$L = -\frac{1}{N} \sum \log P(y|x)$$

$$L = -\frac{1}{N} \sum_{i=1}^{N} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Optimization:

Start: $\mathbf{w}^{(0)} = \text{random}$

For T iterations:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla_{\mathbf{w}} L$$

Logistic Regression

Loss function:

$$L = -\frac{1}{N} \sum \log P(y|x)$$

$$L = -\frac{1}{N} \sum_{i=1}^{N} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Optimization:

Start: $\mathbf{w}^{(0)} = \text{random}$

For T iterations:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \nabla_{\mathbf{w}} L$$

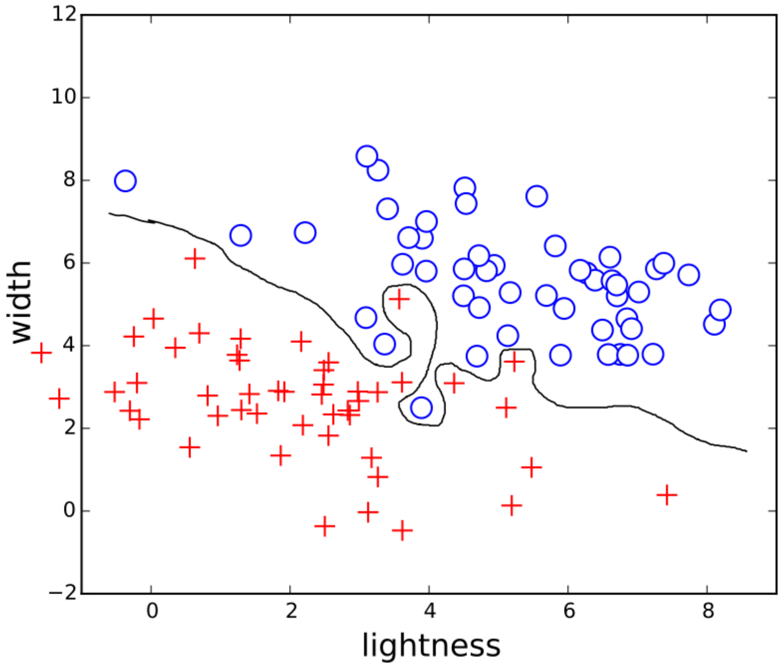
Calculating $\nabla_{\mathbf{w}} L$:

Use chain rule. Or use software that does it for you (e.g. Theano)

Overfitting

What matters is generalization error, but we minimize training error

If our model is too complex, it may overfit



Overfitting

Overfitting

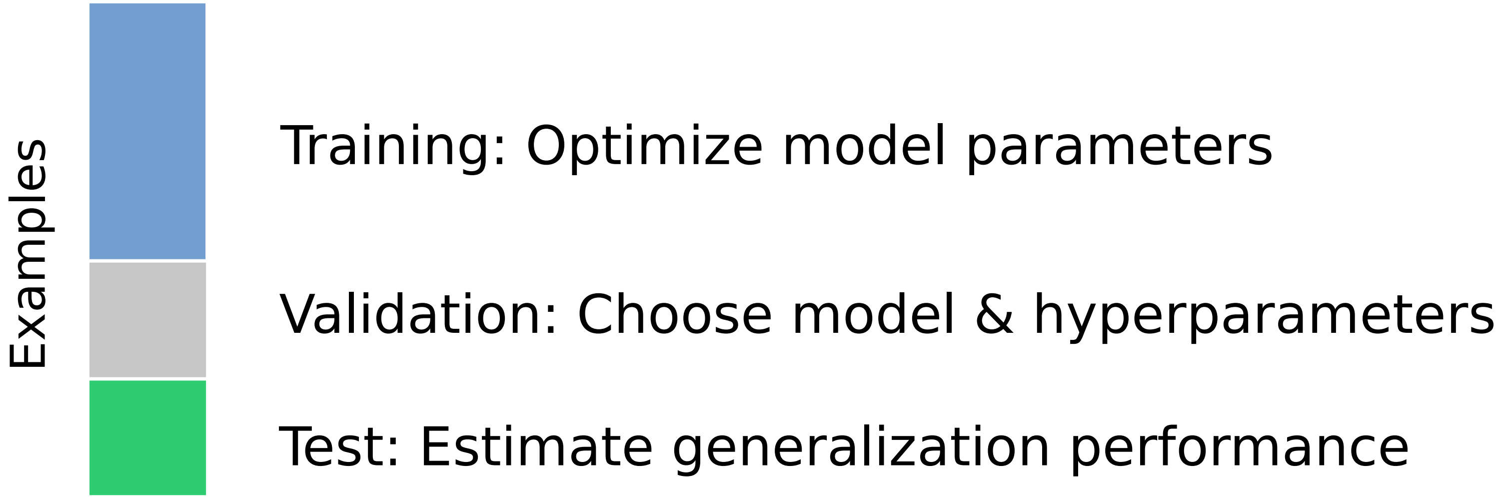
- **Estimate the generalization error**
  - Keep separate test set to estimate generalization error (need to be disjoint from training set not to be biased)

Overfitting

- **Estimate the generalization error**
  - Keep separate test set to estimate generalization error (need to be disjoint from training set not to be biased)
  - To choose hyperparameters of the model (e.g. choice of model, feature extractors, etc.) use yet another disjoint set

Overfitting

- **Estimate the generalization error**
  - Keep separate test set to estimate generalization error (need to be disjoint from training set not to be biased)
  - To choose hyperparameters of the model (e.g. choice of model, feature extractors, etc.) use yet another disjoint set

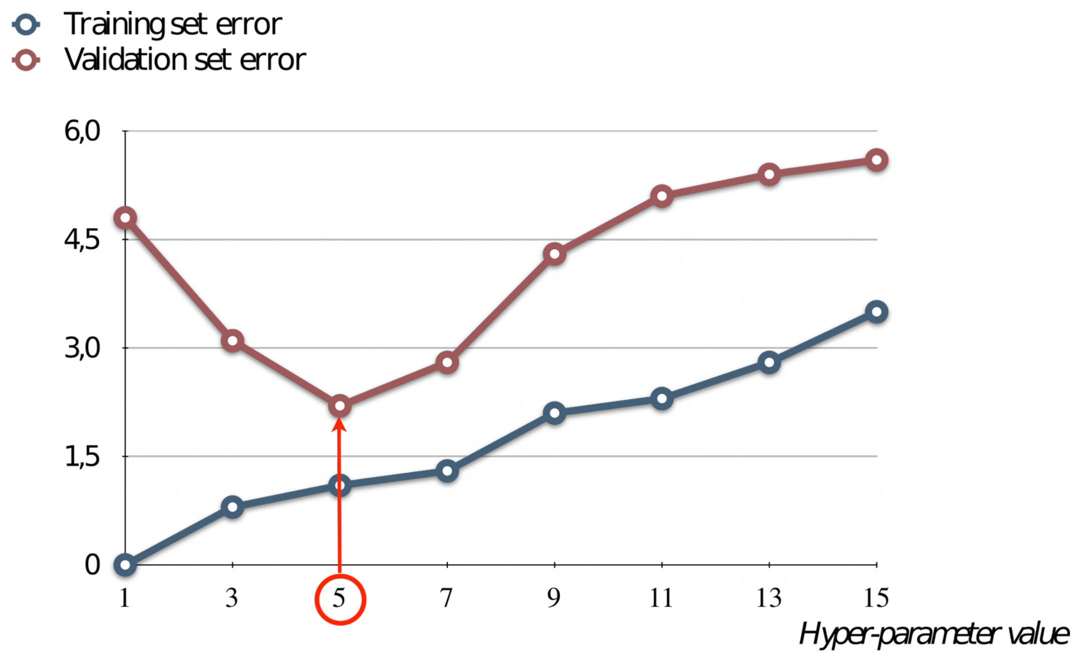


Model selection and overfitting

Train different models on the training set

Evaluate performance on validation. Pick best model

Test model performance on test set

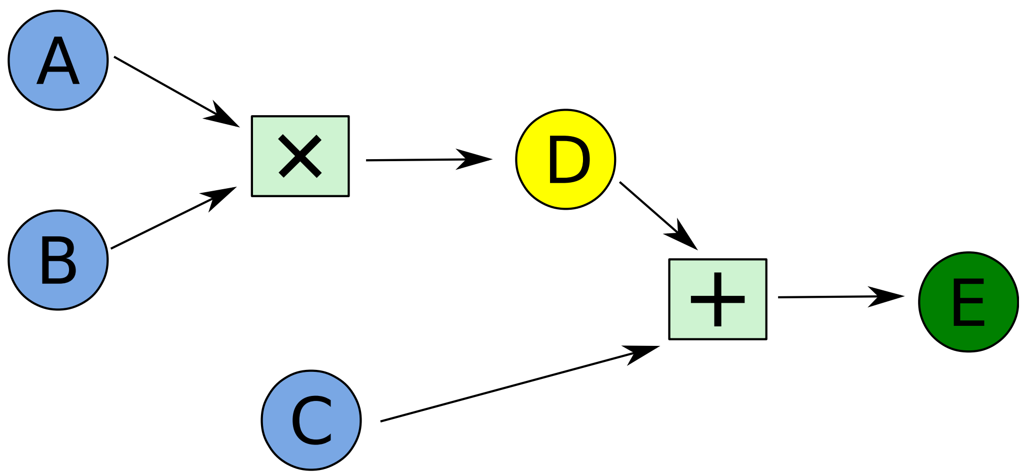


Hyper-parameter value which yields smallest error on validaiton set is 5
(it was 1 for the training set)

Introduction to Theano

Symbolic Computation

Expressions are defined as a graph



Blue: inputs

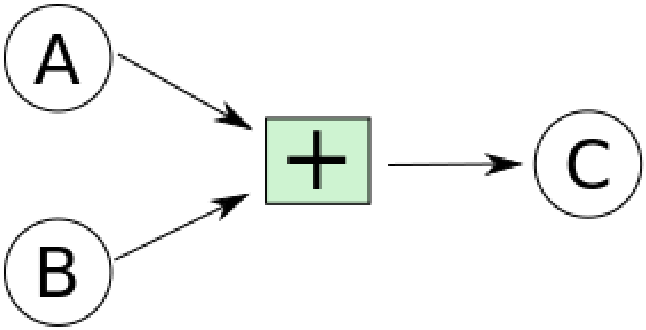
Yellow: intermediate nodes

Green: output

Introduction to Theano

Expressions need to be compiled

```
a = T.scalar()
b = T.scalar()
c = 3*a + 2*b
f = theano.function([a,b],c)
f(2,2) #returns 10
```



Enable automatic differentiation:

```
df_da = T.grad(c, a)
g = theano.function([a,b], df_da)
g(2,2) # returns 3
```

Ipython Notebook

DEMO