Recitation 1
Machine Learning Overview

Sid Jain   Konstantin Krismer   Ge (Saber) Liu

MIT - 6.802 / 6.874 / 20.390 / 20.490 / HST.506 - Spring 2019

2019-02-07 / 2019-02-08

What is Machine Learning?

[Shalev-Shwartz and Ben-David, 2014]:

> *"Learning is the process of converting experience into expertise or knowledge."*

What is Machine Learning?

[Shalev-Shwartz and Ben-David, 2014]:

> *"Learning is the process of converting experience into expertise or knowledge."*

[Mohri et al., 2012]:

> *"Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions."*

What is Machine Learning?

[Shalev-Shwartz and Ben-David, 2014]:

> *"Learning is the process of converting experience into expertise or knowledge."*

[Mohri et al., 2012]:

> *"Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions."*

[Murphy, 2012]:

> *"The goal of machine learning is to develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest."*

What is Machine Learning?

[Shalev-Shwartz and Ben-David, 2014]:

> *"Learning is the process of converting experience into expertise or knowledge."*

[Mohri et al., 2012]:

> *"Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions."*

[Murphy, 2012]:

> *"The goal of machine learning is to develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest."*

[Hastie et al., 2001]:

> *"[...] state the learning task as follows: given the value of an input vector* $\boldsymbol{x}$*, make a good prediction of the output* $\boldsymbol{y}$*, denoted by* $\hat{\boldsymbol{y}}$*"*

What is Machine Learning?

A computer program is said to learn from
experience E

with respect to some
class of tasks T

and
performance measure P,

if its performance at tasks in T, as measured by P, improves with experience E.

[Mitchell, 1997]

What is Machine Learning?

A computer program is said to learn from
experience E

with respect to some
class of tasks T

and
performance measure P,

if its performance at tasks in T, as measured by P, improves with experience E.

[Mitchell, 1997]

Problem Set 1

- experience E: training set of images of handwritten digits with labels
- task T: classifying handwritten digits within images
- performance measure P: percent of test set digits correctly classified

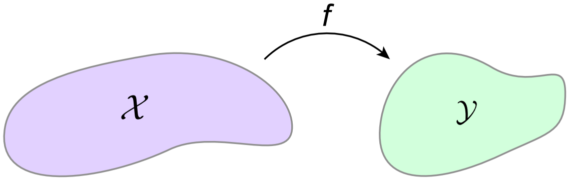
Notation

a, b, c_i	scalar (slanted, lower-case)
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	vector (bold, slanted, lower-case)
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	matrix (bold, slanted, upper-case)
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	tensor (bold, upright, upper-case)
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	set (calligraphic, slanted, upper-case)

Notation

a, b, c_i	scalar (slanted, lower-case)
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	vector (bold, slanted, lower-case)
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	matrix (bold, slanted, upper-case)
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	tensor (bold, upright, upper-case)
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	set (calligraphic, slanted, upper-case)
$\mathcal{X}$	input space or feature space
$\boldsymbol{X}, \mathbf{X}$	dataset example matrix or tensor
$\boldsymbol{x}^{(i)}$	ith example of dataset, one row of $\boldsymbol{X}$
$x_j^{(i)}, x_j$	feature j of example $\boldsymbol{x}^{(i)}$
$\mathcal{Y}$	label space
$\boldsymbol{y}^{(i)}$	label of example i
$\hat{\boldsymbol{y}}^{(i)}$	predicted label of example i

Terminology



Input $\boldsymbol{X} \in \mathcal{X}$:

- **features** (in machine learning)
- predictors (in statistics)
- independent variables (in statistics)
- regressors (in regression models)
- input variables
- covariates

Output $\boldsymbol{y} \in \mathcal{Y}$:

- **labels** (in machine learning)
- responses (in statistics)
- dependent variables (in statistics)
- regressand (in regression models)
- target variables

Training set $\mathcal{S}_{\text{training}} = \{(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N} \in \{\mathcal{X}, \mathcal{Y}\}^N$, where N is number of training examples

An example is a collection of features (and an associated label)

Training: use $\mathcal{S}_{\text{training}}$ to learn functional relationship $f \colon \mathcal{X} \to \mathcal{Y}$

Terminology

$$f: \quad \mathcal{X} \to \mathcal{Y}$$
$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \hat{\boldsymbol{y}}$$

$\boldsymbol{\theta}$:

- **weights** and **biases** (intercepts)
- coefficients $\boldsymbol{\beta}$
- parameters

f:

- model
- hypothesis h
- classifier
- predictor
- discriminative models: P(Y|X)
- generative models: P(X, Y)

Problem Set 1

$$\boldsymbol{x} \in [0,1]^{784}$$
$$\hat{\boldsymbol{y}} \in [0,1]^{10}$$
$$\boldsymbol{W} \in \mathbb{R}^{784 \times 10}$$
$$\boldsymbol{b} \in \mathbb{R}^{10}$$
$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \phi_{\text{softmax}}(\boldsymbol{W}^\intercal \boldsymbol{x} + \boldsymbol{b})$$

Data in PS1

Problem Set 1

input space:
$\mathcal{X} = \{0, 1, \ldots, 255\}^{28\times 28}$

after rescaling:
$\mathcal{X}' = [0, 1]^{28\times 28}$

after flattening:
$\mathcal{X}'' = [0, 1]^{784}$

$$\overbrace{\begin{array}{c|cccc|} & 1 & 2 & \cdots & 28 \\ 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,28} \\ 2 & x_{2,1} & x_{2,2} & \cdots & x_{2,28} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 28 & x_{28,1} & x_{28,2} & \cdots & x_{28,28} \end{array}}^{\boldsymbol{x}^{(i)} \in \mathcal{X}}$$

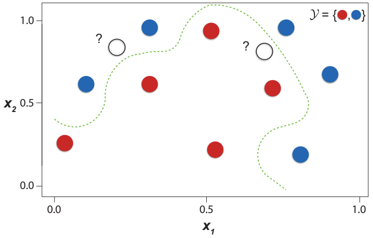
integer-encoded label space:
$\mathcal{Y}_\text{i} = \{0, 1, \ldots, 9\}$

one-hot-encoded label space:
$\mathcal{Y}_\text{h} = [0, 1]^{10}$

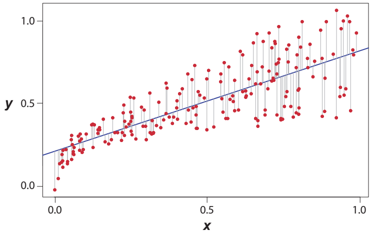
$$\overbrace{\begin{array}{cccc} 1 & 2 & \cdots & 10 \\ \hline y_1 & y_2 & \cdots & y_{10} \end{array}}^{\boldsymbol{y}^{(i)} \in \mathcal{Y}_h}$$

Types of Machine Learning

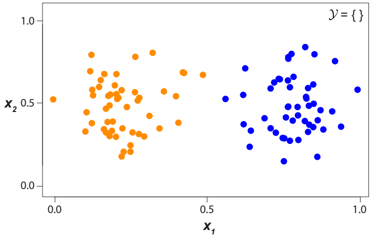
Classification



Regression



Unsupervised learning



$\mathcal{Y} \neq \emptyset$	supervised or semi-supervised learning
$\mathcal{Y} = \mathbb{R}$	regression
$\mathcal{Y} = \mathbb{R}^K, K > 1$	multivariate regression
$\mathcal{Y} = \{0, 1\}$	binary classification
$\mathcal{Y} = \{1, ..., K\}$	multi-class classification (integer encoding)
$\mathcal{Y} = \{0, 1\}^K, K > 1$	multi-label classification
$\mathcal{Y} = \emptyset$	unsupervised learning

Types of Machine Learning

Problem Set 1

- task: every $\boldsymbol{X}$ has an associated $\boldsymbol{y} \implies$ supervised learning
- subtask: $\mathcal{Y} = \{0, \ldots, 9\} \implies$ multi-class classification
- method: we use softmax regression (also known as multinomial logistic regression) as multi-class classification method

Objective functions

An **objective function** $\mathcal{J}(\Theta)$ is the function that you optimize when training machine learning models. It is usually in the form of (but not limited to) one or combinations of the following:

Loss / cost / error function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$:

Classification

- 0-1 loss
- cross-entropy loss
- hinge loss

Regression

- mean squared error (MSE, L_2 norm)
- mean absolute error (MAE, L_1 norm)
- Huber loss (hybrid between L_1 and L_2 norm)

Probabilistic inference

- Kullback-Leibler divergence (KL divergence)

Likelihood function / posterior:

- negative log-likelihood (NLL) in maximum likelihood estimation (MLE)
- posterior in maximum a posteriori estimation (MAP)

Regularizers and constraints

- L_1 regularization $||\Theta||_1$
- L_2 regularization $||\Theta||_2^2$
- max-norm

Loss functions for classification

0-1 loss:

$$\mathcal{L}_{0\text{-}1}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{N} \mathbb{1}([\hat{y}^{(i)}] \neq y^{(i)}) = \sum_{i=1}^{N} \begin{cases} 1, & \text{for } \hat{y}^{(i)} \neq y^{(i)} \\ 0, & \text{for } \hat{y}^{(i)} = y^{(i)} \end{cases}$$

where $[x]$ is the function that rounds x to the nearest integer.

Loss functions for classification

0-1 loss:

$$\mathcal{L}_{0\text{-}1}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{N} \mathbb{1}([\hat{y}^{(i)}] \neq y^{(i)}) = \sum_{i=1}^{N} \begin{cases} 1, & \text{for } \hat{y}^{(i)} \neq y^{(i)} \\ 0, & \text{for } \hat{y}^{(i)} = y^{(i)} \end{cases}$$

where $[x]$ is the function that rounds x to the nearest integer.

Binary cross-entropy loss (for binary classification):

$$\begin{aligned} \mathcal{L}_{\text{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) &= \sum_{i=1}^{N} -y^{(i)} \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \\ &= \sum_{i=1}^{N} \begin{cases} -\log(\hat{y}^{(i)}), & \text{for } y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}), & \text{for } y^{(i)} = 0 \end{cases} \end{aligned}$$

Probabilistic interpretation:
$\mathcal{L}_{\text{BCE}} = \text{NLL}$, if likelihood is defined using the Bernoulli distribution

Loss functions for classification

0-1 loss:

$$\mathcal{L}_{0\text{-}1}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{N} \mathbb{1}([\hat{y}^{(i)}] \neq y^{(i)}) = \sum_{i=1}^{N} \begin{cases} 1, & \text{for } \hat{y}^{(i)} \neq y^{(i)} \\ 0, & \text{for } \hat{y}^{(i)} = y^{(i)} \end{cases}$$

where $[x]$ is the function that rounds x to the nearest integer.

Binary cross-entropy loss (for binary classification):

$$\begin{aligned} \mathcal{L}_{\text{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) &= \sum_{i=1}^{N} -y^{(i)} \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \\ &= \sum_{i=1}^{N} \begin{cases} -\log(\hat{y}^{(i)}), & \text{for } y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}), & \text{for } y^{(i)} = 0 \end{cases} \end{aligned}$$

Probabilistic interpretation:
$\mathcal{L}_{\text{BCE}} =$ NLL, if likelihood is defined using the Bernoulli distribution

$\boldsymbol{y}$	$\hat{\boldsymbol{y}}$	$[\hat{\boldsymbol{y}}]$	$\mathcal{L}_{0\text{-}1}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	$\mathcal{L}_{\text{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$
$[1, 0, 0]$	$[0.9, 0.2, 0.4]$	$[1, 0, 0]$	0	0.84
$[1, 1, 0]$	$[0.6, 0.4, 0.1]$	$[1, 0, 0]$	1	1.53
$[1, 0, 1]$	$[0.1, 0.7, 0.3]$	$[0, 1, 0]$	3	4.71

Loss functions for classification

Problem Set 1

Categorical cross-entropy loss (for multi-class classification with K classes):

$$\mathcal{L}_{\mathrm{CCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{N} \sum_{j=1}^{K} y_j^{(i)} \log(\hat{y}_j^{(i)}),$$

$$\text{where} \quad \hat{y}_j^{(i)} = \frac{\exp(z_j^{(i)})}{\sum_{k=1}^{K} \exp(z_k^{(i)})} \quad \text{if softmax is used}$$

note: $y_j^{(i)} = 1$ only if $\boldsymbol{x}^{(i)}$ belongs to class j and otherwise $y_j^{(i)} = 0$

Probabilistic interpretation:
$\mathcal{L}_{\mathrm{CCE}} =$ NLL, if likelihood is defined using the categorical distribution

Loss functions for regression

Mean squared error:

$$\mathcal{L}_{\mathrm{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$$

Probabilistic interpretation:
$\mathcal{L}_{\mathrm{MSE}} = \mathrm{NLL}$, under the assumptation that the noise is normally distributed, with constant mean and variance

Loss functions for regression

Mean squared error:

$$\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

Probabilistic interpretation:
$\mathcal{L}_{\text{MSE}} = \text{NLL}$, under the assumptation that the noise is normally distributed, with constant mean and variance

Mean absolute error:

$$\mathcal{L}_{\text{MAE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - \hat{y}^{(i)}|$$

Loss functions for regression

Mean squared error:

$$\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

Probabilistic interpretation:
$\mathcal{L}_{\text{MSE}} =$ NLL, under the assumptation that the noise is normally distributed, with constant mean and variance

Mean absolute error:

$$\mathcal{L}_{\text{MAE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - \hat{y}^{(i)}|$$

$\boldsymbol{y}$	$\hat{\boldsymbol{y}}$	$\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	$\mathcal{L}_{\text{MAE}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$
$[3.2, 1.2, 0.3]$	$[3.1, 1.3, 0.4]$	0.01	0.1
$[2.1, 0.1, -5.1]$	$[2.0, -0.1, 1.2]$	13.25	2.2
$[-0.1, 3.1, 0.5]$	$[0.1, 3.3, -0.5]$	0.36	0.47

Empirical risk minimization

Expected risk (loss) associated with hypothesis $h(\boldsymbol{x})$:

$$\mathcal{R}_{\text{exp}}(h) = \mathbb{E}(\mathcal{L}(h(\boldsymbol{x}), \boldsymbol{y})) = \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(h(\boldsymbol{x}), \boldsymbol{y})dp(\boldsymbol{x}, \boldsymbol{y})$$

Minimize $\mathcal{R}_{\text{exp}}(h)$ to find optimal hypothesis h:

$$h = \operatorname*{argmin}_{h\in\mathcal{F}} \mathcal{R}_{\text{exp}}(h)$$

Empirical risk minimization

Expected risk (loss) associated with hypothesis $h(\boldsymbol{x})$:

$$\mathcal{R}_{\text{exp}}(h) = \mathbb{E}(\mathcal{L}(h(\boldsymbol{x}), \boldsymbol{y})) = \int_{\mathcal{X} \times \mathcal{Y}} \mathcal{L}(h(\boldsymbol{x}), \boldsymbol{y}) dp(\boldsymbol{x}, \boldsymbol{y})$$

Minimize $\mathcal{R}_{\text{exp}}(h)$ to find optimal hypothesis h:

$$h = \operatorname*{argmin}_{h \in \mathcal{F}} \mathcal{R}_{\text{exp}}(h)$$

Problem:

- distribution $p(\boldsymbol{x}, \boldsymbol{y})$ unknown
- $\mathcal{F}$ is too large (set of all functions from $\mathcal{X}$ to $\mathcal{Y}$)

Empirical risk minimization

Empirical risk associated with hypothesis $h(\boldsymbol{x})$:

$$\mathcal{R}_{\text{emp}}(h) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(h(\boldsymbol{x}^{(i)}), \boldsymbol{y}^{(i)})$$

Minimize $\mathcal{R}_{\text{emp}}(h)$ to find $\hat{h}$:

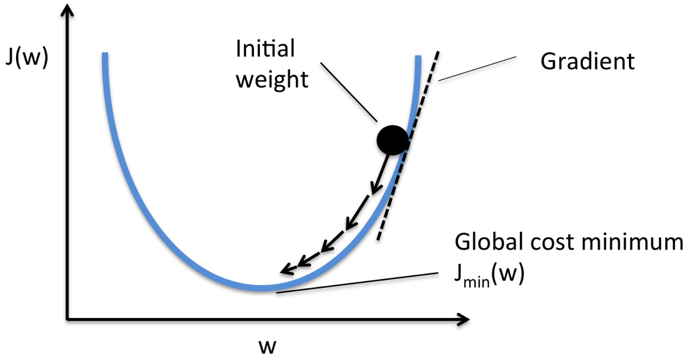
$$\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{R}_{\text{emp}}(h)$$

In practice:

- instead of $p(\boldsymbol{x}, \boldsymbol{y})$, we use training set $\mathcal{S}_{\text{training}}$
- instead of $\mathcal{F}$, we use $\mathcal{H} \subset \mathcal{F}$, e.g., all polynomials of degree 5

Optimizing objective function

Gradient descent



- initialize model parameters $\theta_0, \theta_1, ..., \theta_m$
- repeat until converge, for all θ_i

$$\theta_i^t \leftarrow \theta_i^{t-1} - \lambda \frac{\partial}{\partial \theta_i^{t-1}} \mathcal{J}(\Theta),$$

where the objective function $\mathcal{J}(\Theta)$ is evaluated over all training data $\{(\mathbf{X}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$.

Problem Set 1

Stochastic Gradient Descent (SGD): in each step, randomly sample a mini-batch from the training data and update the parameters using gradients calculated from the mini-batch only.

Evolution of optimizers

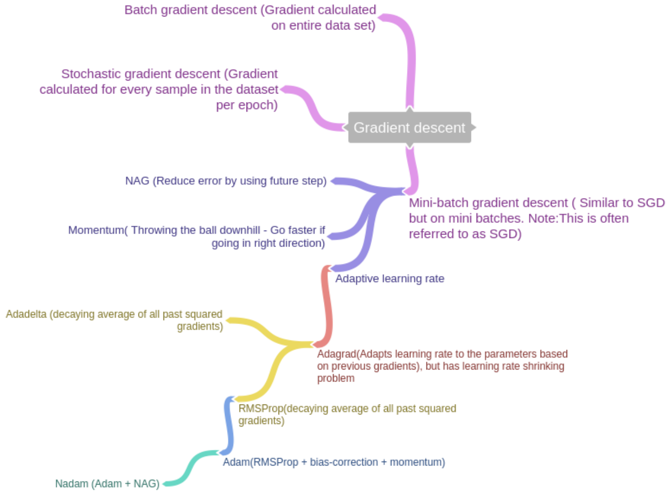

Batch gradient descent (Gradient calculated on entire data set)
Stochastic gradient descent (Gradient calculated for every sample in the dataset per epoch)
Gradient descent
NAG (Reduce error by using future step)
Mini-batch gradient descent (Similar to SGD but on mini batches. Note:This is often referred to as SGD)
Momentum(Throwing the ball downhill - Go faster if going in right direction)
Adaptive learning rate
Adadelta (decaying average of all past squared gradients)
Adagrad(Adapts learning rate to the parameters based on previous gradients), but has learning rate shrinking problem
RMSProp(decaying average of all past squared gradients)
Adam(RMSProp + bias-correction + momentum)
Nadam (Adam + NAG)

Figure: Evolution of gradient descent optimization algorithms (image by Desh Raj)

Update equations

Method	Update equation
SGD	$g_t = \nabla_{\theta_t} J(\theta_t)$ $\Delta\theta_t = -\eta \cdot g_t$ $\theta_t = \theta_t + \Delta\theta_t$
Momentum	$\Delta\theta_t = -\gamma\, v_{t-1} - \eta g_t$
NAG	$\Delta\theta_t = -\gamma\, v_{t-1} - \eta \nabla_\theta J(\theta - \gamma v_{t-1})$
Adagrad	$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$
Adadelta	$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$
RMSprop	$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$
Adam	$\Delta\theta_t = -\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$

Figure: Update equations for different gradient descent optimization algorithms [Ruder, 2016]

Training, validation, test sets

Training set ($\mathcal{S}_{\text{training}}$):

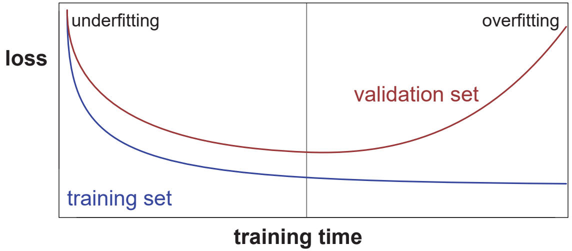
- set of examples used for learning
- usually 60 - 80 % of the data

Validation set ($\mathcal{S}_{\text{validation}}$):

- set of examples used to tune the model hyperparameters
- usually 10 - 20 % of the data

Test set ($\mathcal{S}_{\text{test}}$):

- set of examples used only to assess the performance of fully-trained model
- after assessing test set performance, model must not be tuned further
- usually 10 - 30 % of the data



Confusion matrix and derived metrics

		True condition		
	Total population	Condition positive	Condition negative	Accuracy = $\frac{\Sigma\ \text{True positive} + \Sigma\ \text{True negative}}{\Sigma\ \text{Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Precision = $\frac{\Sigma\ \text{True positive}}{\Sigma\ \text{Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	
		Recall, Sensitivity $= \frac{\Sigma\ \text{True positive}}{\Sigma\ \text{Condition positive}}$	Specificity $= \frac{\Sigma\ \text{True negative}}{\Sigma\ \text{Condition negative}}$	F_1 score = $\frac{1}{\frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}}$

Problem Set 1

Accuracy: proportion of true predictions - (TP + TN) / (TP + FP + TN + FN)

Quo vadis, 6.874?

- neural networks (NNs)
  - convolutional neural networks (CNNs)
  - recurrent neural networks (RNNs)
  - residual neural networks
  - (variational) autoencoders (VAEs)
  - generative adversarial networks (GANs)
- regularization
  - L_1 regularization
  - L_2 regularization
  - dropout
  - early stopping
- model selection
  - cross-validation (CV)
  - Akaike information criterion (AIC)
  - Bayesian information criterion (BIC)

- model interpretation methods
  - sufficient input subsets (SIS)
  - saliency maps
  - LIME
- dimensionality reduction methods
  - principal component analysis (PCA)
  - t-SNE
  - autoencoders
  - non-negative matrix factorization (NMF)
- hyperparameter optimization and AutoML

References

Hastie, T., Tibshirani, R., and Friedman, J. (2001).
The Elements of Statistical Learning.
Springer, New York, NY, USA.

Mitchell, T. M. (1997).
Machine Learning.
McGraw-Hill, Inc., New York, NY, USA.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012).
Foundations of Machine Learning.
MIT Press, Cambridge, MA, USA.

Murphy, K. P. (2012).
Machine learning : a probabilistic perspective.
MIT Press, Cambridge, MA, USA.

Ruder, S. (2016).
An overview of gradient descent optimization algorithms.
arXiv, 1609.04747.

Shalev-Shwartz, S. and Ben-David, S. (2014).
Understanding Machine Learning: From Theory to Algorithms.
Cambridge University Press, New York, NY, USA.