Machine Learning:

Introduction and Fundamentals

PRINCETON
UNIVERSITY

Conceptual Overview

Giving computers the ability to learn without explicitly programming them.

Arthur Samuel, 1959

Computer algorithms that:

- Learn an output-input relationships through data
- Characterize patterns and relationships amongst system variables

(Some) Applications of Machine Learning

- Natural Language Processing: chatbots.
- Computer Vision: object detection, segmentation, depth estimation.
- Search: retrieval, ordering.
- Recommendations: ads, entertainment, e-commerce.
- Forecasting: financial markets.
- Anomaly detection: freud, spam, cyber-threads.
- Robotics.
- Automation: self-driving cars, smart spaces.
- Science: particle physics, astrophysics, cosmology.

Motivation for Machine Learning

- Use case: given an image, determine whether there are any faces present and return the bounding box of each face, i.e. **face detection**.
- Difficult computer vision problem because of variability: expressions, illuminations, skin types, scale, orientation, background, occlusion, resolution.
- Designing feature-based methods based on hand crafted edge detectors (kernels or filters) to target facial features is an impossible task.
- Instead: train a **model** using available data to learn these filters.

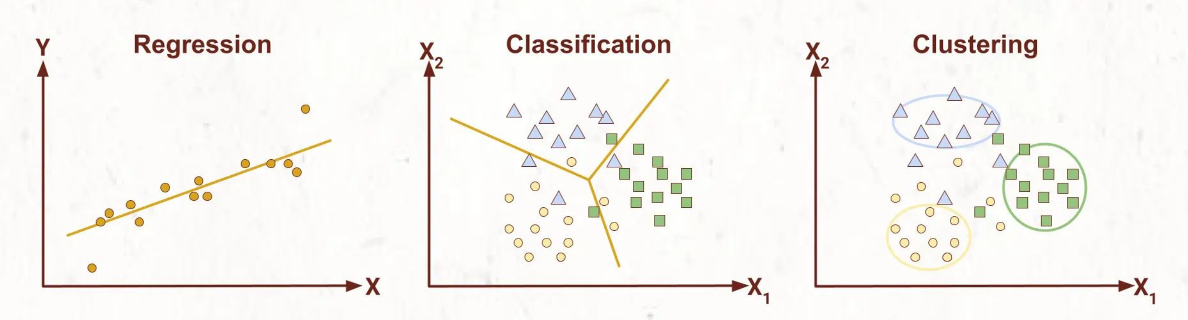
What is a model?

Concept of a Model

- A model is a parameterized function f_θ where $\theta \in \mathbb{R}^n$ are the parameters.
- A model is a mathematical characterization of system(s) of interest.
- A model **learns** from data, i.e. we ***fit*** a model on a dataset.
- Given N examples with features and targets, a training dataset $D = \{(x_i, y_i)\}^N_{i=1}$ parameters θ are learned via a learning algorithm that optimize an objective.
- A model can be seen as a computer program with θ and x_i as input arguments performing a **prediction** or **inference** on new data points.

Types of Machine Learning Problems

- **Classification**: predict a label, i.e. $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set of classes.
- **Regression**: predict a continuous variable, $y_i \in \mathbb{R}$.
- **Clustering**: group similar set of objects.



Not Mine

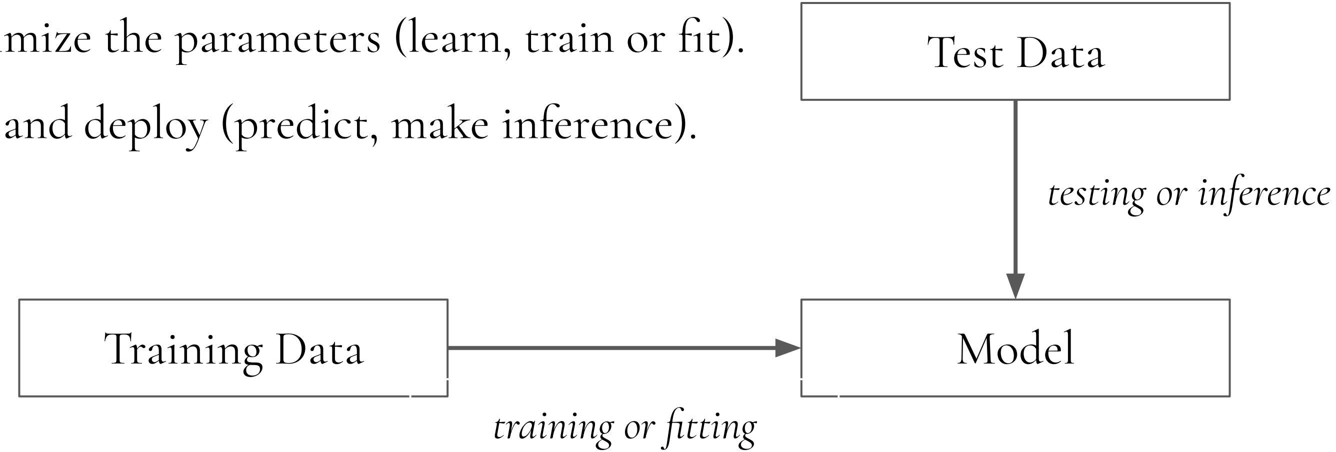
Others: dimensionality reduction, generation, association, collaborative filtering.

Types of learning

- **Supervised**: labels are available.
    - Examples: classification, regression.
- **Unsupervised**: labels are unavailable.
    - Examples: clustering, dimensionality reduction.
- **Semi-supervised**: some labels are available.
    - Examples: some generative models.
- **Self-supervised**: some learning signal required, e.g. outputs distance.
- **Reinforcement learning:** an agent learns through interaction with a system.

Typical pipeline

- Collect the data (model ***experience***).
- Choose a model (and model ***capacity***).
- Optimize the parameters (learn, train or fit).
- Test and deploy (predict, make inference).



Loss function

- Find best parameters which minimize the ***loss function*** L.
- Examples for supervised learning:
  - **Squared error** (for regression): $L(f_\theta(x), y) = (f_\theta(x) - y)^2$
  - **Cross-entropy** (for classification): $L(f_\theta(x), y) = -y \log(f_\theta(x)) - (1\text{-}y)\log(1\text{-}f_\theta(x))$
- Minimizing the loss function should result in maximizing the **metric**.

Evaluation Metrics: Regression

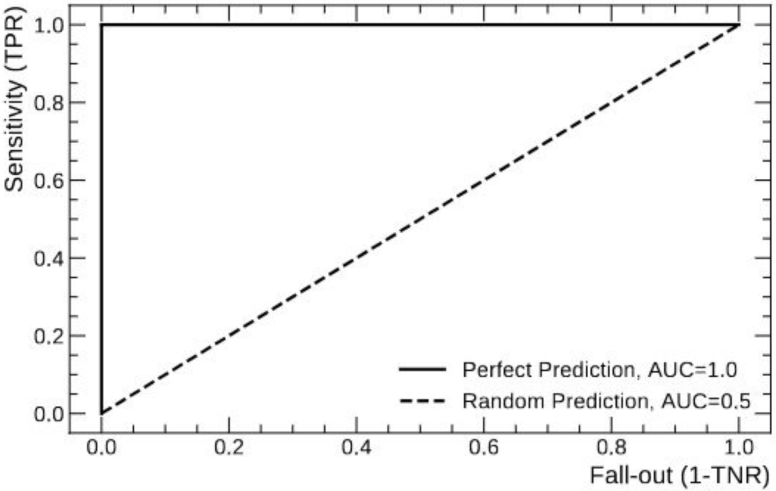
- **Metric**: a function and measure of a quantitative assessment a model.
- Regression metrics:
  - **Mean Squared Error**
  - Mean Absolute Error
  - Median Absolute Error
  - R2 Score
  - Mean Pinball Loss
  - Mean Absolute Percentage Error

Evaluation Metrics: Classification

- True positive (TP): correctly indicates the presence of a condition.
- False positive (FP, Type I error): wrongly indicates that a particular condition is present.
- True negative (TN): correctly indicates the absence of a condition.
- False negative (FN, Type II error): wrongly indicates that a particular condition is absent.
- Classification metrics:
  - Accuracy: (TP + TN) / (P + N)
  - Sensitivity: TP/P
  - False Positive Rate: FP/N
  - Receiver Operating Characteristic (ROC): Sensitivity vs. False Positive Rate

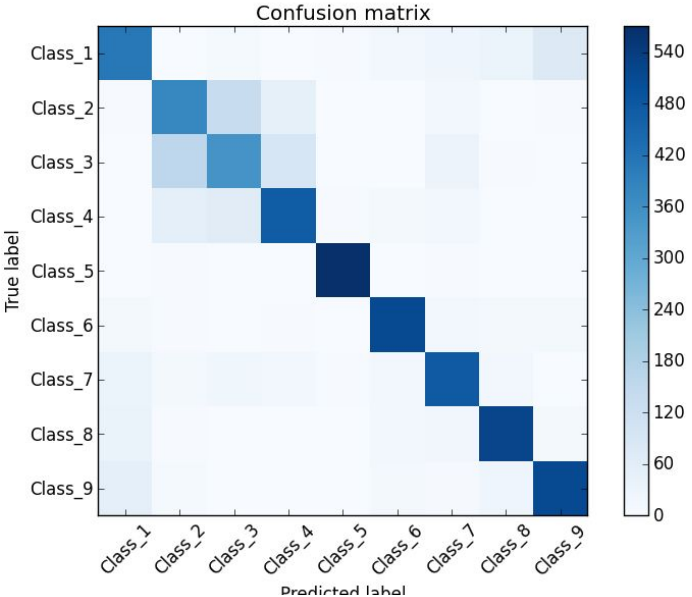
Evaluation Metrics: Classification

- Receiver Operating Characteristic (**ROC**): Sensitivity vs. False Positive Rate.
- ROC Area Under the Curve (**AUC**): integral over the ROC line, the higher the better.



Evaluation Metrics: Classification, Confusion Matrix

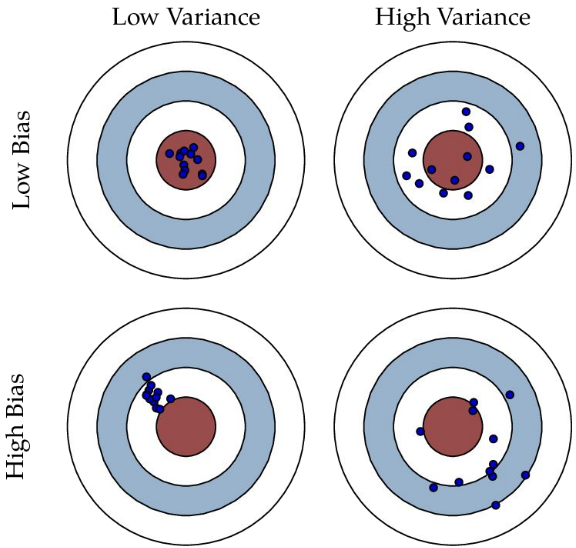
- Confusion matrix: a table that allows interpretation of the algorithm performance.



Not Mine

Bias-Variance Trade-off

- ***Bias*** is the model error from erroneous assumption, i.e. systematic error
- ***Variance*** is the model sensitivity to small changes in the data, i.e. noise sensitivity.



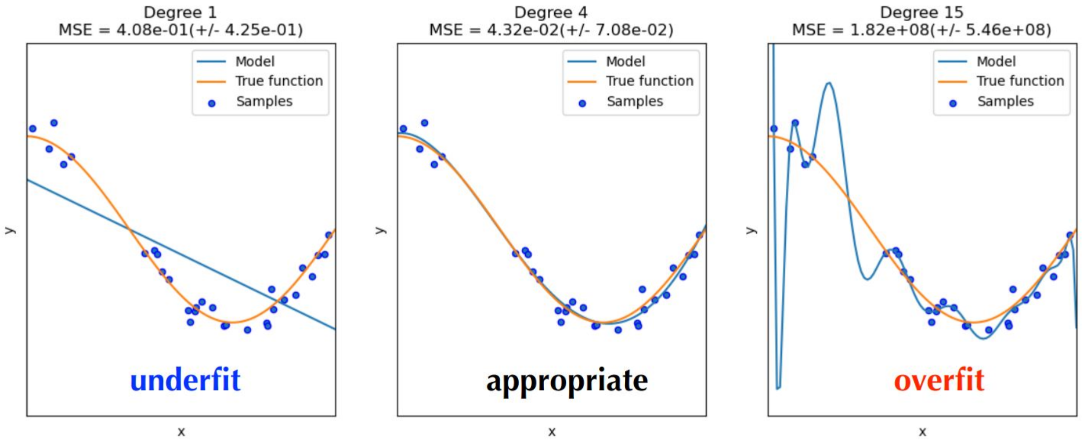
[Not Mine]

Model Generalization

- Model **generalization**: ability to react properly to new, previously unseen data.
- **Underfitting** (low variance, high bias) with poor train and test results.
  - Inaccurate inference.
  - Model is too simple or it can't capture underlying data structure.
  - Model does not have enough capacity.
- **Overfitting** (high variance, low bias) with good training error but bad test results.
  - Inaccurate inference.
  - Model captures noise instead of the input structure (low generalization).
  - Model has too much capacity.

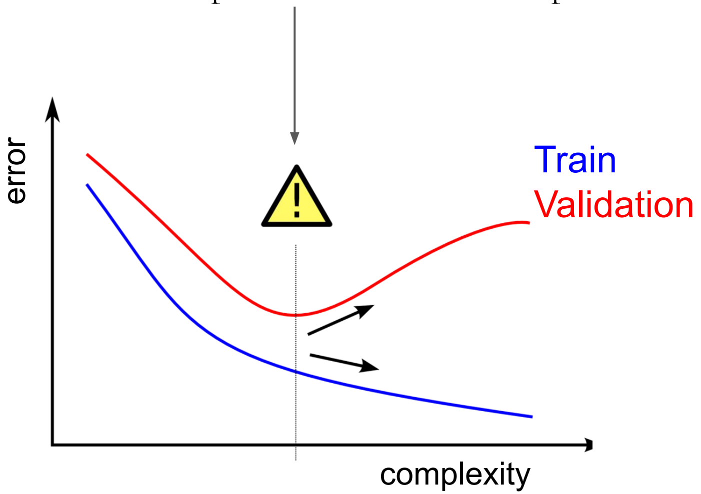
Model Generalization

- Model **generalization**: ability to react properly to new, previously unseen data.
- **Underfitting** (low variance, high bias) with poor train and test results.
- **Overfitting** (high variance, low bias) with good training error but bad test results.



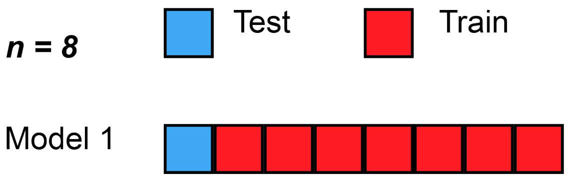
Monitoring Generalization

- Split the dataset.
    - Training set: use to fit the model.
    - Validation set: verify generalization used for model selection.
    - Test set: use for the final independent check after all parameters are fixed.



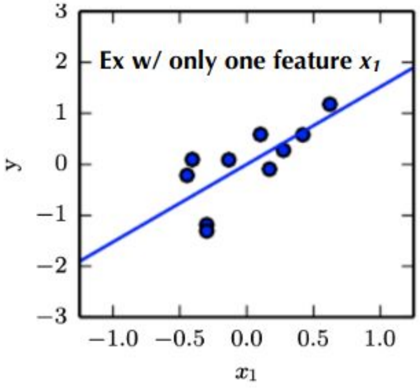
Cross-validation

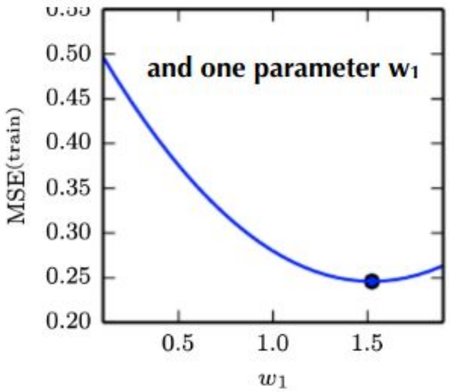
- Cross-validation: better estimation (with variance) of model performance.
    - Split the dataset in K chunks (called folds).
    - Train on K-1 chunks, validate on 1 chunk and average results.
- Too many folds: small bias, large variance (due to small split sizes), costl.
- Too few folds: large bias, small variance, cheap.



Linear Regression

- Given $D = \{(x_i, y_i)\}^N_{i=1}$, where $x \in \mathbb{R}^m$ and $y \in \mathbb{R}$, and m is the number of features.
- We look for a linear model $f(\mathbf{w}, x) = y = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w} = \{w_0, w_1, \ldots, w_m\}$.
- We will use Mean Squared Error as a loss function, $L(\mathbf{w}) = (f(\mathbf{w}, x) - y)^2/N$, to minimize $\mathbf{w}^* = \mathbf{argmin}_\mathbf{w} L(\mathbf{w})$.



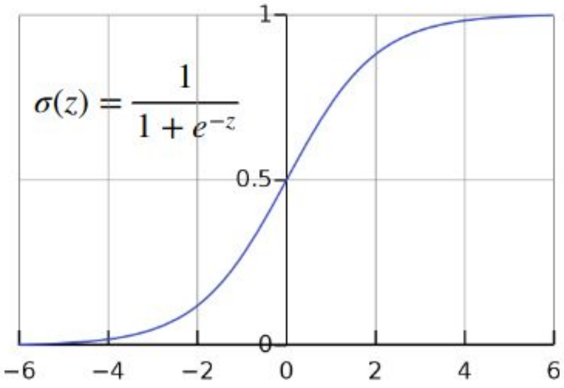


Linear Regression: Solution

- Rewrite linear model as: $\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$, or $\mathbf{y} = \mathbf{Xw}$
- Rewrite loss as: $||\mathbf{y} - \mathbf{Xw}||^2 = \frac{1}{n}(\mathbf{y} - \mathbf{Xw})^T(\mathbf{y} - \mathbf{Xw})$
- The solution can be solved analytically: $\mathbf{w}* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$

Logistic Regression

- Let's assume a binary classification problem.
- Given $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $x \in \mathbb{R}^m$ and $y \in \{0,1\}$, and m is the number of features.
- Convert the distance from the boundary into a probability: **sigmoid function**.
- The further from boundary, the more certain about a class.
- $p(y=1 \mid \mathbf{x}) = \dfrac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$



Multi-Class Classification

- For multi-class classification, i.e. $y \in \{c_1, c_2, \ldots, c_n\}$, y should be represented with one-hot vector, i.e. $y_i = c_k = (0,\ldots,1,\ldots 0)$, where only kth element is 1.
- Softmax: multi-class generalization of logistic sigmoid function, $p(c_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T \mathbf{x}}}{\sum_{j=1}^{N} e^{\mathbf{w}_j^T \mathbf{x}}}$

$$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \\ 0.7 \\ 1.1 \end{bmatrix} \longrightarrow \boxed{\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}} \longrightarrow \begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$$

Logistic Regression

- We write $p(y \mid \mathbf{x})$ as Bernoulli random variable:

$$p(y_i = y \mid \mathbf{x}_i) = (p_i)^{y_i}(1-p_i)^{1-y_i} = \begin{cases} p_i & \text{if } y_i = 1 \\ 1-p_i & \text{if } y_i = 0 \end{cases}$$

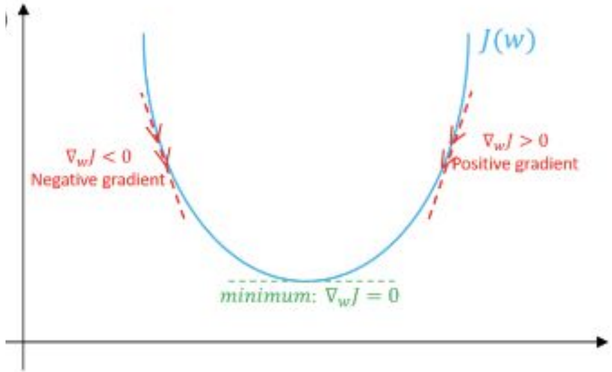
- Negative log-likelihood: $-\ln \prod_{\mathrm{i}} (\mathrm{p_i})^{\mathrm{y_i}}(1-\mathrm{p_i})^{1-\mathrm{y_i}} = -\sum_i y_i \ln(\mathrm{p_i}) + (1-\mathrm{y_i})\ln(1-\mathrm{p_i})$

- No analytical solution to **w***.

Gradient Descent

- We normally minimize loss by evaluating the derivatives (direction towards minimum).
- Gradient descent computes the gradient of the loss function w.r.t. to the current parameters θ.
- At each training step k update model parameters to move towards steepest decline:

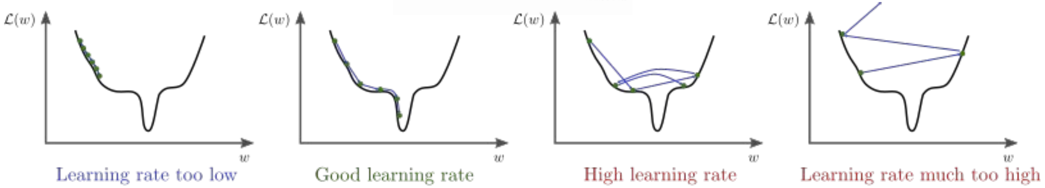
  $\theta_{k+1} \leftarrow \theta_k - \eta \cdot \nabla L(\theta_k)$, where $L(\theta_k)$ is the loss and η is the step size and k is the time
- Adjustment step is determined by learning rate η (hyperparameter).

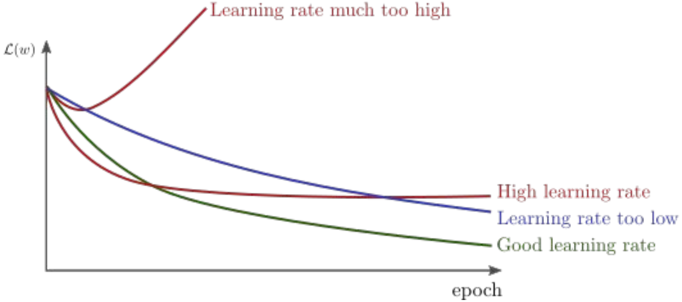


Choice of the Learning Rate

- Choice of the learning rate is critical for the successful training.

$$w \leftarrow w - \eta \frac{d\mathcal{L}}{dw}$$

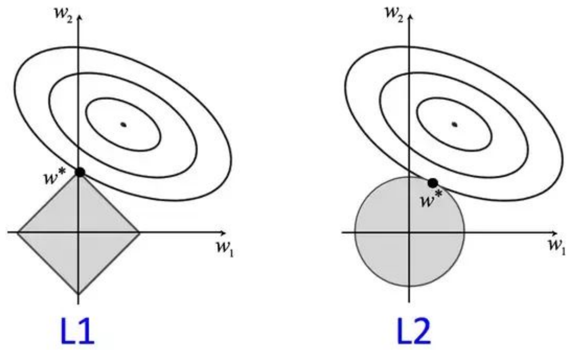




Credit

L1/ L2 Regularization

- Limit the capacity of the model by penalizing the value of weights.
- L1 regularization (Ridge): absolute value of weights for sparsity (feature selection): $\lambda \sum_{i=1}^{n} |\theta_i|$
- L2 regularization (LASSO): penalize square of weights: $\lambda \sum_{i=1}^{n} \theta_i^2$
- λ is a hyperparameter.



Hands On Session