From Feature Selection to Instance-wise Feature Acquisition

Tutorial @ SDM 2024

Daphney-Stavroula Zois¹      Charalampos Chelmis²

¹Electrical and Computer Engineering Department

²Computer Science Department

Saturday, April 20th, 2024

¹This research is based upon work supported by the National Science Foundation grants ECCS-1737443 & CNS-1942330 and a Google AI for Social Good award.
Organizers

Daphney–Stavroula Zois
Associate Professor in ECE
University at Albany

- Machine Learning
- Statistical Signal Processing

https://www.albany.edu/~dz973423/

Charalampos Chelmis
Associate Professor in CS
University at Albany

- Socially Important Data Science
- Big Data Analytics

http://www.cs.albany.edu/~cchelmis/
Tutorial Objectives

- Contrast feature selection to feature acquisition, and introduce related nomenclature
- Overview state-of-the-art and summarize research progress on this area
- Draw connections to recent trends in machine learning (e.g., model interpretability, fairness)
- Identify challenges and opportunities for future work
Tutorial Outline

- Introduction
  - Typical machine learning problem
  - Feature selection and variants
  - Applications and main challenges

- Online/Streaming feature selection
  - Problem definition
  - Main idea & methods
  - Variants (e.g., streaming data, feature interactions, group feature selection)

- Instance–wise feature acquisition
  - Problem definition
  - Static approaches
  - Dynamic methods

- Advanced Topics
  - Model interpretability
  - Incorporating fairness constraints
  - Dealing with structure (e.g., Bayesian network classification, hierarchical classification)
Relevant Tutorials

  - Focused on post hoc explainability, and discusses among others how features contribute towards a prediction
  - https://explainml-tutorial.github.io/neurips20

- Subset Selection in Machine Learning: Theory, Applications, and Hands On [at AAAI 2021]
  - Focused on the theoretical underpinnings of subset selection and discussed related applications, such as active and human assisted learning
  - https://explainml-tutorial.github.io/aaai21
Introduction
Typical Machine Learning Problem

- Training set $D$ consisting of $(x, y)$ pairs
  - Features $x$ are usually represented as fixed-length numeric feature vectors
  - Labels $y$ are typically modeled as integers
- **Goal:** Learn function $f : x \rightarrow y$ so the label(s) of unseen instances can be predicted
  - A loss function (e.g., zero-one) is selected
  - The empirical risk is then minimized
Feature Selection

- There are many “characteristics” that can help us recognize a cat from a dog, e.g.,
  - Overall size
  - Existence of whiskers
  - Shape of ears
  - etc

- Feature selection: select small subset of elements in $x$ that can be used to derive a good model
  - Features must be “as good as possible” wrt some criterion $C$
  - Sparse wrt to $x$
Benefits of Feature Selection

- As the number of features becomes large:
  - Learning models tend to overfit
  - High storage requirements and computational costs
  - Distances lose meaning
- This is where feature selection comes in
  - Remove irrelevant and redundant features
  - Enhance generalization performance
  - Increase computational efficiency (i.e., speed up the learning process)
  - Decrease memory storage
  - Improve model interpretability
Feature Selection Variants

- **Dimensionality reduction** (e.g., Principal Component Analysis)

- **Standard (offline) supervised feature selection**
Dimensionality Reduction

- Project original high dimensional features to new feature space with low dimensionality
- Newly constructed feature space is usually (non)linear combination of original features
Standard (Offline) Supervised Feature Selection [GE03]

- Feature subsets evaluated wrt information content, predictive accuracy of a given classifier or both
  - Filter methods: independent of learning algorithm
  - Wrapper methods: iteratively assess quality of selected features based on classifier’s learning performance
  - Embedded methods: embed feature selection into learning algorithm
- Smallest feature subset satisfying constraint is maintained

Training: all candidate features are available upfront
Testing: same final selected features used for classification
Applications

- Webspam page detection (16 million features) [WCP06]
- Educational data mining for predicting student performance (> 29 million features) [SNMR\(^{+10}\)]
- Hot topics detection in social media
- Bionformatics (full set of features is hard to acquired due to high cost of wet lab experiments)
- Planetary imaging, online visual tracking, etc
Main Challenges

- Exhaustive search over the entire feature space is computationally expensive in high-dimensional settings.
- Data instances and/or features may not be available in advance (e.g., online/streaming settings) or may be missing.
- In practice (e.g., medicine and criminal justice) features have an associated cost:
  - Acquisition (e.g., medical tests, evidence collection)
  - Privacy (e.g., revealing personally identifiable information)
  - Fairness (e.g., may amplify bias)
  - Energy consumption (e.g., communication, storage, or computational cost)
- Concept/distribution drift.
- Feature dependencies (e.g., multi-collinearity, group structure, multiview settings).
- Predictive power of different feature subsets may vary by subgroups of data instances (e.g., prognosis for different subpopulations).
Online/Streaming Feature Selection
Problem Definition [HZL$^+$18]

- Also known as incremental feature selection
- **Goal**: choose subset of features from larger set of potentially redundant features without access to full feature space in advance

**Training**: features arrive one at a time/batches  
**Testing**: same final selected features used for classification

- Representative methods can be categorized as threshold-based or rough set theory-based
Threshold-based Streaming Feature Selection

- Newly arriving feature is selected if specific constraint is satisfied
- Representative methods include:
  - Grafting [PLT03]
  - Alpha-investing [ZFSU05]
  - OSFS / Fast-OSFS [WYD+12]
  - SAOLA [YWDP16]
  - OSSFS-DD [ZZYW22]
Features are categorized into four disjoint groups:

- **irrelevant**: \( P(C = c_i | S = s, F_i = f_i) = P(C = c_i | S = s) \) for all \( S \subseteq F \setminus \{F_i\} \)
- **strongly relevant**: if above condition not met
- **redundant**: has Markov blanket \( M \) (i.e., \( P(F_i | M, Y) = P(F_i | M) \) for all \( Y \in F \setminus (M \cup \bigcup F_i) \)) within \( F \)
- **non–redundant**: \( P(C = c_i | S = s, F_i = f_i) \neq P(C = c_i | S = s) \) for some \( S \subset F \setminus \{F_i\} \)
Goal: At each step $t_i$, maintain minimum size feature subset $S_{t_i}^*$ that maximizes predictive classification performance

Key steps:
- Determine relevance of feature $F_i$ to class label $C$
  - If $P(C|F_i) = P(C)$, then discard $F_i$
  - Else, check if $F_i$ is redundant wrt already selected features
- If $F_i$ is relevant and not redundant, add it to the selected feature subset
- Pruning step: find the subset $\zeta$ that maximizes the probability $P(C|\zeta)$
Maintaining minimum size feature subset at each step requires examining all possible feature subsets. Does not scale with number of features. Therefore, problem is rewritten in terms of mutual information.

Mutual information between features is computed online using pairwise comparisons based on heuristics. Mutual information between features conditioned on all feature subsets need not be computed.
Scalable and Accurate OnLine Approach (SAOLA) [YWDP16]

**Algorithm 1:** The SAOLA Algorithm.

1: **Input:** $F_i$: predictive features, $C$: the class attribute; 
\[ \delta: \text{a relevance threshold (}0 \leq \delta < 1), \]
\[ S_{t_{i-1}}^*: \text{the selected feature set at time } t_{i-1}; \]
\[ Output: S_{t_i}^*: \text{the selected feature set at time } t_i; \]

2: repeat
3: get a new feature $F_i$ at time $t_i$;
4: /*Solve Eq.(2)*/
5: if $I(F_i; C) \leq \delta$ then
6: Discard $F_i$;
7: Go to Step 21;
8: end if
9: for each feature $Y \in S_{t_{i-1}}^*$ do
10: /*Solve Eq.(3)*/
11: if $I(Y; C) > I(F_i; C)$ & $I(F_i; Y) \geq I(F_i; C)$ then
12: Discard $F_i$; \hspace{1cm} And never consider it again!
13: Go to Step 21;
14: end if
15: /*Solve Eq.(4)*/
16: if $I(F_i; C) > I(Y; C)$ & $I(F_i; Y) \geq I(Y; C)$ then
17: \[ S_{t_{i-1}}^* = S_{t_{i-1}}^* - Y; \]
18: end if
19: end for
20: $S_{t_i}^* = S_{t_{i-1}}^* \cup F_i$;
21: until no features are available
22: Output $S_{t_i}^*$;

Determine the **relevance** of feature $F_i$ to class label $C$

Determine whether $F_i$ should be retained given the current feature set $S_{t_{i-1}}^*$

Check if some features within $S_{t_{i-1}}^*$ can be removed due to the inclusion of new feature $F_i$
Online Streaming Feature Selection (OSFS) [WYD⁺12]

- **Goal:** find optimal subset comprising non–redundant and strongly relevant features
  - Features are categorized into four disjoint groups
  - Unlike SAOLA, uses $G^2$ test to measure conditional independence
- Alternating two–step process
  - **Relevance analysis:** determine if streaming feature is relevant, and if so, add to candidate feature set and Markov blanket of class label $C$
  - **Redundancy analysis:** identify and remove redundant features in Markov blanket of class label $C$
    - Key insight: if a feature is marked redundant, it remains redundant even if some features within its Markov blanket are removed later on
- Stopping criteria (prediction accuracy, maximum number of iterations, all features examined)
Online Streaming Feature Selection (OSFS) [WYD+12]

- Redundancy analysis re-examines relevance of each feature in candidate set wrt class label every time a new feature is added (time-consuming)

- Fast OSFS:
  - If current streaming feature (as opposed to each and every feature) is relevant but redundant, remove it from candidate feature set
  - Else, add current feature in candidate feature set, and check redundancy of each feature in candidate set wrt subsets that include newly added feature
In online streaming feature selection, discarded features are never considered again.

- For weakly relevant features making a decision (selecting or discarding) immediately is risky.
- ∀ new arriving feature $f_t$
  - If strongly relevant, add it into the candidate feature subset $S_C$.
  - If irrelevant, discard it immediately.
  - If weakly relevant, add it into undetermined feature subset $S_U$ and defer decision.
Compute membership score, $\gamma_f(d) \in [0, 1]$, between feature $f$ and the decision class $d$ using Normalized Mutual Information

- if $\beta \leq \gamma_f(d) \leq 1$, $f$ is strongly relevant to $d$
- if $\alpha < \gamma_f(d) < \beta$, $f$ is weakly relevant to $d$
- if $0 \leq \gamma_f(d) \leq \alpha$, then $f$ is irrelevant to $d$

But how to choose proper thresholds of $\alpha$ and $\beta$?

- Assume normally distributed data, and features arriving at random
- Membership scores in the whole feature space are also normally distributed with mean value $\mu$ and standard deviation $\sigma$
- Set $\alpha = \mu - \sigma$ and $\beta$

Without knowledge of the entire feature space the thresholds cannot be set a–priori

Thankfully, the mean and standard deviation can be dynamically updated $\forall f_t$

- $\mu_t = \mu_{t-1} + \frac{\gamma_t - \mu_{t-1}}{t}$ and $\sigma_t = \sqrt{\frac{(t-2)\sigma_{t-1}^2 + (\gamma_t - \mu_{t-1})(\gamma_t - \mu_t)}{t-1}}$
Feature redundancy

- Two features $f_1$ and $f_2$ must contain some common information if
  $$I(f_1, f_2; d) < I(f_1; d) + I(f_2; d)$$
- If additionally $I(f_1, f_2; d) < 2\beta$ remove the feature with the smaller value of $I(f_1; d)$ or $I(f_2; d)$
- Note: For each new feature, must check for redundancy between that feature and every feature currently in $S_C$

Feature uncertainty

- $f_i$ is added to $S_U$ if $\alpha < I(f_i; d) < \beta$
- if $\exists f_j \in S_U$ s.t. $I(f_i, f_j; d) \geq 2\beta$, add both $f_i$ and $f_j$ into $S_C$
- Features that don’t satisfy this are discarded when $S_U$ reaches a threshold to avoid $S_U$ becoming too large
Threshold–based streaming feature selection typically require prior information about feature space.

Representative methods include:
- OFS–Density [ZHLW19a]
- OFS–A3M [ZHLW19b]
OFS–Density [ZHLW19a]

- Two types of neighborhoods
  - $\delta$ neighborhood (set $\{y | (x, y) \leq \delta\}$, where $\Delta$ and $\delta$ are a distance metric and threshold respectively)
  - k-nearest neighborhood (determined by a fixed number of neighbors)
- Goal is to minimize the size of the boundary region when feature subset $B$ is used
New neighborhood relationship is defined
- All neighbors of $x$ are sorted by distance (nearest to farthest) on feature subset $B$
- Pairwise distance between consecutive points in this set is computed
- For some neighbor $x_k$ (Inflection Point), pairwise distance decreases for the first time
- The samples between $x$ and $x_k$ are used as the nearest neighbors of $x$
At time $t$ feature $f_t$ arrives, while $S_{t-1}$ is the set of selected candidate features.

The goal is to select features from $S_{t-1} \cup \{f_t\}$ with high correlation.

- Calculate dependency, $\gamma_{f_t}(D)$ of $f_t$ with target class label $D$.
- Calculate the mean $R(S_{t-1}, D)$ of dependency values $\forall f_j \in S_{t-1}$.
- Discard $f_t$ if the dependency of $f_t$ is less than $R(S_{t-1}, D)$.

High dependency

- If $\gamma_{S_{t-1} \cup \{f_t\}}(D) \geq \gamma_s$ add $f_t$ to $S_{t-1}$.

Low redundancy

- Discard all features $f_j$ in $S_t$ for which $\gamma_{S_t}(D) - \gamma_{S_t-f_j}(D) = 0$.
- In practise, the equality constraint is relaxed to an interval restriction.
Sparse Online Learning

- **Goal**: learn sparse linear classifier from sequence of high-dimensional training instances
- **Number of features** used by model must be given

| **Training**: data instances arrive sequentially to iteratively update classifier function |
| **Testing**: same final selected features used for classification |
Online Feature Selection (OFS) [WZHJ13]

- **Setting:** Binary classification, where each data instance $x_t$ is to be classified by a linear function $\text{sgn}(\mathbf{w}^\top \mathbf{x}_t)$.
  - Full vector is available for each data instance
- **Goal:** design effective strategy for OFS under constraint that classifier $w_t$ has at most $B$ nonzero elements, $||\mathbf{w}_t|| \leq B$
  - At most $B$ features of $\mathbf{x}_t$ are used for classification
  - Simply truncating features with small weights can lead to many misclassifications
Online Feature Selection (OFS) [WZHKJ13]

Algorithm 3 OFS via Sparse Projection. (OFS)

1: Input
   • $\lambda$: regularization parameter
   • $\eta$: step size
   • $B$: the number of selected features
2: Initialization
   • $w_1 = 0$
3: for $t = 1, 2, \ldots, T$ do
4:    Receive $x_t$
5:    Make prediction $\text{sgn}(w_t^T x_t)$
6:    Receive $y_t$
7:    if $y_t w_t^T x_t \leq 1$ then
8:       $\tilde{w}_{t+1} = (1 - \lambda \eta)w_t + \eta y_t x_t$
9:       $\hat{w}_{t+1} = \min\{1, \frac{1}{\|\tilde{w}_{t+1}\|_2}\} \tilde{w}_{t+1}$
10:      $w_{t+1} = \text{Truncate}(\hat{w}_{t+1}, B)$
11:    else
12:      $w_{t+1} = (1 - \lambda \eta)w_t$
13:    end if
14: end for

- A linear classifier $w_t$ is trained online with at most $B$ non–zero elements
- When a training instance $(x_t, y_t)$ is misclassified, the classifier is first updated by online gradient descent and then projected to a $L1$ ball to ensure that the norm of the classifier is bounded
- If $\hat{w}_{t+1}$ has more than $B$ non–zero elements, only the $B$ elements with the largest absolute weight are retained
Online Feature Selection (OFS) [WZHJ13]

- **Challenge:** Although only $B$ weights are non-zero, every attribute in $x_t$ must be measured and computed.

- **Solution:** $B$ out of all $d$ attributes are randomly selected for a number of training data instances, while for the remaining data instances, the $B$ attributes for which the classifier $w_t$ has non-zero values are selected.

$$
\begin{align*}
3: & \text{ for } t = 1, 2, \ldots, T \text{ do} \\
4: & \quad \text{Sample } Z_t \text{ from a Bernoulli distribution with probability } \epsilon. \\
5: & \quad \text{if } Z_t = 1 \text{ then} \\
6: & \quad \quad \text{Randomly choose } B \text{ attributes } C_t \text{ from } [d] \\
7: & \quad \text{else} \\
8: & \quad \quad \text{Choose the attributes that have non-zero values in } w_t, \text{ i.e., } C_t = \{i : [w_t]_i \neq 0\} \\
9: & \quad \text{end if} \\
10: & \quad \text{Receive } \tilde{x}_t \text{ by only requiring the attributes in } C_t \\
11: & \quad \text{Make prediction } \text{sgn}(w_t^T \tilde{x}_t) \\
12: & \quad \text{Receive } y_t \\
13: & \quad \text{if } y_t w_t^T \tilde{x}_t \leq 1 \text{ then} \\
14: & \quad \quad \text{Compute } \hat{x}_t \text{ as} \\
15: & \quad \quad \quad [\hat{x}_t]_i = \frac{[\tilde{x}_t]_i}{\frac{B}{d} \epsilon + I([w_t]_i \neq 0)(1 - \epsilon)}, i = 1, \ldots, d \\
16: & \quad \quad \tilde{w}_{t+1} = w_t + y_t \eta \hat{x}_t \\
17: & \quad \text{else} \\
18: & \quad \quad \text{Compute } \text{sgn}(w_t^T \tilde{x}_t) \\
19: & \quad \quad \tilde{w}_{t+1} = \text{Truncate}(\tilde{w}_{t+1}, B) \\
20: & \quad \text{end if} \\
21: & \text{end for}
\end{align*}
$$
Second–order Online Feature Selection (SOFS) [WHMY17]

- Main drawback for OFS is its linear time complexity wrt feature dimensionality
- **Goal**: improve performance and time complexity using second–order online learning techniques
- **Main idea**: use confidence–weighted (CW) method [DCP08]
  - Assume that weight vector of linear classifier follows Gaussian distribution
  - Based on observed training example \((x^t, y^t)\), CW updates mean vector and covariance matrix of Gaussian distribution
  - Ensure that probability of correct prediction on observed training example is bigger than specified threshold \(\tau\) while staying close to previous distribution

\[
(\hat{\mu}^{t+1}, \Sigma^{t+1}) = \arg\min_{\mu, \Sigma} \text{KL}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu^t, \Sigma^t)) \\
\text{s.t. } \Pr[y^t \text{sgn}(w \cdot x^t) > 0] \geq \tau
\]
Second–order Online Feature Selection (SOFS) [WHMY17]

- Kullback–Leibler (KL) divergence can be easily computed in terms of mean vectors and covariance matrices
- Solve optimization problem with adaptive regularization of the prediction function (AROW) for each new observed training example [CKD13]
- Update most confident $B$ weight variables, whose covariance values $\Sigma_{jj}$ are among the $B$ smallest
- **MeanHeap–based implementation** to store $B$ smallest diagonal values of covariance matrix $\Sigma^t$
- SOFS has linear time complexity wrt average number of nonzero features per instance
Group–SAOLA [YWDP16]

- **Goal:** select (in an online manner) **feature groups** which are **sparse** at the levels of both features and groups simultaneously
  - Extension of SAOLA for streaming features arriving in groups
- Feature groups appear in a sequential order, one at a time
  - Must optimize selections within each group, as well as between groups
Group–SAOLA [YWDP16]

- Extends notion of relevance to groups:
  - **irrelevant**: \( I(C; G_i) = 0 \)
    - simplified as \( I(C; F_i) \leq \delta, \forall F_i \in G_i \)
  - **redundant**: \( I(C; G_i|G \setminus G_i) = 0 \)
    - simplified as \( I(F_j; C) > I(F_i; C) \) and \( I(F_j; F_i) \geq I(F_i; C) \) \( \forall F_i \in G_i, \exists F_j \in G_j \), where \( G_j \in \Psi_{t_i} \), the set of groups selected at time \( t_i-1 \)

- Defines intra–group feature redundancy
  - **redundant**: \( I(C; F_i|S) = 0 \) for some \( S \subset G_i \setminus \{F_i\} \)
    - simplified as \( I(Y; C) > I(F_i; C) \) and \( I(F_i; Y) \geq I(F_i; C) \) for some \( Y \in G_i \)
/*Evaluate irrelevant groups*/
if \( \forall F_i \in G_i, I(F_i; C) \leq \delta \) then
    Discard \( G_i \);
    Go to Step 39;
end if

/*Evaluate feature redundancy in \( G_i \)*/
for \( j = 1 \) to \(|G_i|\) do
    if \( \exists Y \in \{G_i - \{F_j\}\}, I(Y; C) > I(F_j; C) \)
        & \( I(Y; F_j) \geq I(F_j; C) \) then
        Remove \( F_j \) from \( G_i \);
        Continue;
    end if
/*Otherwise*/
    if \( I(F_j; C) > I(Y; C) \) & \( I(F_j; Y) \geq I(Y; C) \) then
        Remove \( Y \) from \( G_i \);
    end if
end for

Determine the relevance of group \( G_i \) to class label \( C \)

/*Evaluate group redundancy in \( \{\Psi_{k-1} \cup G_i\} \)*/
for \( j = 1 \) to \(|\Psi_{k-1}|\) do
    if \( \exists F_k \in G_j \subset \Psi_{k-1}, \ \exists F_i \in G_i, \ I(F_i; C) > I(F_k; C) \)
        & \( I(F_i; F_k) \geq I(F_k; C) \) then
        Remove \( F_k \) from \( G_j \);
    end if
/*Otherwise*/
    if \( I(F_k; C) > I(F_i; C) \) & \( I(F_k; F_i) \geq I(F_i; C) \) then
        Remove \( F_i \) from \( G_i \);
    end if
    if \( G_i \) is empty then
        \( \Psi_{k-1} = \Psi_{k-1} - G_j \);
    end if
    if \( G_i \) is empty then
        Break;
    end if
end for

Identify redundant features within group \( G_i \)
Identify redundant groups and features from the currently selected groups
Instance–wise Feature Selection
Problem Definition

- **Informative features may vary by data instance** (e.g., heart failure prognosis across subpopulations [KLA+15])

- Ease of **interpretation** of popular but complex machine learning models

- **Goal**: identify small number of relevant features that explain machine learning model output for each data instance individually during testing

<table>
<thead>
<tr>
<th>Training: all candidate features are available upfront</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing: different (fixed or varying) number of features are selected for each data instance and used for model interpretation</td>
</tr>
</tbody>
</table>
Instance–wise Feature Selection

- Representative methods include:
  - SHAP [LL17]
  - L2X [CSWJ18]
  - INVASE [YJVdS18]
  - Mixture of Deep Neural Networks [XW19]
  - Instance–wise Feature Grouping [MWZ+20]
  - GroupFS [XLTW22]
  - DIWIFT [LCZ+23]

- Challenges:
  - Access to all features of test instance is needed before selecting relevant subset
  - Scalability issues for large feature spaces
A Unified Approach to Interpreting Model Predictions [LL17]

- Numerous model interpretability methods, but unclear how they are related or how to choose one over another

- **Goal**: unified framework for interpreting predictions
  - new class of additive feature importance measures unifying six existing methods
  - theoretical results showing the existence of a unique solution for this class with a set of desirable properties

*Figure source: LIME [RSG16]*
Let $f$ be the prediction model to be explained, and $g$ the explanation model.

Explanation models use simplified vectors $x'$ that map to the original instances through a mapping function $x = h_x(x')$.

- Local methods (e.g., LIME [RSG16]) explain $f(x)$, $\forall$ data instance $x$.
  - Try to ensure $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$.

Additive feature attribution methods use a linear function of binary variables, i.e.,

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

where $z' \in \{0, 1\}$, $M$ is the number of simplified input features, and $\phi \in \mathbb{R}$, as explanation model.

- Each feature $i$ is attributed effect $\phi_i$.
- The effects of all feature attributions are summed up to approximate $f(x)$.
Example Additive feature attribution method: LIME [LL17]

- LIME samples instances, gets predictions using $f$, and weighs them by the proximity to the instance being explained.
- Interprets individual model predictions by locally approximating $f$.

- Mapping $h_x$ depends on input type:
  - For bag of words, converts a vector of 1’s or 0’s into word counts if $x' = 1$, or 0 if $x' = 0$.
  - For images, a set of super pixels is used; if $x' = 1$ the super pixel’s original value is used, and the average of neighboring pixels is used otherwise.
Classic Shapley Value Estimation [LL17]

- Shapley regression
  - Feature importance for linear models in the presence of multicollinearity
  - Model is trained on all feature subsets $S \subseteq F$
  - Importance value represents the effect on the model prediction of including that feature
  - Computationally expensive!

- Shapley sampling
  - Sampling approximations
  - Approximating the effect of removing a variable from the model by integrating over samples from the training dataset
  - Eliminates the need to retrain the model and allows fewer than $2^{|F|}$ differences to be computed

- Quantitative input influence
  - Nearly identical to Shapley sampling values
SHAP (SHapley Additive exPlanation) Values [LL17]

- Shapley values of a conditional expectation function of model $f$
  - Obtained by solving for the only one possible explanation model $g$
- Mapping, $h_x(z') = z_S$, where $z_S$ has missing values for features not in the set $S$
  - Since most models cannot handle arbitrary patterns of missing input values, $f(z_S)$ is approximated with $E[f(z)|z_S]$

![Diagram showing SHAP values calculation](image)

- Sample explanation of how to get from the base value $E[f(z)]$ (if we did not know any features to the current output), using feature $x_1$, features $x_1$ and $x_2$ etc
- When the model is non-linear or features are not independent, the order in which features are added to the expectation matters
  - SHAP values arise from averaging the $\phi$ values across all possible orderings!
SHAP (SHapley Additive exPlanation) Values [LL17]

Why only one possible explanation model $g$?
- Two properties in addition to local accuracy
  - Missingness: constrains features where $x'_i = 0$ to have no attributed impact
  - Consistency: if a model changes so that some simplified input’s contribution increases (or stays the same regardless of the other inputs), that input’s attribution does not decrease

Values $\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!} [f_x(z') - f_x(z' \setminus i)]$ derived using combined cooperative game theory
- $|z'|$ is the number of non–zero entries in $z'$, and $z' \subseteq x'$ represents all $z'$ vectors where the non–zero entries are a subset of the non–zero entries in $x'$

Exact computation of SHAP values is challenging
- Model–agnostic approximation methods (Shapley sampling and Kernel SHAP)
- Model–type–specific approximation methods (Max SHAP, Deep SHAP)
- Feature independence and model linearity to simplify the computation of expected values
Learning to Explain (L2X) [CSWJ18]

- **Goal**: maximize mutual information between response variable of model and selected features, as function of choice of selection rule

\[
\max_{\mathcal{E}} I(X_S; Y) \quad \text{subject to} \quad S \sim \mathcal{E}(X)
\]

- Hyperparameter \(k\) : represents number of explaining features
- Applicable to classification/regression

- **Solution**: variational approximation
  - Derive lower bound on mutual information
  - Approximate model distribution conditioned on feature subset by rich family of functions
Learning to Explain (L2X) [CSWJ18]

- Relaxed problem

\[
\max_{\mathcal{E}, \mathcal{Q}} \mathbb{E} \left[ \log \mathcal{Q}_S(Y|X_S) \right] \quad \text{subject to} \quad S \sim \mathcal{E}(X)
\]

- Main idea:
  - Continuous approximation of feature subset sampling leads to

\[
\max_{\theta, \alpha} \mathbb{E}_{X,Y,\zeta} \left[ \log g_\alpha(V(\theta, \zeta) \odot X, Y) \right],
\]

where \( g_\alpha \) is neural network that approximates model conditional distribution and \( \theta \) parameterizes explainer

- Learned explainer maps each data instance \( X \) to weight vector \( w_\theta(X) \)
- Features \( X \) for specific data instance ranked based on \( w_\theta(X) \)
- Keep \( k \) features with largest weights for explanation
Goal: minimize KL divergence between conditional distributions $Y|X$ and $Y|X_S$ inducing sparsity using an $\ell_0$ penalty term

$$\min_{S(\cdot)} \mathbb{E}_{x \sim p_X} \left[ \text{KL}(Y|X = x \mid Y|X^S(x) = x^S(x)) + \lambda ||S(x)|| \right]$$

Solution: actor–critic architecture with three neural networks
- Use baseline network for variance reduction
- Use predictor network to provide reward to selector network
INstance-wise VArriable SElection (INVASE) [YJVdS18]

- Different number of relevant variables are selected for each data instance
- Can be used also for feature selection and prediction tasks
L2X and INVASE do not constrain search space for each data instance

Mixture of Deep Neural Networks [XW19] limits number of possible relevant feature subsets to $K$

- Each data instance $x$ has unique relevant feature subset
- Identify which model (model selector neural network) out of $K$ (feature subset selector neural networks) data instance comes from
- Select most relevant feature subject based on model sensitivity’s magnitude
Each data instance may be associated with different set of relevant features
Hard to understand feature importance pattern for entire data distribution

INVASE + K–means:
- Train instance–wise feature selector for each data instance
- Apply K–means clustering to all feature selectors
- Assigned cluster center is group–wise feature selector

Mixture of Experts selector:
**DIWIFT [LCZ+23]**

- **Feature-level influence function**: influence of perturbation \((x_i, y_i) \rightarrow (x_i + \delta_i, y_i)\) on loss
- Base pre-trained model w/o feature selection
- **Self-attention network** outputs instance-wise feature selection probabilities
- Compute influence function