Interactive session - LIME

In LIME, what is the order of the high-level steps?

- **Generate neighborhood → Get predictions → Weight by proximity → Train interpretable model → Explain**
- Train model → Generate neighborhood → Weight by proximity → Get predictions → Explain
- Get predictions → Generate neighborhood → Train interpretable model → Weight by proximity → Explain
- Generate neighborhood → Train interpretable model on original labels → Weight by proximity → Explain

What is an "interpretable representation" in LIME for images?

- **Superpixel/patch segments encoded as a binary vector**
- The gradient map of the image
- The raw pixel matrix (WxHxC)
- A learned embedding from the neural network

In the LIME objective — explanation(x) = argmin L(f, g, πx) + Ω(g) — what does Ω(g) represent and why is it minimized?

- The proximity between x and perturbed samples — minimized to focus on nearby points
- The prediction error of the original model f — minimized to improve accuracy
- **The complexity of the surrogate model g — minimized to keep explanations interpretable**
- The number of perturbed samples generated — minimized to reduce computation time

Why might LIME produce unrealistic neighbor samples, and what is the main reason for this?

- Because the linear surrogate model is too simple to capture complex boundaries
- **Because perturbations are generated independently per feature, ignoring correlations between features**
- Because proximity is measured using cosine similarity instead of Euclidean distance
- Because the number of neighbors sampled is too small

A data scientist runs LIME twice on the same instance and gets different explanations. Which of the following actions would most directly reduce this problem?

- Switch from a linear surrogate to a decision tree
- **Increase the number of perturbed samples generated for the neighborhood**
- Use a smaller value of K (number of interpretable features)
- Replace cosine similarity with Euclidean distance as the proximity measure

Which of the following best describes the trade-off captured by the full LIME objective L(f, g, πx) + Ω(g)?

- The trade-off between model accuracy on training data and generalization to unseen data

- **The trade-off between faithfully approximating the black-box locally and keeping the surrogate model simple enough to interpret**
- The trade-off between the size of the neighborhood and the speed of computation
- The trade-off between using interpretable features for explanation and raw features for training

Interactive session – Explaining by removing

A PredDiff score for attribute A on instance x is −0.15. What is the correct interpretation?

- Attribute A has no influence on the prediction
- **Attribute A pushes the prediction against class c by 0.15**
- Attribute A pushes the prediction toward class c by 0.15
- The model is 15% less accurate when A is removed
- The model is 15% more accurate when A is removed

Which of the following is a limitation of PredDiff that Shapley values are specifically designed to address?

- PredDiff cannot produce feature attributions
- **PredDiff does not consider interactions between features when computing individual attributions**
- PredDiff requires a differentiable model
- PredDiff only works for binary classification

In the Shapley value formula, what does the term $v(S) - v(S\{i\})$ represent?

- The total prediction for coalition S
- **The marginal contribution of player i to coalition S**
- The average prediction across all coalitions
- The penalty for adding player i to an already complete coalition

For a model with 20 features, how many coalitions must be evaluated to compute the exact Shapley value for a single feature?

- 20
- 400
- **$2^{20} = 1{,}048{,}576$**
- $20! \approx 2.4 \times 10^{18}$

Feature B never changes the model output regardless of which coalition it joins. According to the Null Player axiom, what Shapley value does it receive? And which real-world scenario does this best reflect?

- $\phi_B = 1.0$; a feature that perfectly predicts the target
- **$\phi_B = 0$; a completely irrelevant feature that the model ignores**
- $\phi_B = E[f(X)]$; a feature equal to the average prediction
- $\phi_B = -1.0$; a feature that actively harms prediction

Interactive session – Gradient

What is the key idea behind Integrated Gradients?

- Compute gradients at a single point and normalize them using feature statistics
- **Accumulate gradients along a path from a baseline input to the actual input**
- Replace gradients with feature importance learned from a surrogate model
- Average gradients across multiple models trained with different initializations

What is the main limitation of Grad-CAM compared to pixel-level attribution methods?

- It cannot be applied to convolutional neural networks trained on image data
- **It highlights broad regions rather than providing fine-grained feature importance**
- It requires access to the full training dataset to compute explanations
- It cannot distinguish between positive and negative feature contributions

What distinguishes GradCAM from Vanilla Gradient in terms of where the gradient is propagated?

- GradCAM propagates the gradient to the input image; Vanilla Gradient stops at the last conv layer
- **GradCAM propagates to the last convolutional layer; Vanilla Gradient goes all the way to the input**
- Both propagate to exactly the same layer
- GradCAM only uses forward pass information; no backpropagation is needed

Sensitivity is a key axiom for attribution methods. Which statement best defines it?

- Models with identical input-output behavior should produce identical attribution scores
- **If two inputs differ in one feature and yield different outputs, that feature must have non-zero attribution**
- The total attribution across features should equal the difference between prediction and baseline