Categorical data: Feature crosses

Feature crosses are created by crossing (taking the Cartesian product of) two or more categorical or bucketed features of the dataset. Like polynomial transforms, feature crosses allow linear models to handle nonlinearities. Feature crosses also encode interactions between features.

For example, consider a leaf dataset with the categorical features:

- `edges`, containing values `smooth`, `toothed`, and `lobed`
- `arrangement`, containing values `opposite` and `alternate`

Assume the order above is the order of the feature columns in a one-hot representation, so that a leaf with `smooth` edges and `opposite` arrangement is represented as `{(1, 0, 0), (1, 0)}`.

The feature cross, or Cartesian product, of these two features would be:

```
{Smooth_Opposite, Smooth_Alternate, Toothed_Opposite, Toothed_Alternate, Lobed_Opposite, Lobed_Alternate}
```

where the value of each term is the product of the base feature values, such that:

- `Smooth_Opposite = edges[0] * arrangement[0]`
- `Smooth_Alternate = edges[0] * arrangement[1]`
- `Toothed_Opposite = edges[1] * arrangement[0]`
- `Toothed_Alternate = edges[1] * arrangement[1]`
- `Lobed_Opposite = edges[2] * arrangement[0]`
- `Lobed_Alternate = edges[2] * arrangement[1]`

For example, if a leaf has a `lobed` edge and an `alternate` arrangement, the feature-cross vector will have a value of 1 for `Lobed_Alternate`, and a value of 0 for all other terms:

```
{0, 0, 0, 0, 0, 1}
```

This dataset could be used to classify leaves by tree species, since these characteristics do not vary within a species.

Click here to compare polynomial transforms with feature crosses

> Feature crosses are somewhat analogous to Polynomial transforms. Both combine multiple features into a new synthetic feature that the model can train on to learn nonlinearities. Polynomial transforms typically combine numerical data, while feature crosses combine categorical data.

When to use feature crosses

Domain knowledge can suggest a useful combination of features to cross. Without that domain knowledge, it can be difficult to determine effective feature crosses or polynomial transforms by hand. It's often possible, if computationally expensive, to use neural networks to *automatically* find and apply useful feature combinations during training.

Be careful—crossing two sparse features produces an even sparser new feature than the two original features. For example, if feature A is a 100-element sparse feature and feature B is a 200-element sparse feature, a feature cross of A and B yields a 20,000-element sparse feature.

Key terms:

- Feature cross
- Neural network