DYALOG
Eastbourne 2026

APLearn: Machine Learning in APL

Borna Ahmadzadeh

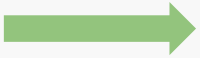
borna.ahz@gmail.com

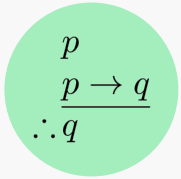
github.com/bobmcdear

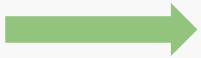
About Me

- Third-year computer student at York University in Toronto, Canada
- Programming since second grade
- Passionate about artificial intelligence
- Experience in computer vision, NLP, and traditional data science
- APL outsider; perspective that of machine learning developer!

Path to APL







Haskell

Programming beyond imperative languages

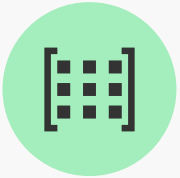
Foundations

Type systems, category theory, ...

APL

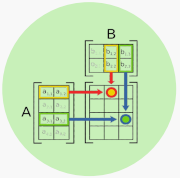
Notation as a tool of thought

Serendipity of Deep Learning & APL



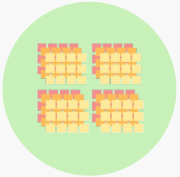
Arrays

Input and parameters are multi-dimensional arrays.



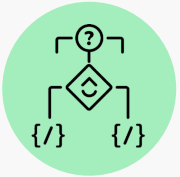
Parallel Operations

Key operations include matrix multiplication, reductions, etc.



Regular shapes

Data is neither jagged nor deeply nested.



Conditional-free

Conditional statements and dynamic control flow are absent.

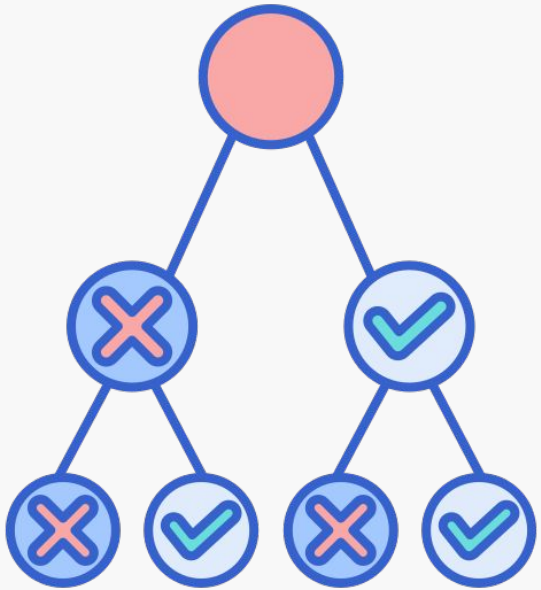
trap

GPT-like transformers in APL

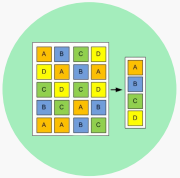
```
:Namespace INF

MASK←-1E10×~(⍳32)∘.≥⍳32
UNSQZ←{((⍴⍵),1)⍴⍵}
AVG←{UNSQZ (+/⍵)÷¯1↑⍴⍵}

FWD←{
    s←⍺
    LN←{(⍺⊃s.BLN)+⍤1⊢(⍺⊃s.WLN)×⍤1⊢diff÷⍤1⊢(1E¯5+AVG 2*⍨diff←⍵-⍤1⊢AVG ⍵)*0.5}
    MHSA←{
        q k v←((⍺⊃s.BQKV)+⍤3⊢⍵+.×⍺⊃s.WQKV)∘{0 2 1 3⍉⍵⌿⍤1⊢⍺}¨⍳3
        attn←{e÷⍤1⊢UNSQZ+/e←*⍵-⍤1⊢UNSQZ⌈/⍵}MASK[⍳2⌷⍴q;⍳2⌷⍴k]+⍤2⊢(q+.×⍤2⊢⍉⍤2⊢k)÷(¯1↑⍴k)*0.5
        (⍺⊃s.BO)+⍤1⊢((⍴⍵)⍴0 2 1 3⍉attn+.×⍤2⊢v)+.×⍺⊃s.WO
    }
    MLP←{(⍺⊃s.B2)+⍤1⊢({0.5×⍵×1+7○0.797885×⍵+0.044715×⍵*3}(⍺⊃s.B1)+⍤1⊢⍵+.×⍺⊃s.W1)+.×⍺⊃s.W2}
    BLK←{ind inp←⍵ ⋄ (ind+1) ({⍵+ind MLP ⍵ LN⍨1+2×ind}inp+ind MHSA inp LN⍨2×ind)}
    s.WH+.×⍨(2×≢s.W1)LN 1⊃BLK⍣(≢s.W1)⊢0 (s.WPE[⍳1⌷⍴⍵;]+⍤2⊢s.WTE[⍵;])
}

GEN←{inp cnt←⍵ ⋄ ⍺{⍵,(⊃⍒)⍤1⊢¯1↑[1]⍺ FWD (-(≢⍺.WPE)⌊1⌷⍴⍵)↑[1]⍵}⍣cnt⊢inp}

:EndNamespace
```

What About Machine Learning?

- Linear models, decision trees, clustering, and more
- Unglamorous but effective, cheap, and simple
- Still dominant in key industries
- Heterogeneous algorithms without an overarching theme
- Conditional statements, iterative processes, graphs, ...

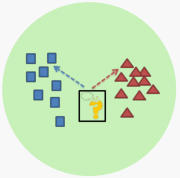


APLearn



Pre-processing

Standardization, categorical encoding, and train-test splits

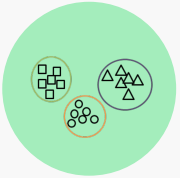


Supervised

Linear and logistic regression, vector machines, random forests, ...

Metrics

Mean absolute and squared errors, accuracy, F1 score, ...



Unsupervised

Dimensionality reduction and clustering

Design Principles

- Inspired by scikit-learn with a functional flavour
- Model and transformation states represented by record-like namespaces
- Consistent interface across `fit`, `pred`, and `trans`
- Built-in composability

Function	Role
`fit`	Learns state from data (α) given hyperparameters (ω).
`pred`	Makes predictions about data (α) using state (ω).
`trans`	Applies transformation to data (α) using state (ω).

```
]Import # APLSource

⍝ Reads data and moves target to first column for ease
(data header)←⎕CSV 'adult.csv' 0 4 1
data header←(header⍳⊂'income')⌽⍤1⍉data header

⍝ Encodes categorical features and target; target is now last
cat_names←'workclass' 'education' 'marital-status' 'occupation' 'relationship' 'race' 'gender' 'native-country'
data←data PREPROC.ONE_HOT.trans data PREPROC.ONE_HOT.fit header⍳cat_names
data←data PREPROC.ORD.trans data PREPROC.ORD.fit 0

⍝ Creates 80:20 training-validation split and separates input & target
train val←data MISC.SPLIT.train_val 0.2
(X_t y_t) (X_v y_v)←(¯1+≠⊖data) MISC.SPLIT.xy⍨⍤1 train val

⍝ Normalizes data, trains, takes argmax of probabilities, and evaluates accuracy
X_t X_v←(X_t PREPROC.NORM.fit 0)∘(PREPROC.NORM.trans⍨)⍤1 X_t X_v
st←X_t y_t SUP.LOG_REG.fit 0.01
out←0⌈⊖⍒⍤1 1⊢X_v SUP.LOG_REG.pred st
⎕←y_v MISC.METRICS.acc out
```

```
# Loads data.
df = pd.read_csv('adult.csv')

# One-hot encodes categorical columns.
cat_cols = ['workclass','education','marital-status', 'occupation', 'relationship','race','gender','native-country']
df = pd.get_dummies(df, columns=cat_cols)

# Converts target variable from string into integer.
df['income'] = (df['income'] == '>50K').astype(int)

# Splits data into training and validation sets.
train, val = train_test_split(df, test_size=0.2, shuffle=True)

# Separates independent and dependent variables.
X_t, y_t = train.drop(columns=['income']), train['income']
X_v, y_v = val.drop(columns=['income']), val['income']

# Normalizes features.
scaler = StandardScaler()
X_t = scaler.fit_transform(X_t)
X_v = scaler.transform(X_v)

# Creates model, trains, makes predictions, and computes accuracy.
log_reg = LogisticRegression(C=1 / 0.01)
log_reg.fit(X_t, y_t)
log_reg_y_hat = log_reg.predict(X_v)
print(accuracy_score(y_v, log_reg_y_hat))
```

Advantages and Drawbacks

- Algorithms seamlessly mapped from paper to APL
- Free from software noise like classes, pointers, …
- Codebase in one place—no Python + NumPy + Fortran
- Native functionality for statistics and data science

but…

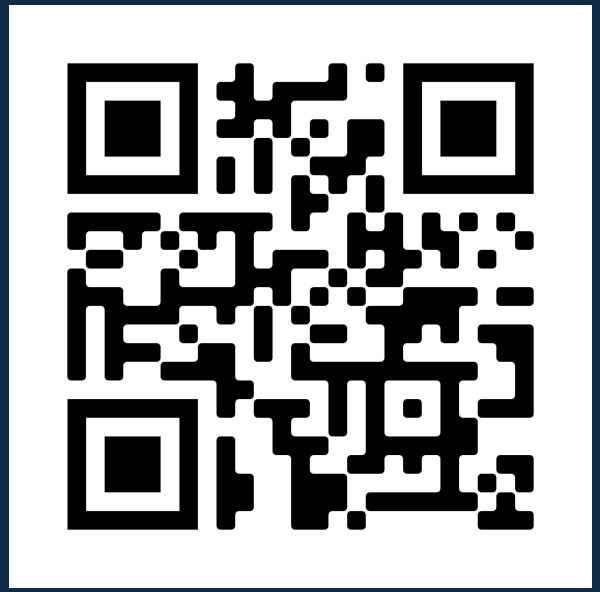
- Difficult to switch from traditional imperative mindset
- Not as array-centric as deep learning
- Slower than scikit-learn; compilation not straightforward
- Smaller ecosystem

Future Directions

- Enhanced handling of trees
- Complete Co-dfns support
- Automatic differentiation
- Advanced solvers and backends
- Rigorous unit tests
- More idiomatic APL
- Better documentation

Pull requests and issues are appreciated and welcome!

Thank you for your time!



linkedin.com/in/borna-ahmadzadeh

github.com/bobmcdear