STAT 5526

Lec 1: Introduction to Data Science

Machine Learning | Statistical Inference | Pattern Discovery

Xin Xing
VT

Decision Theory & Hold-out Method

DECISION THEORY

Framework for making optimal predictions by minimizing expected loss. Choose the decision that minimizes risk.

KEY CONCEPTS

Loss Function L(y, ŷ)

Quantifies cost of prediction errors

Risk R(f) = E[L(Y, f(X))]

Expected loss over data distribution

Bayes Optimal

Best possible decision rule

Common Loss Functions

- **Squared:** $(y - \hat{y})^2$ → regression
- **0-1 Loss:** $I(y \neq \hat{y})$ → classification
- **Absolute:** $|y - \hat{y}|$ → robust regression

HOLD-OUT METHOD

Split data into training and test sets. Train on training set, evaluate on held-out test set to estimate generalization error.

DATA SPLITTING

Test set must remain untouched until final evaluation

Training Error

Error on data used for fitting. Often optimistic.

Test Error

Error on held-out data. Estimates true risk.

CROSS-VALIDATION

K-fold CV: Rotate through K splits for more robust error estimates when data is limited.

The Bias-Variance Tradeoff

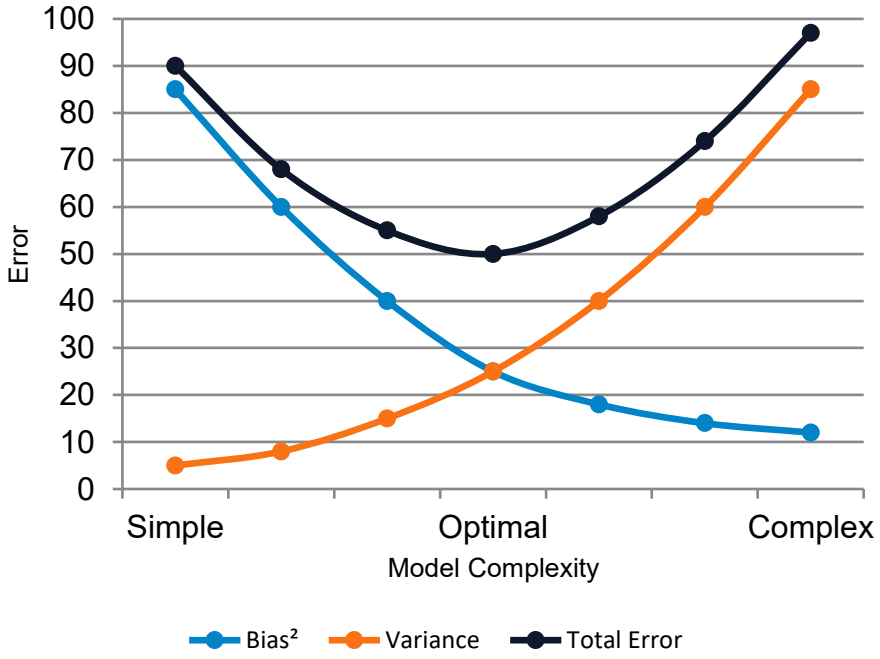
A fundamental concept explaining the tradeoff between model bias and variance in predictions.

TOTAL ERROR

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

High Bias (Underfitting)
Model is too simple, misses patterns

High Variance (Overfitting)
Model is too complex, fits noise



Regularized Regression

LASSO OBJECTIVE

$$\min \Sigma(y_i - \hat{y}_i)^2 + \lambda\Sigma|\beta_j|$$

L1

LASSO Penalty

Drives coefficients to zero, enables feature selection

L2

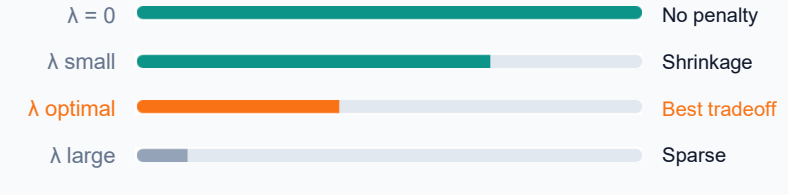
Ridge Penalty

Shrinks coefficients but never to exactly zero

Why LASSO?

- Automatic feature selection
- Prevents overfitting
- Improves interpretability
- Works with high-dimensional data

TUNING λ (LAMBDA)



Use cross-validation to select optimal λ

Decision Trees

CORE IDEA

Recursive partitioning of feature space using binary splits. Each node asks a yes/no question about a feature.

SPLITTING CRITERIA

Classification:

Gini impurity, Entropy (Information Gain)

Regression:

MSE, MAE reduction

Key Parameters

- **max_depth:** Maximum tree depth
- **min_samples_split:** Min samples to split
- **min_samples_leaf:** Min samples per leaf

Pros

- Highly interpretable
- No feature scaling needed
- Handles non-linear data
- Feature importance built-in

Cons

- Prone to overfitting
- High variance
- Unstable (small data changes)
- Biased toward dominant classes

PRUNING

Reduce overfitting by removing branches that provide little predictive power. Pre-pruning (stop early) vs Post-pruning (trim after).

FOUNDATION FOR ENSEMBLES

Decision trees are the building blocks for Random Forests, Gradient Boosting, and other powerful ensemble methods.

Boosting

CORE IDEA

Train weak learners sequentially, each focusing on mistakes of previous ones. Combine into strong learner through weighted voting.

BOOSTING PROCESS

1. Train weak model on data
2. Identify misclassified samples
3. Increase weight on errors
4. Train next model on reweighted data
5. Repeat and combine all models

KEY INSIGHT

Boosting reduces bias by iteratively correcting errors, while bagging (Random Forest) reduces variance.

POPULAR ALGORITHMS

XGBoost

Optimized gradient boosting with regularization, parallel processing, and handling missing values. Industry standard.

LightGBM

Gradient boosting with leaf-wise growth, histogram binning. Faster training, lower memory on large datasets.

CatBoost

Handles categorical features natively, ordered boosting to reduce overfitting. Great out-of-box performance.

AdaBoost

Original adaptive boosting algorithm. Simple, interpretable, but sensitive to noisy data and outliers.

Random Forest

ENSEMBLE METHOD

Combine many decision trees trained on random subsets of data and features. Final prediction by majority vote (classification) or averaging (regression).

TWO SOURCES OF RANDOMNESS

1. **Bagging (Bootstrap)**
   Each tree trained on random sample with replacement
2. **Feature Subsampling**
   Each split considers random subset of features (√p)

Key Parameters

- **n_estimators:** Number of trees (100-500)
- **max_features:** Features per split
- **oob_score:** Out-of-bag validation

WHY IT WORKS

Averaging many uncorrelated trees reduces variance without increasing bias. "Wisdom of crowds" for machine learning.

Advantages

- Reduces overfitting
- Handles high dimensions
- Robust to outliers
- Parallelizable

Limitations

- Less interpretable
- Slower than single tree
- Memory intensive
- Can overfit noisy data

FEATURE IMPORTANCE

Built-in ranking of feature importance based on impurity decrease or permutation importance across all trees.

The Multiple Testing Problem

THE PROBLEM

With m independent tests at α = 0.05:

$$P(\geq 1 \text{ false positive}) = 1 - (0.95)^m$$

FALSE POSITIVE RATE BY # OF TESTS

CORRECTION METHODS

Bonferroni Correction

$$\alpha' = \alpha / m$$

Simple but conservative. Controls FWER.

Benjamini-Hochberg (BH)

$$FDR = E[V/R]$$

Less conservative. Controls false discovery rate.

Holm-Bonferroni

Step-down procedure. More powerful than Bonferroni.

Controlled Variable Selection

THE CHALLENGE

Select important variables while controlling the rate of false discoveries. Balance discovery power with error control.

FDR IN VARIABLE SELECTION

$$FDR = E[V / \max(R, 1)]$$

V = # false discoveries (null variables selected)

R = # total discoveries (variables selected)

Why Control FDR?

- Avoid spurious findings in high dimensions
- Reproducible feature selection
- Interpretable error guarantees

KNOCKOFF FILTER

Create synthetic "knockoff" variables that mimic correlation structure but are independent of response. Compare real vs knockoff importance.

KNOCKOFF PROCEDURE

1. Generate knockoff variables $\tilde{X}$
2. Compute importance for $[X, \tilde{X}]$
3. Calculate $W_j = |Z_j| - |\tilde{Z}_j|$
4. Select variables with $W_j \geq$ threshold

Model-X
Known X distribution

Fixed-X
Linear model assumption

Conformal Inference

CORE IDEA

Distribution-free prediction intervals with guaranteed coverage, regardless of the underlying model

COVERAGE GUARANTEE

$$P(Y \in C(X)) \geq 1 - \alpha$$

Valid for any distribution, any model

How It Works

1. Split data into training and calibration sets
2. Compute nonconformity scores on calibration
3. Find quantile threshold $\hat{q}$ at level $(1-\alpha)$
4. Prediction set: $C(x) = \{y : s(x,y) \leq \hat{q}\}$

KEY ADVANTAGES

- Model-agnostic (works with any predictor)
- Finite-sample guarantees (not asymptotic)
- No distributional assumptions
- Easy to implement

Split Conformal

Simple, fast; uses held-out calibration set

Full Conformal

More efficient; computationally intensive

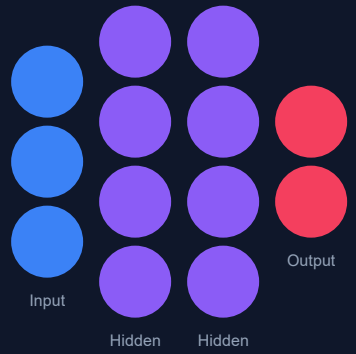
APPLICATIONS

Medical diagnosis | Anomaly detection | Risk assessment | Uncertainty quantification

Introduction to Neural Networks

NEURAL NETWORK

Layers of interconnected neurons that learn hierarchical representations



Key Components

- **Weights & biases:** Learnable parameters
- **Activation:** ReLU, sigmoid, tanh
- **Loss function:** Measures prediction error
- **Backpropagation:** Gradient computation

WHY "DEEP"?

Multiple hidden layers learn increasingly abstract features automatically

CNNs

Convolutional networks for images and spatial data

RNNs / LSTMs

Recurrent networks for sequences and time series

Transformers

Attention-based models for NLP and beyond

GANs / VAEs

Generative models for creating new data

Computer Vision
Image classification, detection

NLP
Translation, summarization

Speech
Recognition, synthesis

RL
Games, robotics

Introduction to Vibe Coding

WHAT IS VIBE CODING?

A new paradigm where you describe what you want in natural language and AI generates the code. Focus on intent, not syntax.

THE SHIFT

Traditional		Vibe Coding
`Write code`	→	`Describe intent`

Key Principles

- Describe outcomes, not implementations
- Iterate through conversation
- Review and refine AI output
- Trust the vibe, verify the results

WHY IT MATTERS

- Democratizes software development
- Faster prototyping and iteration
- Focus on problem-solving, not syntax
- Lower barrier to entry

AI CODING TOOLS

GitHub Copilot · Claude Code · Cursor · Replit AI · v0 by Vercel

BEST PRACTICES

Be specific with prompts • Break complex tasks into steps • Always review generated code • Understand what the code does • Test thoroughly

Using VS Code for R

WHY VS CODE?

Modern, lightweight editor with excellent R support. Great for combining R with other languages and version control.

SETUP STEPS

1. Install R from CRAN
2. Install VS Code
3. Install R extension (REditorSupport)
4. Install languageserver in R
5. Install httpgd for plots

R Console Commands

```
install.packages("languageserver")
install.packages("httpgd")
```

KEY EXTENSIONS

- **R** - Language support & debugging
- **R Debugger** - Breakpoints & stepping
- **Quarto** - R Markdown alternative
- **GitHub Copilot** - AI assistance

KEYBOARD SHORTCUTS

`Ctrl+Enter`	Run current line/selection
`Ctrl+Shift+S`	Source entire file
`F2`	Go to definition

Plots
httpgd viewer panel

Terminal
Integrated R console

Variables
Environment viewer

Key Takeaways

Decision Theory
Loss functions & hold-out method

Bias-Variance Tradeoff
Balance complexity to minimize error

Regularized Regression
Ridge, LASSO, Elastic Net

Decision Trees
Interpretable recursive partitioning

Boosting
Sequential error correction

Random Forests
Bagging ensemble of trees

Multiple Testing
FDR control & corrections

Controlled Variable Selection
Knockoff filter for FDR

Conformal Inference
Distribution-free prediction sets

Neural Networks
Deep learning fundamentals

Vibe Coding
AI-assisted development

VS Code for R
Modern R development setup

Data Science combines statistical theory, machine learning algorithms, and rigorous inference to extract reliable insights from data.