Deployment Model Monitoring and Stability Analysis

Jeremy Seeman

University of Chicago
Center for Data Science and Public Policy

jeremys@uchicago.edu

July 30, 2018

From model selection to deployment

Model deployment:

The process of designing a machine learning process to predict outcomes autonomously with limited to no human interaction.

What does a typical "deployed" machine learning model look like?

- Automation of time dependence
  - Time-dependent features are recomputed with more recent data
  - Models are periodically retrained with more recent data
- Automation of model selection and evaluation
  - Feature contributions are re-optimized
  - Hyperparameters are re-optimized

Deployment fallacies

What are we implicitly assuming in this deployment setting?

- Data assumptions:
  - Data collection methods and ETL processes produce stable streams of data, and changes to these processes are directly observable
  - New data is always "more predictive or important" than old data
  - Model output does not produce feedback effects that alter feature or label distributions
- Optimization assumptions:
  - Model re-optimization is sufficient to ensure consistent estimation, even under changes in conditional outcome distributions
  - The optimal set of hyperparameters is stationary in time
  - The optimization procedure is scalable

Problem: none of these are generically true in practice!

Why is this a bad thing?

Model effectiveness can be sorely limited under these assumptions!

- Data cleaning methods may no longer be valid
- Re-optimization may be prone to overfitting on newer data
- Older data that's systematically excluded from models may still be informative
- Model may not support data from new populations (i.e. model does not generalize)
- Partners who use model output for decision making can introduce new confounding variables

Goal of deployment monitoring: change detection

Time-dependence of modeling processes is the root cause!

Time-dependence of ETL processes, feature distributions, model structure, and model performance explicitly affect our modeling goals.

Primary goal is **change detection**, minimizing two kinds of errors:

- *Type I error*: our estimated model changes in response to new data, but the change does not reflect ground truth
- *Type II error*: our model fails to change in response to new data that reflects a change in ground truth

What makes this a hard problem?

No ground truth: difficult to distinguish between "meaningful" and "meaningless" model change

Model changes are not individually causal: there are often multiple plausible explanations for why models change over time

Model changes are not independent: time-dependence can introduce new interactions between nearly every component of our model

The model monitoring toolkit

Our approach:

Provide a toolkit for monitoring ML systems as a whole, from raw data collection to predictions and scores

- Deterministic methods: used to catch systems-level "errors"
  - Ex: broken ETL processes fails to load raw input data
  - Ex2: feature generation produces mathematically inconsistent values
- Probabilistic methods: used to flag statistical "warnings"
  - Ex: feature distributions changes significantly after new data collection
  - Ex2: entity-level outcomes are inconsistent relative to one another
  - Ex3: model optimization "results" (ex: optimal hyperparameters, residual structure) show evidence of overfitting, lack of robustness or generalizability, bias, etc.
  - Ex4: feature contributions and dependence are inconsistent across models with different hyperparameters

Outline

Data setup: trajectories

All model inputs and outputs are time-dependent, indexed by $t \in [T]$:

- At each time, we observe N entities (any repeated observation):
  - Each entity generates K features $X_{n,k,t}$, $k \in [K]$, $n \in [N]$
  - Each entity generates one outcome / label: $Y_{n,t}$, $n \in [N]$
- No assumptions made about entity or feature independence

Modeling goal:

Use supervised learning to model $Y_{n,t}|X_{n,t}$ for all $n \in [N]$, $t \in [T]$

Additional assumptions:

- Labels are directly observable (often not true in public policy scenarios)
- Number of monitored entities does not change over time (counterexamples: survival models in epidemiology, dropouts in panel studies)
- Individual models have a fixed feature definitions

Example: CMPD Early Intervention System (EIS)

(we will use examples from this project throughout the slides)

Modeling goal:

Given police officers' dispatch history, arrest history, etc., predict officers that are likely to have an adverse interaction with the public.

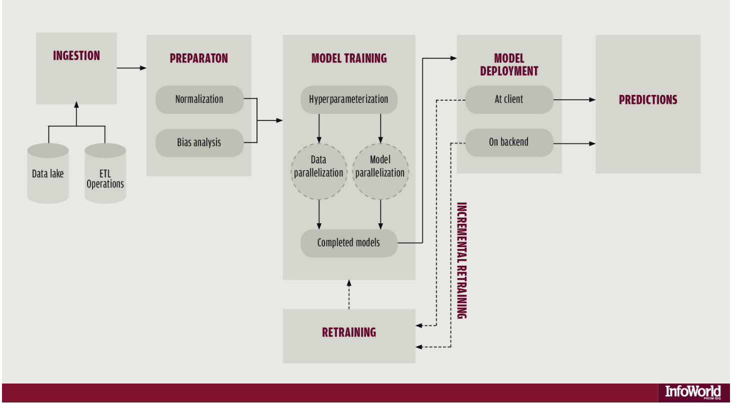
Adverse interactions can be defined as:

- unjustified uses of force
- officer injuries
- preventable accidents
- sustained complaints

Models are retrained with new data daily, with many features aggregated in a rolling windows (ex: total arrests in last month)

First priority: deterministic systems-level issues

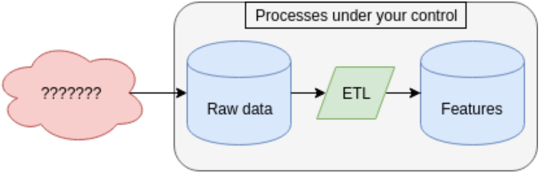
Determinstic errors in ETL processes propagate through ML systems, which means every possible process must be integration tested.



Designing pipeline integration tests

Feature generating ETL processes can have explicit tests for consistency:

- Number of inserted or updated database rows are reasonable
- Features do not contain illegal values
- Entity identifiers are properly joined to existing data
- Feature calculations successfully incorporate new data



Known feature dependence

In many modeling contexts, features are distribution point estimates or other aggregation estimates of an underlying latent random variable.

Example: CMPD feature blocks

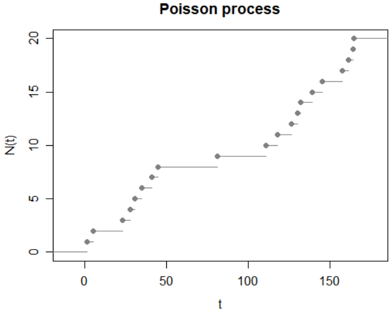
- Latent variable: count process of theft arrests per officer
- Features: over (1 day, 1 week, 1 month, 1 year, 5 years), aggregate (count of all arrests, average rate of arrests)

	feature
1	arrests_id_p1d_arrestscrimetype_theft_avg
2	arrests_id_p1d_arrestscrimetype_theft_sum
3	arrests_id_p1m_arrestscrimetype_theft_avg
4	arrests_id_p1m_arrestscrimetype_theft_sum
5	arrests_id_p1w_arrestscrimetype_theft_avg
6	arrests_id_p1w_arrestscrimetype_theft_sum
7	arrests_id_p1y_arrestscrimetype_theft_avg
8	arrests_id_p1y_arrestscrimetype_theft_sum
9	arrests_id_p5y_arrestscrimetype_theft_avg
10	arrests_id_p5y_arrestscrimetype_theft_sum

Examples of feature generating processes

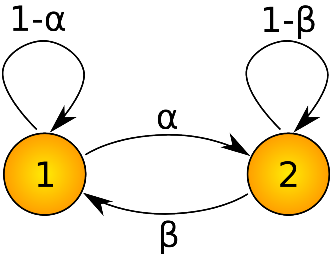
Count processes: a process that "counts" the number of times a given "event" occurs over time

Example: Poisson process



State-transition process: a process where entities can be in a single "state" and transition to different states as time progresses

Example: Markov chain



Pipeline testing with known feature dependence

If features have a known block structure, we have more deterministic constraints on our incoming data.

Example: CMPD feature blocks

- Total number of theft arrests should be nondecreasing in time
- If $t_1 < t_2$ and the average number of theft arrests including t_2 is positive, then the average number of theft arrests including t_1 through t_2 is also positive

	feature
1	arrests_id_p1d_arrestscrimetype_theft_avg
2	arrests_id_p1d_arrestscrimetype_theft_sum
3	arrests_id_p1m_arrestscrimetype_theft_avg
4	arrests_id_p1m_arrestscrimetype_theft_sum
5	arrests_id_p1w_arrestscrimetype_theft_avg
6	arrests_id_p1w_arrestscrimetype_theft_sum
7	arrests_id_p1y_arrestscrimetype_theft_avg
8	arrests_id_p1y_arrestscrimetype_theft_sum
9	arrests_id_p5y_arrestscrimetype_theft_avg
10	arrests_id_p5y_arrestscrimetype_theft_sum

Shared probabilistic tools

Probabilistic methods will share a number of common tools for computational analysis of random variables:

1. Distribution quantization: estimating the distribution of a random variable from a finite set of parameters
2. Distribution differences: calculating measures of distance between distributions
3. Change detection tools: estimating change breakpoints in time series

This is because we'll be analyzing many different random variables:

- Feature distributions at the entity level: $X_{n,k,t}$
- Predicted outcomes / scores: $\mathbb{E}[Y_{n,t}|X_{n,t}]$
- True outcomes / labels: $Y_{n,t}$
- Model performance: $L(Y_t, X_t)$
- Loss function contributions: $\epsilon_{n,t}$

Estimating probability distributions

Change detection depends on feature distribution estimates:

$$F_{k,t}(s) = \mathbb{P}(X_{k,t} \leq s) \quad \text{where} \quad X_{1,k,t} \dots X_{N,k,t} \sim X_{k,t}$$

Empirical distributions can uniformly estimate any distribution:

$$\hat{F}_{k,t}(s) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{X_{n,k,t} \leq s\}$$

As $N \to \infty$, $\hat{F}_{k,t} \to F_{k,t}$ strongly and uniformly (Glivenko-Cantelli theorem).

For distributions with known discrete support $\mathcal{S}$, $\hat{F}_{k,t}$ is neither memory-intensive nor computationally expensive.

Continuously-supported distribution methods

...but for continuously supported distributions, empirical distributions are infeasible, both statistically and computationally.

Joint goals of optimization:

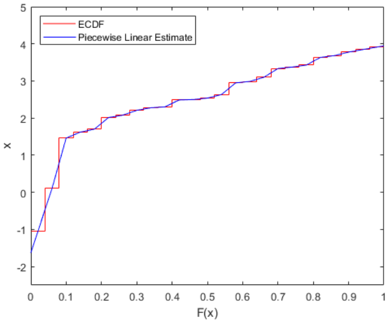
- Vector quantization: represent a continuous distribution with a discrete choice of parameters
- Functional form: optimize the functional form to match the sample as closely as possible

We will (briefly) review some common quantization methods in the next few slides.

Example 1: direct interpolation methods

Define breakpoints in s (histogram-style) or $[0,1]$ (quantile-style) and interpolate between point estimates of distribution functions:

- Choice of breakpoints can encode "soft" prior information:
  - Often more robust than specifying explicit parametric models
  - Ex: heavy-tailed distributions, tracking extremal quantiles
- Histograms are a naive (i.e. constant) kind of interpolation



Example 2: generative distribution models

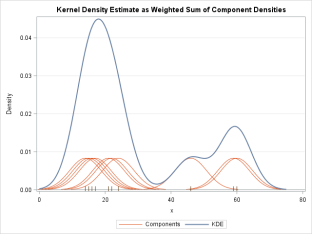
Kernel density estimation involves replacing point estimtaes with combinations of smooth functions:

$$\hat{F}_{k,t}(s) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{C_h} K\left(\frac{\|s - X_{i,k,t}\|}{h} \right)$$

Allows user to define expectations for local behavior based on kernel choice and bandwidth.

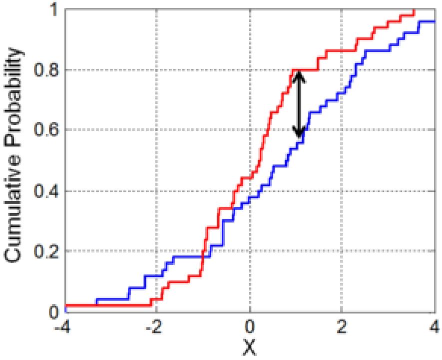
Still need a sparse representation of $\hat{F}_{k,t}$!

- Parametric: mixture models, empirical Bayesian methods
- Nonparametric: clustering, network-based autoencoders



CDF distances

Given two distributions, we need to calculate a functional distance:



There is no "right" way to calculate functional distances!

Comparing two CDFs

One possible approach: think about functional differences in CDFs in L^p-norm:

$$d_p(X_{k,t}, X_{k,t+\tau}) = \int_{-\infty}^{\infty} |F_{k,t}(s) - F_{k,t+\tau}(s)|^p \; ds$$

Can also consider differences in the quantile function (inverse CDF), which has unique properties using transport theory [1]:

$$W_p(X_{k,t}, X_{k,t+\tau}) = \int_0^1 |F_{k,t}^{-1}(s) - F_{k,t+\tau}^{-1}(s)|^p \; ds \quad \text{(in 1D)}$$

These are reminiscent of many classical distribution tests:

- $p = 1$: Mallow's distance (also known as "Earth-mover's distance")
- $p = 2$: energy distance (used in goodness-of-fit tests)
- $p = \infty$: total variation (used in Kolmogorov-Smirnov tests)

Why can't we use KL divergence or other f-divergences?

Many standard distribution distances used in parametric analysis (such as KL divergence) have the same functional form:

$$D_f(X_{k,t}||X_{k,t+\tau}) = \int_\Omega f\left(\frac{dX_{k,t}}{dX_{k,t+\tau}}\right) dX_{k,t+\tau}$$

Example: $f(x) = x\log(x)$ yields KL divergence.

Problems with this approach:

- When the support of your two distributions differ, the distance may be "infinite" or otherwise undefined
- The distance is not a true metric, since it is not symmetric

Entity-level rank correlations

Repeated measurements at the entity level yield additional information!

Given $X_{1,k,t} \dots X_{N,k,t}$, define their entity ranks $X_{(i),k,t} \equiv R_{i,k,t}$, and then for two matrices A, B define:

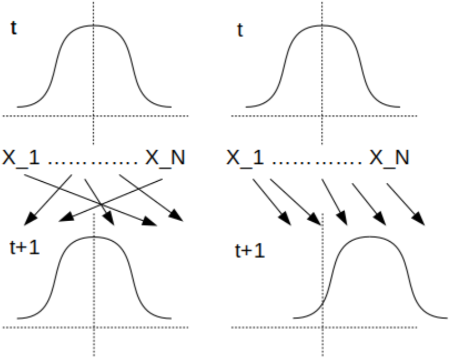
$$\rho(X_{k,t}, X_{k,t+\tau}) = \sum_{i,j=1}^{n} A_{ij} B_{ij} \Big[\sum_{i,j=1}^{n} A_{ij}^2 \sum_{i,j=1}^{n} B_{ij}^2 \Big]^{-1}$$

Choices for A_{ij}, B_{ij} yield many classical nonparametric statistics [3]:

- Spearman Rank Correlation: $A_{ij} = R_{j,k,t} - R_{i,k,t}$
- Kendall's Tau: $A_{ij} = \mathrm{sgn}(R_{j,k,t} - R_{i,k,t})$

Necessity of rank correlation and distribution changes

Distribution-level changes and entity-level changes need to be measured simultaneously for proper detection:



Left: rank correlation change without distribution change | Right: distribution change without rank correlation change

What constitutes distribution change?

Measures of association have ambiguous relationships with time-dependent trends, making it hard to characterize!

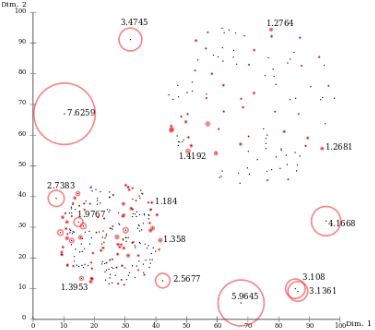
- Point estimates of association are noisy, and can have large time-dependent variance
- Associations typically decay as τ increases, but the presence, rate, and form of decay are ambiguous
- "Slow" changes in distributions over time are harder to detect than "fast" point changes

Point changes: outlier detection

Many non-parametric methods exist for outlier detection that are more robust than simple extremal statistics.

Example: local outlier factor (LOF) analysis

Use k-means distance to aggregate outlier weight based on nearest sample points.



Long term effects: trend filtering

Change point identification relies on modeling methods to "denoise" time series and identify different time-dependent regimes.

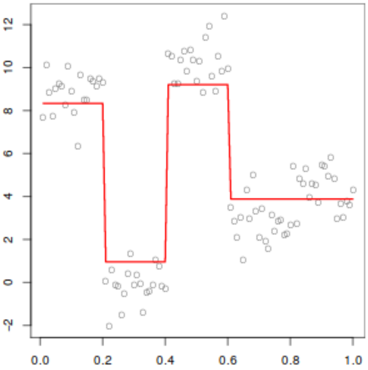
Example: ℓ_0 trend filtering

[7] Optimize:

$$\arg\min_{y} \left[\frac{1}{2} \|y - x\|_2^2 + \lambda \|Dx\|_1 \right]$$

where:

$$D = \begin{bmatrix} -1 & 1 & \\ \ddots & \ddots & \ddots \\ & -1 & 1 \end{bmatrix}$$



Time-dependence of association decay

Any normalized association metric that maps to $[0, 1]$ can be converted into an integrated autocorrelation time (IAT):

$$\tau_\rho = \sum_{t=-\infty}^{\infty} \rho(X_t, X_{t+\tau}) \approx 1 + 2\sum_{t=1}^{T} \rho(X_t, X_{t+\tau}) \quad \text{for sufficiently large } T$$

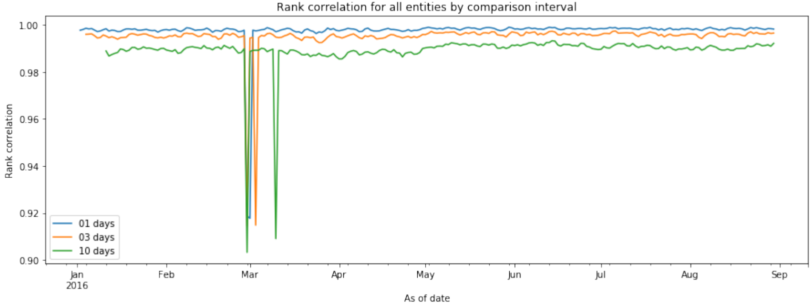
IAT can be useful as a proxy for the aggregate time at which past samples have an impact on future samples (under the chosen correlation metric)

NB: IAT is computable for stationary sampling processes, but may be effectively infinite for small T or highly non-stationary sampling proceses

Distribution change example

Example: CMPD officer scores

- Point changes are easy to detect with a large number of entities and frequent sampling
- Rank autocorrelation decays as τ increases, but decay is slow so IAT may not be finite



How should distribution changes relate to each other?

In general, we want to verify that our model has desirable statistical properties in the deployment setting:

- Consistency: model results are reproducible and will ensure high probability convergence to ground truth
- Generalizability: models remain accurate as new entity and feature samples are added
- Model structure stability: features that contribute meaningfully to model output do not change rapidly over time
- Residual stability: residuals are relatively uniform, and residual structure does not indicate different performance for different groups (especially protected socioeconomic classes)

Algorithmic robustness

Informal definition

A model is *robust* if and only if whenever a training sample is close to a testing sample, the training error is close to the testing error.

Formal: a model $\mathcal{M}$ with training set $S \subset Z \equiv X \times Y$ is $(J, \epsilon(S))$-robust if Z can be partitioned into J disjoint sets $\{C_i\}_{i=1}^{J}$ such that for all $s \in S$:

$$z, s \in C_i \implies |L_{\mathcal{M}}(s) - L_{\mathcal{M}}(z)| \leq \epsilon(S)$$

[4, 8]

Generalizability is equivalent to robustness!

Informal definition

A model is *generalizable* or *scalable* if and only if the performance of the model is not impacted by increasing training and testing sample sizes.

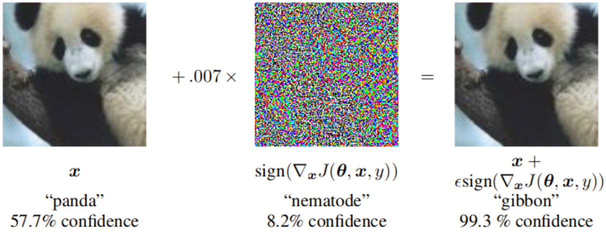
Formal: a model $\mathcal{M}$ generalizes w.r.t. S if, given a sequence of increasing size training sets $s_n \supset S$ and testing sets t_n we have:

$$\limsup_n \left\{ \mathbb{E}_t[L_{\mathcal{M}}(t_n)] - \mathcal{L}_{\mathcal{M}}(s_n) \right\} \leq 0$$

Theorem: asymptotic behaviors of robustness and generalizability are equivalent [4, 8]

Violations of robustness

Robustness is hard to directly measure, but the opposite is somewhat easier: there's plenty of active research on how to generate shortest-distance adversarial examples [2, 6]



Examples like these demonstrate that many algorithms in practice fail to generalize, and thus fail on larger datasets!

Feature contributions

How do we estimate the effect of a given feature under the model? Again, many different options!

Many options look at empirical plots of different distributions:

- Partial dependence plots
- Individual conditional expectations

However, since these are functional forms, they require the same numerical tools for distribution differences.

Feature importances

Relative feature importance is often determined by permutation test [5]:

1. For each feature $k \in [K]$:
   1. Randomly permute $X_{n,k,t}$ in k for all $n \in [N]$
   2. Retrain and observe the change in a target function (examples: loss function, conditional information, etc.) $\Delta_{k,t}$
2. Re-normalize $\Delta_{k,t}$ to get feature importances $I_{k,t}$ s.t. $\sum_{k=1}^{K} I_{k,t} = 1$

Explicit methods may replace full permutation tests if model form is known and mathematically interpretable.

- Useful when permutation tests are computationally expensive
- Ex: Random forests, mean decrease in Gini impurity

Should feature importances be stable?

Highly ambiguous!

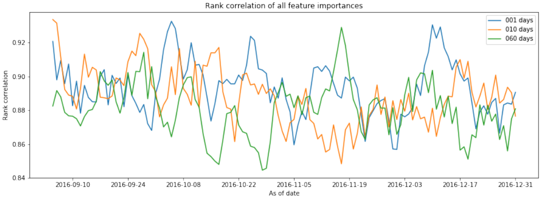
- Because of how feature importances are calculated, they are often weak estimators of structural dependence
- More consistent estimators are often computationally prohibitive

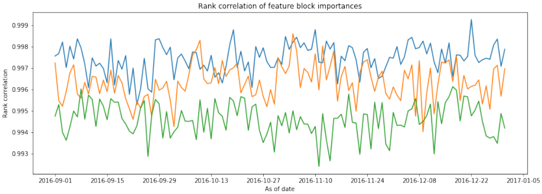
Practical workaround: aggregating feature importances by blocks

- Sum feature importances that correspond to the same feature block
- Aggregated importances reflect model dependence on latent variables

Example: CMPD block feature importances

For a large number of features, individual feature importances are noisy, but block feature importances are more stable.





Multiple models

New setup: consider a set of models $m \in [M]$, each with hyperparameters $h_{q,m}$ for $q \in [Q]$; assume that at each susccessive retraining, the model which minimizes the loss function m_t^* has hyperparameters $h_{q,t}^*$

Unintended side effects:

- Residual structure can vary between models
- Robustness between train-test sets is ambiguous; decreasing test loss does not guarantee improved robustness
- As Q increases:
  - Optimal models become more prone to overfitting
  - Sets of hyperparameters $h_{q,t}$ may be statistically indistinguishable

Residual structure and bias

Let $\epsilon_{n,m,t}$ be the contribution to a loss function for a given entity-model-time combination:

- Different entity subsets N_a, N_b may have different residual structures, ex: $\mathbb{E}[\epsilon_{n,m,t}|n \in N_a] \neq \mathbb{E}[\epsilon_{n,m,t}|n \in N_b]$
- More complex models are more likely to have non-uniform residual structures, which have unknown effects on N_a vs. N_b
- Example: N_a and N_b are different socioeconomic bias categories (race, gender, income, etc.)

Remember:

Unless otherwise specified, most loss functions are uniform in entity and do not control for generalized residual structures.

Proxying overfitting using train-test error

m_t^* compared with m_t may have structural differences in train-test errors.

Toy example:

Type	$\mathbb{E}_{m_t^*}[\epsilon_{n,m,t}]$	$\mathrm{Var}_{m_t^*}(\epsilon_{n,m,t})$	...	$\mathbb{E}_{m_t}[\epsilon_{n,m,t}]$	$\mathrm{Var}_{m_t}(\epsilon_{n,m,t})$
train	.05	.06	...	.18	.02
test	.10	.08	...	.20	.03

A sub-optimal model in point estimates of loss may be more easily scalable if the training and testing residuals are more similarly distributed!

Open statistical questions

Many interactions are difficult to characterize, and are open questions in statistics research:

- Interactions of temporal feature importance measures with distribution changes
- Generalized ensemble methods for enforcing uniform loss contributions
- Loss function optimization with non-deterministic effects

How do we apply this information?

Always frame model monitoring goals in terms of large-scale project goals.

Example considerations:

- How will project partners use model output?
  - Explicit monitoring for subsets of at-risk populations
  - Application of interventions under limited resources
- How will project partners alter model input?
  - Incorporation similarly-structured data from new structures
  - Comparison of models for different labels of same latent phenomenon

Generic modeling guidelines

Worried about this?	**Use this**	**Look for this**
ETL process not producing feature output deterministically	Pipeline integration tests	Unexpected behavior from failing integration tests
ETL process not producing clean or meaningful data	Entity-level feature distributions	Discontinuities in rank correlations and distances
Feature distributions are drifting over time	Entity-level feature distributions	Time dependence of distribution distances
Predictions are inconsistent	Entity-level score distribution	Rank correlation between entity scores
Loss function is non-uniform in entities	Sub-population measures of residual independence	Bias in residual distributions
Model is not scalable	Extremal contributions to the loss function	Frequency and severity of adversarial examples
Hyperparameter optimization is causing overfitting	train and test loss contribution distributions for different hyperparameters	Time dependence of error distributions, and train-test differences

Additional topics we would cover if we had time

- Vector quantization of distributions: finding low-dimensional representations of distributions
- Latent variable feature representations: count processes, state-transition processes, network-based processes
- Modeling inter-block dependence: empirical copulas, numerical estimates of optimal transport
- Algorithmic pseudo-robustness under non-ergodic settings
- Feature importance alternatives and their stability
- Optimization techniques for avoiding adversarial examples

References I

Vladimir I Bogachev and Aleksandr V Kolesnikov. "The Monge-Kantorovich problem: achievements, connections, and perspectives". In: *Russian Mathematical Surveys* 67.5 (2012), p. 785. URL: http://stacks.iop.org/0036-0279/67/i=5/a=R01.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: (2014), pp. 1–11. ISSN: 0012-7183. DOI: 10.1109/CVPR.2015.7298594. arXiv: 1412.6572. URL: http://arxiv.org/abs/1412.6572.

William H Kruskal. "Ordinal Measures of Association". In: *Journal of the American Statistical Association* 53.284 (1958), pp. 814–861. ISSN: 01621459. URL: http://www.jstor.org/stable/2281954.

Sayan Mukherjee et al. "Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization". In: *Advances in Computational Mathematics* 25.1-3 (2006), pp. 161–193. ISSN: 10197168. DOI: 10.1007/s10444-004-7634-z.

References II

P. Radivojac et al. "Feature selection filters based on the permutation test". In: *Machine Learning: Ecml 2004, Proceedings* 3201 (2004), pp. 334–346. ISSN: 03029743. DOI: 10.1007/978-3-540-30115-8_32.

Christian Szegedy et al. "Intriguing properties of neural networks, Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus". In: *arXiv preprint* (2014), pp. 1–10. arXiv: 1312.6199v4. URL: https://arxiv.org/pdf/1312.6199v4.pdf.

Ryan J. Tibshirani. "Adaptive piecewise polynomial estimation via trend filtering". In: *Annals of Statistics* 42.1 (2014), pp. 285–323. ISSN: 00905364. DOI: 10.1214/13-AOS1189. arXiv: arXiv:1304.2986v2.

Huan Xu and Shie Mannor. "Robustness and generalization". In: *Machine Learning* 86.3 (2012), pp. 391–423. ISSN: 08856125. DOI: 10.1007/s10994-011-5268-1. arXiv: 1005.2243.