DSAN 5300 Exam 2 Study Guide

DSAN 5300 Staff

Thursday, March 13, 2025

Part 1: Data-Splitting

Here the idea is that, we will be evaluating your understanding of different ways to split a dataset: **how** the split is performed and **why** we might prefer a particular splitting approach in a particular data-analysis scenario.

Train-Test Split $\rightsquigarrow$ Generalizability

This is not a 5300 topic on its own—you learned it in DSAN 5000!—but is nonetheless important to take note of here since its the basis for **why** we use Cross-Validation for model evaluation in 5300 (e.g., for tuning the hyperparameters of the models we look at each week).

It turns out to be an… oddly deep topic (hence why Jeff spent too long ranting about it in his section during Week 4), but the key rationale for **why** we split our full dataset into a **Training Set** and a **Test Set** boils down to the fact that our **fundamental goal** in statistical learning is to…

- Find a **prediction function** $\widehat{f}(x)$ that…
- Does a good job predicting **labels** y_i based on **features** x_i…
- For data (x_i, y_i) that has **not (yet) been observed**

This contrasts with the more "naïve" goal of optimizing predictions of y_i relative to already-observed **training** datapoints.

So, under this framing of our goal, we *set aside* a small proportion (usually 20%) of the data as our **Test Set**, which we do *not* look at while *training* the model, so that it can instead be used to evaluate the model's performance on **unseen** data.

Cross-Validation

Given that setup, then, the different versions of Cross-Validation all fall under the general rubric of: generating "mini" training sets (sub-training sets) and test sets (validation sets), as *smaller* subsets of the full training data, in order to **estimate** how well a given model might perform on **unseen** data before we move to the *final* test-set-based evaluation once we have settled upon and trained a final version of our model.

The key versions of Cross-Validation to know are:

- The Validation Set approach,
- Leave-One-Out Cross-Validation (LOOCV), and
- K-Fold Cross-Validation.

For example, you should understand the relationship between K-Fold CV and LOOCV: that LOOCV is exactly just K-Fold CV with K set to be equal to the number of datapoints n.

Part 2: Stepwise Model Selection

- Best Subset Selection
- Forward Stepwise Selection
- Backward Stepwise Selection

Here you can focus on, e.g., what are the relative **strengths** and **weakness** of these approaches? In particular, you should understand how:

- Best Subset Selection is guaranteed to find an optimal subset of features, but is usually prohibitively slow/inefficient, since it searches over all $\binom{p}{1} + \binom{p}{2} + \cdots + \binom{p}{p}$ possible subsets of features, whereas
- Forward and Backward Stepwise Selection are computationally tractable but at the cost of **not** being guaranteed to find an optimal subset of features.

Part 3: Regularization and L^p-Norms

Here the most helpful things to study are as follows:

Which particular norms are used in which particular regularization methods?

- Lasso penalizes model complexity via a penalty λ on $\|\beta\|_1$, the L^1 norm of $\beta = (\beta_0, \beta_1, ..., \beta_p)$
- Ridge Regression penalizes model complexity via a penalty λ on $\|\beta\|_2^2$, the squared L^2 norm of β
- Elastic Net Regularization "averages" these approaches, in a sense, by penalizing model complexity via *two* penalty values:
  - λ_1 as a penalty on $\|\beta\|_1$ and \$ λ_2 as a penalty on $\|\beta\|_2^2$

What do the different L^p constraints "look like", in terms of how they constrain the full space of possible model parameters β?

Here, as a helpful/fun way to prepare, is a practice problem for this part:

Assume we are estimating a regression with two parameters β_1 and β_2, both real numbers so that the space of estimates is $\mathbb{R}^2$ (we can imagine β_1 as the x-axis in this space and β_2 as the y-axis). Each L^p-norm induces a **"unit circle"** within this space, defined to be the **set of points** $\beta = (\beta_1, \beta_2)$ for which $\|\beta\|_p = 1$.

Since the L^2 norm corresponds to "standard" Euclidean distance, for example, the shape of the unit circle it induces is the shape we typically already call "the unit circle" in geometry:

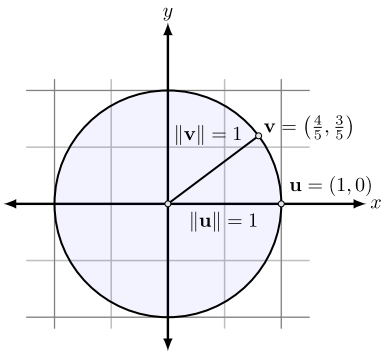

Figure 1: From this slide

The L^1 norm, however, induces a different unit circle: the set of all points exactly 1 unit away from the origin in L^1 space looks like a diamond:

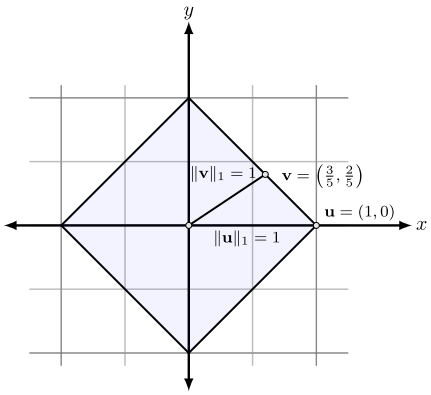

Figure 2: From this slide

Practice Problem 3A:

- What do the unit circles look like for values of p where $0 < p < 1$? (This was discussed in the Google Space, but it can be helpful to think of *why* the unit circle looks this way)
- The unit disks (the spaces enclosed by the unit circles) in L^1 and L^2 space are **convex**, meaning that if you pick any two points β_A and β_B within the space and draw a line between them, any point along this line is also within the space.[1] Are the unit circles produced when $0 < p < 1$ also **convex**?

Practice Problem 3B:

The definition of the L^p norm,

$$\|\beta\|_p = \left(\sum_{j=1}^{J} |\beta_j|^p \right)^{\frac{1}{p}},$$

works fine for deriving (e.g.) the L^2, L^1, and $L^{2/3}$ norms. But there is one additional widely-used norm, the L^∞ norm, that we can't exactly derive by "plugging in" ∞, but *can* easily derive by taking the **limit** as $p \to \infty$:

$$\|\beta\|_\infty \stackrel{\text{def}}{=} \lim_{p \to \infty} \|\beta\|_p$$

- Show (or, just, try your best to show! Or look up!) that this limit can be evaluated to arrive at the closed-form expression $\|\beta\|_\infty = \max\{\beta_0, \beta_1, \dots, \beta_J\}$
- More importantly: what does the **unit circle** induced by this norm look like? Try to draw it on an xy-plane, the same way you would draw a unit circle or the diamond induced by the L^1 norm.

Part 4: Basis Functions

In Week 7 we introduced the notion of a set of **basis functions** $\{b_0(X), b_1(X), \dots, b_D(X)\}$ as a method for writing different types of regressions (e.g., linear regression, polynomial regression, piecewise regression, and regression splines) in a single form.

A degree-d polynomial regression, for example, can be thought of as a simple linear regression on d basis functions $\{b_0(X) = 1, b_1(X) = X, b_2(X) = X^2, \dots, b_d(X) = X^d\}$, so that estimating parameters β_0 through β_d gives us

$$Y = \beta_0 b_0(X) + \beta_1 b_1(X) + \cdots + \beta_d b_d(X) = \beta_0 + \beta_1 X + \cdots + \beta_d X^d,$$

the standard form for a degree-d polynomial regression.

[1] We haven't talked about convex optimization in this class, but it turns out to be a very important feature in terms of whether or not we can efficiently **optimize** a given function within a given space…

For this portion, if we provide you with a "true" Data-Generating Process $f(x)$, you should be able to identify whether or not performing regression on a particular set of basis functions $\mathcal{B} = \{b_0(X), b_1(X), ...\}$ would allow you to **learn** the true DGP $f(x)$.

For example, if the true DGP underlying a given dataset was

$$f(x) = 2^x,$$

but the set of basis functions used for a regression was $\mathcal{B} = \{x, x^2\}$, then this regression would **not** be able to learn the true DGP, since there are no numeric constants β_0, β_1, and β_2 that it could learn which would produce

$$2^x = \beta_0 + \beta_1 x + \beta_2 x^2.$$

However, if the true DGP was

$$f(x) = \log_2(x) + 3,$$

and we employed the basis functions $\mathcal{B}' = \{x, x^2, \ln(x)\}$, we **could** learn the true DGP using regression, since learning the numeric constants $\hat{\beta}_0 = 3$, $\hat{\beta}_1 = 0$, $\hat{\beta}_2 = 0$, and $\hat{\beta}_3 = \frac{1}{\ln(2)}$ would recover $f(x)$:

$$\log_2(x) + 3 = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \hat{\beta}_3 \ln(x) = 3 + 0 + 0 + \frac{\ln(x)}{\ln(2)}$$

Part 5: Splines

The basic approach to data modeling underlying the use of splines is as follows:

1. "Chop" the feature space (the domain of $\mathbf{x}$) at K points $\Xi = \{\xi_1, \xi_2, ..., \xi_k\}$[2], called "knot points", thus producing $K + 1$ separate pieces of the original domain of $\mathbf{x}$
2. Model each of these $K + 1$ pieces individually, using tools from Weeks 1-6
3. *Join* the $K + 1$ resulting prediction functions together in a **smooth** way

Here the term "smooth" in Step 3 is typically (99.9% of the time) *operationalized* to mean that the final joined-together prediction function should have a well-defined **second derivative** at each knot point ξ_k. This is why, when a data scientist says "spline" in general, they usually are referring specifically to **cubic splines**:

[2]The symbol Ξ is the capitalized form, and ξ the lowercased form, of the Greek letter "xi", pronounced like "ksy"—like saying "sigh" immediately after a quick "k" sound.

Table 1: First and second derivatives of polynomials evaluated at an arbitrary point ξ_k

Degree	Polynomial	Derivatives at $x = \xi_k$
0	$f(x) = c_0 x^0 = c_0$	$f'(\xi_k) = 0 f''(\xi_k) = 0$
1	$f(\xi_k) = c_0 x^0 + c_1 x^1 = c_0 + c_1 x$	$f'(\xi_k) = c_1 f''(\xi_k) = 0$
2	$f(x) = c_0 x^0 + c_1 x^1 + c_2 x^2$	$f'(\xi_k) = c_1 + 2c_2\xi_k f''(\xi_k) = 2c_2$
3	$f(x) = c_0 x^0 + c_1 x^1 + c_2 x^2 + c_3 x^3$	$f'(\xi_k) = c_1 + 2c_2\xi_k + 3c_3\xi_k^2 f''(\xi_k) = 2c_2 + 6c_3\xi_k$

From this table we can see how degree-3 **cubic** polynomials are the "simplest" (in terms of lowest degree) polynomials for which we could learn coefficients c_0 through c_3 that would enable **joining** different pieces together to have a well-defined second derivative at **all** knot points ξ_k[3]. And, in fact, we don't even need to learn 4 separate coefficients c_0, c_1, c_2, and c_3 to achieve this: by looking at the form of $f''(x)$ in the final row of Table 1, we see that by just learning a coefficient c_3 on the cubic term (x^3) at each knot point, we can ensure that the regression solution $\widehat{c_3}$ (for example, the OLS estimate $\widehat{\beta}_j$ for a term $\beta_j \xi_k^3$ on the RHS of a regression model)

This tells us (for reasons that we talked about in more depth during lecture) that we can join our $K+1$ separately-modeled "pieces" together in a **smooth** way by **adding** the **truncated power basis functions**

$$\mathcal{B}_{\text{TP}} = \{(x-\xi_1)_+^3, (x-\xi_2)_+^3, \dots, (x-\xi_K)_+^3\}$$

to any existing regression model with existing basis functions $\mathcal{B}$, where the $+$ in the subscript after the parentheses denotes the truncated power function:

$$(x-\xi)_+^3 \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x-\xi \leq 0 \\ (x-\xi)^3 & \text{if } x-\xi > 0 \end{cases}$$

Thus, for example, if we were fitting a regression model using one of the example bases given in the previous part:

$$\mathcal{B}' = \{x, x^2, \ln(x)\},$$

but decided that in fact we want to fit this model **separately** for datapoints with $x_i \leq 100°\text{C}$ and $x_i > 100°\text{C}$, we can "automatically" achieve this by performing a new regression with basis functions

[3] In a Calculus class, we learn (as basically a helpful rule-of-thumb definition) that a function $f(x)$ has a "well-defined" second derivative at a given point ξ if $\lim_{x\to\xi^+} f''(x) = \lim_{x\to\xi^-} f''(x)$, i.e., if the second derivative has the same value approaching ξ from below and approaching ξ from above. If we go on to take a Real Analysis class this picture gets complicated a bit to arrive at a fully-workable definition (by bringing in infima and suprema), but the intuition from Calculus class should be fine for this class!

$$\mathcal{B}'' = \{x, x^2, \ln(x), (x-100)_+^3\}.$$

This "forces" the regression to now estimate some parameter β_4 as a coefficient for $(x-100)_+^3$, ensuring that resulting prediction function joins the two "pieces" at $x = 100°\text{C}$ in a smooth way.

Given these pieces, some takeaways for yall in terms of what you can study is as follows:

How should the knot points ξ_k themselves be chosen?

- If we have a **pre-existing** reason (with respect to our theory or hypothesis that we're bringing to the data) to expect **different behavior** in different regions of the domain of some feature **x**, we can opt to "manually" place knot points to separate the full domain into these regions
- In the above example, $\xi_1 = 100°\text{C}$ may be chosen because we're studying **water**, which (our theory tells us) undergoes a **phase change** at that temperature.
- In the absence of this type of pre-existing reason, however, we more-commonly choose both the **number** of knot points K and their **locations** $\{\xi_1, \ldots, \xi_k\}$ via **Cross-Validation**

How exactly does adding the truncated power function $(x-\xi_k)_+^3$ as a new basis function allow us to "split" the original regression into pieces (constrained to join together smoothly)?

- The parenthetical part we already discussed above (that smoothness is "enforced" by the 3 in the exponent), but the splitting is a separate "feature" of these truncated power bases.
- For intuition, rather than thinking of these as separate/new features, it can help to think of them as "slope modifiers": if our prediction function is

  $$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 (x_i - \xi_1)_+^3,$$

  then the slope of this prediction function will be $\hat{\beta}_1$ up until the point $x = \xi_1$, after which the slope will become $\hat{\beta}_1 + \hat{\beta}_2$.