[Lecture 1]

1 What is this course about?

Consider the process of building a statistical or machine learning model. We typically first collect training data, then fit a model to that data, and finally use the model to make predictions on new test data.

In theory and in practice, we generally rely on the train and test data coming from the same distribution, or at least being closely related in some way. However, there are several ways this could fail to be the case:

1. The data collection process itself could be noisy and thus not reflect the actual underlying signal we wish to learn. For instance, there could be human error in labelling or annotation, or measurement error due to imperfect sensors.

2. There could be distributional shift, due to changes in the world over time or because we seek to deploy the model in some new situation (a language model trained on news articles but deployed on twitter). There might also be noise e.g. due to a sensor failing.

Robustness concerns what we should do when the train and test distribution are not the same, for any of the reasons above. There are two underlying perspectives influencing the choice of material in this course. First, we are generally interested in *worst-case* rather than average-case robustness. For instance, when handling data collection errors we will avoid modeling the errors as random noise and instead build procedures that are robust to any errors within some allowed family. We prefer this because average-case robustness requires the errors to satisfy a single, specific distribution for robustness guarantees to be meaningful, while a goal of robustness is to handle unanticipated situations that are difficult to model precisely in advance.

Second, we will study robustness in *high-dimensional* settings. Many natural approaches to robustness that work in low dimensions fail in high dimensions. For instance, the median is a robust estimate of the mean in one dimension, but the per-coordinate median is a poor robust estimator when the dimension is large (its error grows as $\sqrt{d}$ in d dimensions). We will see that more sophisticated estimators can substantially improve on this first attempt.

Complementary to robustness is the idea of *model mis-specification.* When the true distribution p^* lies within our model family, many robustness issues are less severe: we can rely on the model to extrapolate to new settings, and we can often get well-calibrated uncertainty estimates. This motivates the second focus of the course, *nonparametric modeling*, where we consider broad function classes (e.g. all smooth functions) that are more likely to be correctly specified. Another connection between nonparametrics and robustness is that we often want robust methods to work for any distribution within some large, infinite-dimensional class.

Overarching framework. Most robustness questions can be cast in the following way: We let p^* denote the true test distribution we wish to estimate, and assume that training data $X_1, \dots, X_n$ is sampled i.i.d. from some distribution $\tilde{p}$ such that $D(\tilde{p}, p^*) \leq \epsilon$ according to some discrepancy D. We also assume that $p^* \in \mathcal{G}$, which encodes the distributional assumptions we make (e.g. that p^* has bounded moments or tails, which is typically necessary for robust estimation to be possible). We benchmark an estimator $\hat{\theta}(X_1, \dots, X_n)$ according to some cost $L(p^*, \hat{\theta})$ (the test error). The diagram in Figure 1 illustrates this.

This framework captures both of the examples discused at the beginning. However, it will be profitable to think about each case separately due to different emphases:

1. For corrupted training data, we think of $\tilde{p}$ as being corrupted and p^* as being nice.

2. For distributional shift, we think of $\tilde{p}$ and p^* as both being nice (but different).

Additionally, since both $\tilde{p}$ and p^* are nice for distributional shift, we should have greater ambitions and seek to handle larger differences between train and test than in the corruption cases.

Training robustness. Designing robust estimators for training corruptions usually involves reasoning about what the real data "might have" looked like. This could involve operations such as removing outliers, smoothing points away from extremes, etc. Unfortunately, many intuitive algorithms in low dimensions

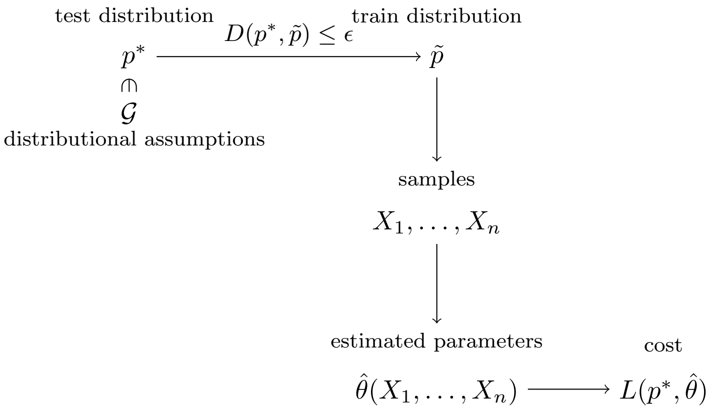


Figure 1: Framework for studying robust learning.

achieve essentially trivial bounds in high dimensions. We will show how to achieve more meaningful bounds, focusing on three aspects:

1. good dependence of the error on the dimension,
2. good finite-sample bounds,
3. computational tractability.

Each of these aspects turns out to require new machinery and we will devote roughly equal space to each.

Distributional shift. For distributional shift, we often seek invariant features or structure that can transfer information from the train to test distributions. We can also counteract distribution shift by training on more diverse data. Finally, we can use model uncertainty to infer out-of-distribution error, but these inferences can be fraught if the model is mis-specified.

Table 1: Comparison of different robust settings.

Train robustness	Distributional shift
p^* nice	p^* and $\tilde{p}$ both nice
undo corruptions	invariant features
	diverse training data

Nonparametric modeling. This brings us to nonparametric methods. The simplest way to be nonparametric is through the model–using a rich class such as smooth functions, or neural networks. This mitigates model mis-specification but raises new statistical challenges. Familiar phenomena such as the central limit theorem cease to hold, and error rates are instead governed by the eigenvalue spectrum of an infinite-dimensional kernel. Aside from the model, we can also be nonparametric at inference time; an example is the bootstrap, which constructs confidence intervals that are more robust to model mis-specification than classical parametric tests.

Summary. We will cover each of training time robustness, distribution shift, and nonparametric modeling, drawing connections as we go. The first third will focus on training time, also building up the statistical machinery needed to prove generalization bounds. Then, we will shift focus to model mis-specification, and to nonparametric methods as a remedy. These including kernel regression, the bootstrap, and neural networks, and we will both see how they mitigate model mis-specification and how to analyze their generalization performance. Finally, we will turn to distribution shift, and see that nonparametric models are often robust to distribution shift when trained on the right data. Training robustness will receive a largely theoretical

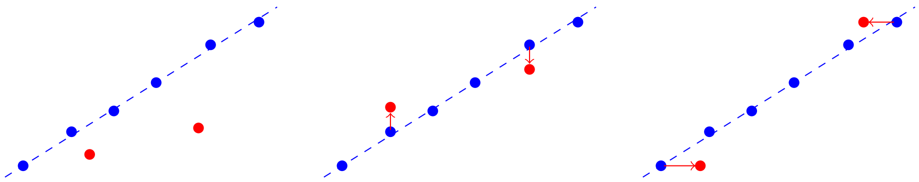


Figure 2: Possible corruptions to be robust to. Left: data contains outliers. Middle: outputs are perturbed (process noise); Right: inputs are perturbed (measurement error).

treatment, while for nonparametrics and distribution shift we will see a mix of theoretical and empirical results.

2 Training Time Robustness

We will start our investigation with training time robustness. As in Figure 1, we observe samples $X_1, \dots, X_n$ from a corrupted training distribution $\tilde{p}$, whose relationship to the true (test) distribution is controlled by the constraint $D(\tilde{p}, p^*) \le \epsilon$. We additionally constrain $p^* \in \mathcal{G}$, which encodes our distributional assumptions.

Note that this setting corresponds to an *oblivious* adversary that can only apply corruptions at the population level (changing p^* to $\tilde{p}$); we can also consider a more powerful *adaptive* adversary that can apply corruptions to the samples themselves. Such an adversary is called adaptive because it is allowed to adapt to the random draw of the samples points $X_1, \dots, X_n$. Formally defining adaptive adversaries is somewhat technical and we defer this to later.

Figure 2 illustrates several ways in which a training distribution could be corrupted. In the left panel, an ϵ fraction of real points have been replaced by outliers. This can be modeled by the discrepancy $D(p, q) = \mathsf{TV}(p, q)$, where TV is the *total variation distance*:

$$\mathsf{TV}(p, q) \overset{\text{def}}{=} \sup\{|p(E) - q(E)| \mid E \text{ is a measurable event}\}. \tag{1}$$

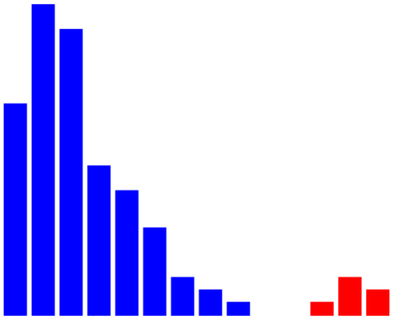
If p and q both have densities then an equivalent characterization is $\mathsf{TV}(p, q) = \frac{1}{2} \int |p(x) - q(x)| dx$.

In the middle and right panels of Figure 2, either the inputs or outputs have been moved slightly. Both operations can be modeled using *Wasserstein distances* (also called earthmover distances), which we will discuss later. For now, however, we will focus on the case of handling outliers. Thus for the next several sections our discrepancy will be the total variation distance $D = \mathsf{TV}$.

2.1 Robustness to Outliers in 1 Dimension

First consider mean estimation in one dimension: we observe n data points $x_1, \dots, x_n \in \mathbb{R}$ drawn from $\tilde{p}$, and our goal is to estimate the mean $\mu = \mathbb{E}_{x \sim p^*}[x]$ of p^*. Accordingly our loss is $L(p^*, \theta) = |\theta - \mu(p^*)|$.

The following histogram illustrates a possible dataset, where the height of each bar represents the number of points with a given value:



Are the red points outliers? Or part of the real data? Depending on the conclusion, the estimated mean could vary substantially. Without further assumptions on the data-generating distribution p^*, we cannot rule out either case. This brings us to an important principle:

With no assumptions on the distribution p^*, robust estimation is impossible.

In particular, we must make assumptions that are strong enough to reject sufficiently extreme points as outliers, or else even a small fraction of such points can dominate the estimate of the mean. For simplicity, here and in the next several sections we will assume that we directly observe the training distribution $\tilde{p}$ rather than samples $x_{1:n}$ from $\tilde{p}$. This allows us to avoid analyzing finite-sample concentration, which requires introducing additional technical tools that we will turn to in Section **??**.

Assumption: bounded variance. One possible assumption is that p^* has bounded variance: $\mathbb{E}_{x \sim p^*}[(x - \mu)^2] \leq \sigma^2$ for some parameter σ. We take $\mathcal{G} = \mathcal{G}_{\mathsf{cov}}(\sigma)$ to be the set of distributions satisfying this constraint.

Under this assumption, we can estimate μ to within error $\mathcal{O}(\sigma\sqrt{\epsilon})$ under TV-perturbations of size ϵ. Indeed, consider the following procedure:

Algorithm 1 `TrimmedMean`

```
Remove the upper and lower $(2\epsilon)$-quantiles from $\tilde{p}$ (so $4\epsilon$ mass is removed in total).
Let $\tilde{p}_{2\epsilon}$ denote the new distribution after re-normalizing, and return the mean of $\tilde{p}_{2\epsilon}$.
```

To analyze Algorithm 1, we will make use of a strengthened version of Chebyshev's inequality, which we recall here (see Section **??** for a proof):

Lemma 2.1 (Chebyshev inequality)**.** *Suppose that p has mean μ and variance σ^2. Then, $\mathbb{P}_{X \sim p}[X \geq \mu + \sigma/\sqrt{\delta}] \leq \delta$. Moreover, if E is any event with probability at least δ, then $|\mathbb{E}_{X \sim p}[X \mid E] - \mu| \leq \sigma\sqrt{\frac{2(1-\delta)}{\delta}}$.*

The first part, which is the standard Chebyshev inequality, says that it is unlikely for a point to be more than a few standard deviations away from μ. The second part says that any large population of points must have a mean close to μ. This second property, which is called *resilience*, is central to robust estimation, and will be studied in more detail in Section **??**.

With Lemma 2.1 in hand, we can prove the following fact about Algorithm 1:

Proposition 2.2. *Assume that $\mathsf{TV}(\tilde{p}, p^*) \leq \epsilon \leq \frac{1}{8}$. Then the output $\hat{\mu}$ of Algorithm 1 satisfies $|\hat{\mu} - \mu| \leq 9\sigma\sqrt{\epsilon}$.*

Proof. If $\mathsf{TV}(\tilde{p}, p^*) \leq \epsilon$, then we can get from p^* to $\tilde{p}$ by adding an ϵ-fraction of points (outliers) and deleting an ϵ-fraction of the original points.

First note that all outliers that exceed the ϵ-quantile of p^* are removed by Algorithm 1. Therefore, all non-removed outliers lie within $\frac{\sigma}{\sqrt{\epsilon}}$ of the mean μ by Chebyshev's inequality.

Second, we and the adversary together remove at most a 5ϵ-fraction of the mass in p^*. Applying Lemma 2.1 with $\delta = 1 - 5\epsilon$, the mean of the remaining good points lies within $\sigma\sqrt{\frac{10\epsilon}{1-5\epsilon}}$ of μ.

Now let ϵ' be the fraction of remaining points which are bad, and note that $\epsilon' \leq \frac{\epsilon}{1-4\epsilon}$. The mean of all the remaining points differs from μ by at most $\epsilon' \cdot \sigma\sqrt{\frac{1}{\epsilon}} + (1 - \epsilon') \cdot \sigma\sqrt{\frac{10\epsilon}{1-5\epsilon}}$, which is at most $(1 + \sqrt{10})\frac{\sqrt{\epsilon}}{1-4\epsilon}\sigma$. This is in turn at most $9\sigma\sqrt{\epsilon}$ assuming that $\epsilon \leq \frac{1}{8}$. □

Optimality. The $\mathcal{O}(\sigma\sqrt{\epsilon})$ dependence is optimal, because the adversary can themselves apply the same trimming procedure we do, and in general this will shift the mean of a bounded covariance distribution by $\mathcal{O}(\sigma\sqrt{\epsilon})$ while keeping the covariance bounded.

Alternate assumptions. The key fact driving the proof of Proposition 2.2 is that any $(1 - \epsilon)$-fraction of the good points has mean at most $\mathcal{O}(\sigma\sqrt{\epsilon})$ away from the true mean due to Chebyshev's inequality (Lemma 2.1), which makes use of the bound σ^2 on the variance. Any other bound on the deviation from the mean would yield an analogous result. For instance, if p^* has bounded kth moment, then the $\mathcal{O}(\sigma\sqrt{\epsilon})$ in Lemma 2.1 can be improved to $\mathcal{O}(\sigma_k \epsilon^{1-1/k})$, where $(\sigma_k)^k$ is a bound on the kth moment; in this case Algorithm 1 will estimate μ with a correspondingly improved error of $\mathcal{O}(\sigma_k \epsilon^{1-1/k})$.

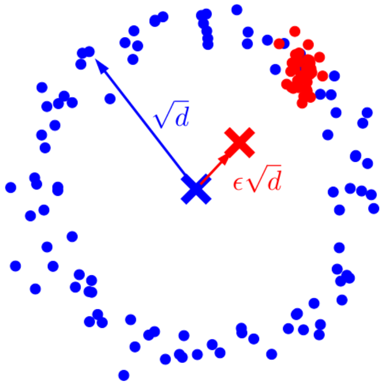


Figure 3: The outliers can lie at distance $\sqrt{d}$ without being detected, skewing the mean by $\epsilon\sqrt{d}$.

2.2 Problems in High Dimensions

In the previous section, we saw how to robustly estimating the mean of a 1-dimensional dataset assuming the true data had bounded variance. Our estimator worked by removing data points that are too far away from the mean, and then returning the mean of the remaining points.

It is tempting to apply this same idea in higher dimensions—for instance, removing points that are far away from the mean in ℓ_2-distance. Unfortunately, this incurs large error in high dimensions.

To see why, consider the following simplified example. The distribution p^* over the true data is an isotropic Gaussian $\mathcal{N}(\mu, I)$, with unknown mean μ and independent variance 1 in every coordinate. In this case, the typical distance $\|x_i - \mu\|_2$ of a sample x_i from the mean μ is roughly $\sqrt{d}$, since there are d coordinates and x_i differs from μ by roughly 1 in every coordinate. (In fact, $\|x_i - \mu\|_2$ can be shown to concentrate around $\sqrt{d}$ with high probability.) This means that the outliers can lie at a distance $\sqrt{d}$ from μ without being detected, thus shifting the mean by $\Theta(\epsilon\sqrt{d})$; Figure 3 depicts this. Therefore, filtering based on ℓ_2 distance incurs an error of at least $\epsilon\sqrt{d}$. This dimension-dependent $\sqrt{d}$ factor often renders bounds meaningless.

In fact, the situation is even worse; not only are the bad points no further from the mean than the good points in ℓ_2-distance, they actually have the same probability density under the true data-generating distribution $\mathcal{N}(\mu, I)$. There is thus no procedure that measures each point in isolation and can avoid the $\sqrt{d}$ factor in the error.

This leads us to an important take-away: *In high dimensions, outliers can substantially perturb the mean while individually looking innocuous.* To handle this, we will instead need to analyze entire populations of outliers at once. In the next section we will do this using *minimum distance functionals*, which will allow us to avoid the dimension-dependent error.