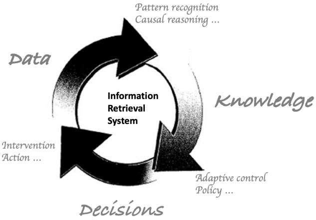
Tutorial: Theoretical Tools for Designing Modern Information Retrieval System (Part1)

Author: *Da Xu, Chuanwei Ruan*

Contact: {daxu5180,ruanchuanwei}@gmail.com



About the authors

Da Xu

Manager of Machine Learning @ Walmart Labs

Stats@Berkeley

Chuanwei Ruan

Senior Machine Learning Engineer @ Instacart

Stats@Stanford

In the past three years…

- Published more than a dozen papers on theoretical investigation and application scenarios of modern RL systems (including *NeurIPS, ICML, ICLR, KDD, WSDM, WWW* …)
- Organized and delivered several sessions and keynote talks on the frontier of RL research

We are interested in…

- Unifying the theoretical properties and production practice of modern IR methodologies
- Driving real-world productions from the principled understandings
- Establishing a systematic view of modern IR via pattern recognition, decision making, and causal inference

About this tutorial

In my team, theory and practice are combined: everything works and we all know why ;-)

Information retrieval is not:

- everything collaborative filtering
- download a data, hack a Git repo, and train a deep learning model
- writing SQL scripts, deploying a linear model, and running A/B testings
- publishing papers bragging about reinforcement learning

It's all of them!

About this tutorial

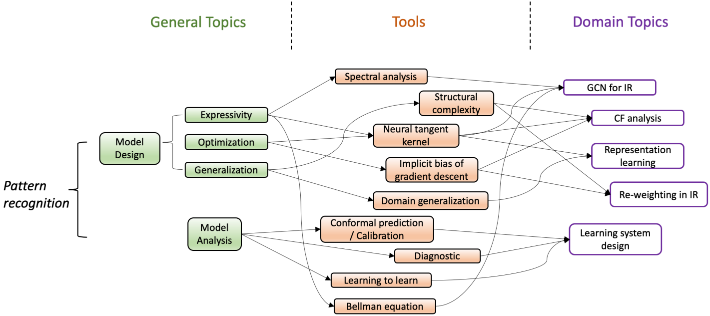
The contents we present here is more than sufficient for a full-semester grad course, so we aim to provide readers with:

- the big picture rather than technical details;
- necessary intuitions & heuristics rather than rigorous justifications;
- rooms for improvement rather than how to improve them exactly.

After this tutorial, **John Snow** still don't know how to crack most of the challenges (and neither do we), but he will be motivated to give another try now that he knows something.

Figure 1: Me at day 1 of my job.

❶ Part 1: Pattern Recognition (Model design & Model analysis)



1. **Part 1: Pattern Recognition** (Model design & Model analysis)

IR Pattern Recognition Setting

Pattern recognition generally concerns with learning the patterns from the observed data, such that they can *generalize* to the unseen testing data.

Let $\mathcal{I} = \{e_1, \ldots, e_k\}$ be the set of entities (e.g. queries, ads, items), and $\mathcal{U} = \{u_1, \ldots, u_q\}$ be the set of users. IR tasks could be *entity-to-entity* or *entity-to-user*.

Please make yourself comfortable imaging your most familiar use case.

Entities or users can possess *features* $X_e, X_u \in \mathcal{X}$, which might be:

1. one-hot encoding such as in the collaborative filtering (CF) setting.
2. numerical vector which is the content-based setting.

When no confusion rises, we use $X_{u,e}$ to denote the joint features.

IR Pattern Recognition Setting

Modern machine learning relies heavily on representation learning. Let $\phi \in \Phi : \mathcal{X} \to \mathbb{R}^d$ be the **presentation mapping**, such that $\phi(e) := \phi(X_e)$. It could be either:

- *pre-trained embedding*, which means the mapping is fixed;
- *(hidden) representation layer* with its own underlying parameterization.

The feedback $Y_{u,e}$ (or $Y_{e,e}$) is categorical, *and starting here our tutorial will become significant different from what your ordinary ML textbooks.*

Definition (IR Feedback)

The feedback data is **implicit-unlabelled** if $Y_{u,e} \in \{?, 0, 1\}$, and is k-scale **explicit-unlabelled** if $Y_{u,e} \in \{?, 0, \ldots, k\}$.

IR Pattern Recognition Setting

We view pattern recognition in IR as *classification with missing data*:

- Why classification? Because predicting both implicit (click) and explicit (rating) can be converted to **finding decision boundaries**.
- Why missingness? Users do not interact with what is not exposed (**exposure bias**), and are more likely to interact with what they found favorable (**selective bias**).

The training data are given by the triplets: $\mathcal{D}_{\text{trn}} = \{(u, e, Y_{u,e})\}$, drawn (*not necessarily independently*) from P_{trn}. We denote its support by *supp*(P_{trn}). The same for $\mathcal{D}_{\text{trn}}$, P_{test}, and *supp*(P_{test}).

IR Pattern Recognition Setting

A moment of truth: we **cannot** expect $P_{\text{trn}} = P_{\text{test}}$, nor even $supp(P_{\text{test}}) \subset supp(P_{\text{trn}})$ in IR:

1. In regular unbiased evaluation, testing data is drawn from a hypothetical *uniform-exposure domain* P_{unif}, who have a much larger support;
2. For real-world deployment, the testing data of interest is the *deployment domain* (P_{deploy}) generated by the candidate IR system, which will have mismatch with P_{trn} unless nothing new is recommended.

Definition (IR pattern recognition)

Pattern recognition in IR aims to learn decision boundaries from $\mathcal{D}_{\text{trn}}$ (e.g. implicitly-unlabelled) that generalize well to P_{test} (e.g. the deployment domain) with $P_{\text{trn}} \neq P_{\text{test}}$.

IR Pattern Recognition Setting

We use such as $f(\theta; u, e)$, $f \in \mathcal{F}$ to denote the pattern recognition model. Its decision boundary is given by: $F(\theta)$. When coupled with representation learning, we use the shorthand: $f \circ \phi(u, e)$.
In IR, the most prevalent solution for learning patterns from data is probably **empirical risk minimization** (**ERM**):

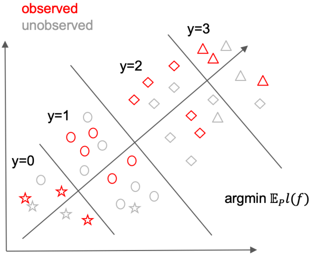
$$\min_{f \in \mathcal{F}} \mathbb{E}_P \ell(f), \textit{where}\, \mathbb{E}_P \ell(f) := \mathbb{E}_{(u,e,Y_{u,e}) \sim P} \ell\big(f(u,e), Y_{u,e}\big).$$

An important extension is weighted ERM that aims at:

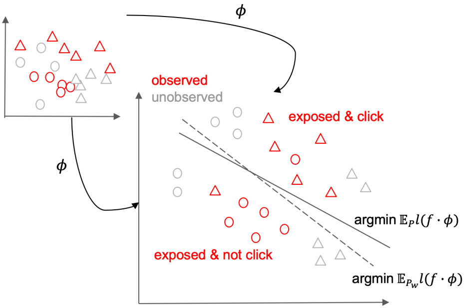
$$\mathbb{E}_{P_w} \ell(f) := \mathbb{E}_{(u,e,Y_{u,e}) \sim P} \ell\big(f(u,e), Y_{u,e}\big) \cdot w(u,e),$$

where P_w is *shifted* by the weighting function $w(\cdot)$.
Essentially, many hope that the weighting function can handle $P_{\text{trn}} \neq P_{\text{test}}$. We will show why this may not be a good idea.

IR Pattern Recognition Setting



Overlapped support
Ranking decision boundaries

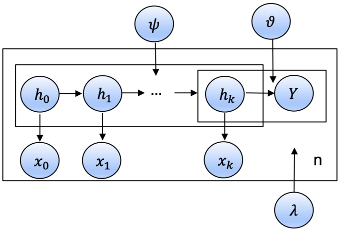


Non-overlapped support
Positive-negative decision boundaries

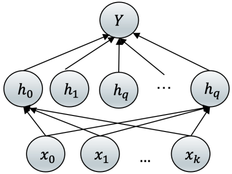
The two cultures of data science

A recurrent theme in AI.

Generative



Discriminative



- Exactly characterize the **generative process**
- Plausible data distributions described by the *graphical model*
- Generally non-convex, need Bayesian optimization technique (MCMC, variational inference, etc), model-aware, **inference** ready
- Establishing **consistency** in recovering the ground truth model, usually falls into the asymptotic regime (e.g. M-estimators)

Inductive bias

Expressivity

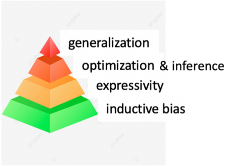
Optimization

Generalization

- Describe a collection of **response surfaces**
- Usually considered as *black box*, especially when over-parameterized
- Also non-convex, use model agnostic gradient descent (GD), **inference** not directly available
- How well the training performance can **generalize** to the unseen testing data

The two cultures of data science

Achieve the best of both worlds? Use our pyramid of model design:



Take linear model for example: $p(Y_{u,e} = 1) = \sigma\big(\langle \theta, X_{u,e} \rangle\big)$:

- **Inductive bias**: describe linear response surfaces;
- **Expressivty**: linear combination of features;
- **Optimization & Inference (diagnostic)**: convex optimization, model examined by linearity, homosedasticity, independence, normality.
- **Generalization**: lead to linear interpolation within training domain and linear extrapolation outside.

Teaser: shallow embedding with SGNS

Things get complicated very quickly moving beyond linear models :(
Consider perhaps the simplest neural network model in IR that trains entity embeddings $\phi(e)$ using the *skip-gram negative sampling* (SGNS) algorithm (**item2vec**):

$$p(Y_{e,e'}=1)=\sigma\big(\langle\phi(e),\tilde{\phi}(e')\rangle\big),\; p(Y_{e,e'}=0)=\sigma\big(-\langle\phi(e),\tilde{\phi}(e')\rangle\big)$$
$$\ell_{e,e'}=-\log p(Y_{e,e'}=1)+k\cdot\mathbb{E}_{\tilde{e}\sim Neg(\mathcal{I})}\big[\log p(Y_{e,\tilde{e}}=0)\big].$$

The representation mappings are given by the embedding matrices $\Phi=\big[\phi(e_1),\ldots,\phi(e_{|\mathcal{I}|})\big]$ and $\tilde{\Phi}=\big[\tilde{\phi}(e_1),\ldots,\tilde{\phi}(e_{|\mathcal{I}|})\big]$. The empirical risk is given by $R(\Phi,\tilde{\Phi})=\sum_{e,e'}\ell_{e,e'}$.

Teaser: shallow embedding with SGNS

It looks like a simple factorization with a non-linear activation:

$$\begin{bmatrix} Y_{11} & \cdots \\ \cdots & \cdots \end{bmatrix} = \sigma\left(\begin{bmatrix} \phi(e_1) \\ \vdots \end{bmatrix} [\tilde{\phi}(e_1), \ldots] \right), \tag{1}$$

but $\langle \phi(\cdot), \tilde{\phi}(\cdot) \rangle$ is not recovering Y! Suppose $\tilde{\Phi}$ is given, then:

$$\nabla_{\phi(e_1)} R(\Phi) = \tilde{\Phi}\text{diag}(\Lambda) \underbrace{\Big\{ \sigma\big([Q_{1,1}, \ldots, Q_{1,|\mathcal{I}|}] \big) - \sigma\big(\langle \phi(e_1), \tilde{\Phi} \rangle \big) \Big\}}_{\text{non-linear error term, with } Q_{i,j} = \log \frac{p_{i,j}}{p_i p_j} + c}.$$

For illustration, if the loss function is given by the least square: $R_{ls}(\Phi) := \big\| Y - \Phi\tilde{\Phi} \big\|_2^2$, then:

$$\nabla_{\phi(e_1)} R_{ls}(\Phi) = \tilde{\Phi}\text{diag}(\Lambda) \underbrace{\Big\{ [Y_{1,1}, \ldots, Y_{1,|\mathcal{I}|}] - \langle \phi(e_1), \tilde{\Phi} \rangle \Big\}}_{\text{linear error term}}.$$

Teaser: shallow embedding with SGNS

Comparing the error terms, we observe:

- SGNS projects error terms to the gradient non-linearly;
- the actual factorization objective is $Q_{i,j}$ instead of $Y_{i,j}$.

Recently, Xu et al. 2021[1] shows that simultaneously optimizing $\Phi, \tilde{\Phi}$ amounts to solving:

$$\min_{\Phi,\tilde{\Phi}} D_{KL}\Big(p(Y|\Phi,\tilde{\Phi})\|p(Y|Q)\Big),$$

where D_{KL} is the KL divergence. The expression is referred to as **sufficient dimension reduction**, as the information of $Y|Q$ is preserved in the optimal sense. Therefore, we reach the conclusion:

1. SGNS tries to express $Q_{i,j} = \log \frac{p_{i,j}}{p_i p_j}$: the discounted co-occurence probability;
2. during optimization, $\phi(e)$ converges to the sufficient dimension reduction of $\big[Q_{e,1}, \ldots, Q_{e,|\mathcal{I}|}\big]$.

[1] Theoretical Understanding of Product Embedding for Ecom ML, WSDM'21

General methodology for analyzing expressivity

Investigating the expressivity of discriminative models is difficult because:

1. the initializations are often random;
2. the parameters are changing during the GD optimization;
3. models are heavily over-parameterized. (why is this challenging?)

Increasing model size d as the number of samples grow n invalidates classical statistical efficiency analysis.

Example (Linear discriminative analysis in high dimension)

Suppose a random vector $X \in \mathbb{R}^d$ is drawn from one of two Gaussian distributions P_1 and P_2 with mean μ_1, μ_2 and identity covariances. A natural decision rule Ψ is: $\log \frac{P_1(X)}{P_2(X)}$.
The decision rule reduces to the error function: $Err(\Psi) = \Phi(-\lambda/2)$, where $\lambda = \|\mu_1 - \mu_2\|_2$, and Φ is the CDF of standard Gaussian.
However, μ_1, μ_2 may need to be estimated from data and thus the decision rule $\hat{\Psi}$. Using the estimated version, the error function becomes:
$Err(\hat{\Psi}) = \Phi\left(\frac{\lambda}{2\sqrt{\lambda^2 + 2d/n}}\right)$!

General methodology for analyzing expressivity

In general, we wish to find a quantity that is invariant to GD optimization, and at the same time independent of the d/n ratio.
Note that by first-order Tyler expansion, it holds:

$$f\big(\theta; u, e\big) = f\big(\theta^{(0)}; u, e\big) + \big\langle \theta - \theta^{(0)}, \nabla f\big(\theta^{(0)}; u, e\big)\big\rangle + \mathcal{O}(\sqrt{1/d}).$$

Remark: we can always get rid of the intercept term by reparameterizing $f(\theta; \cdot) = g(\theta_1; \cdot) - g(\theta_2; \cdot)$.
Since $\theta^{(0)}$ is constant after initialization, we have:

$$f\big(\theta; u, e\big) \overset{d\to\infty}{\approx} \big\langle \theta, \nabla f\big(\theta^{(0)}; u, e\big)\big\rangle + C,$$

which resembles the kernel regression, e.g.

$$f\big(\theta; u, e\big) = \big\langle \theta, \phi(u, e)\big\rangle,$$

where ϕ is the **feature lift** of the RKHS induced by the kernel: $K\big(u, e; u', e'\big) = \big\langle \phi(u, e), \phi(u', e')\big\rangle$.

General methodology for analyzing expressivity

Observe that the gradient of $\nabla f\left(\theta^{(0)}; u, e\right)$ exactly plays the role of feature lift!

Definition (Neural tangent kernel)

$K_{NTK}\left(u, e; u', e'\right) := \left\langle \nabla f\left(\theta^{(0)}; u, e\right), \nabla f\left(\theta^{(0)}; u', e'\right) \right\rangle.$

Note that K_{NTK} depends only on the initialization and model structure. It is not affected by the GD optimization or d/n ratio.
Hence, during GD optimization, the response surface $F(\theta)$ converges to that of the **kernelized predictor**:

$$\arg\min_{f} \|f\|_{K_{NTK}} \; s.t. \; y_{u,e} f(u, e) \geq 1, \; \forall (u, e) \in \mathcal{D}_{\text{trn}}.$$

Here, we use the dual formulation of kernelized predictors.
It means we only need to analyze K_{NTK} to investigate and compare the expressvity of over-parameterized models :)

General methodology to study GD optimization

Neural tangent kernel provides a versatile tool to study the limiting expressivity (as $d \to \infty$).

However, what is the optimization path during a regular session of GD: $\theta^{(t)} = \theta^{(t-1)} - \eta f(\theta^{(t-1)})$?

We mentioned earlier that GD is model agnostic, but GD also carries an **implicit (data-dependent) bias**! Suppose the loss function has exponential-tail behavior. Then the gradient is:

$$\frac{1}{n} \sum_{(u,e) \in \mathcal{D}_{\textbf{trn}}} -y_{u,e} \exp\Big(- y_{u,e} f(\theta; u, e)\Big) \nabla f(\theta; u, e).$$

As gradient descent proceeds, ∇f will decrease. If $\|f(\theta; \cdot)\|_2$ increases at the same time, the gradient is then dominated by the term:

$$\min_{(u,e) \in \mathcal{D}_{\textbf{trn}}} y_{u,e} f(\theta; u, e),$$

which is exactly the **worst margin**!

General methodology to study GD optimization

Therefore, GD will focus on optimizing the worst margin, making the optimization path behave that of the **hard-margin SVM** (if we assume $\mathcal{D}_{trn}$ can be separated by the over-parameterized $f(\theta;\cdot)$):

$$\max_{\|\theta\|_2\leq 1}\ \min_{(u,e)\in\mathcal{D}_{trn}} y_{u,e} f(\theta; u, e).$$

Formally, three mild conditions will guarantee the convergence of the optimization path:

- the loss function has *exponential-tail behavior*;
- $f(\theta;\cdot)$ behaves like a *homogeneous function*: $f(\theta;\cdot) = \|\theta\|_2 f\left(\theta/\|\theta\|_2;\cdot\right)$;
- the model is over-parameterized so it is capable of separating the data.

What is left is simply to analyze the **KKT point** (since the problem may not be convex) of the hard-margin SVM problem according to the properties of f.

General methodology to study generalization

In IR, there are three types of generalizations:

- **transductive**: training and testing data are sampled *without replacement*, think about the movie rating completion problem – the same (u, e) pair will not appear twice;
- **inductive**: training and testing data are i.i.d samples from the same distribution, think about the item-item recommendation problem – the same (e, e) pair may appear many times;
- **cross domain**: training and testing data are i.i.d samples from different distributions.

The two key notions for investigating generalization are: **structural complexity** and **domain discrepancy**.

General methodology to study generalization

Heuristically, structural complexity measures how well the predictor can fit random signals.
Let $\sigma_{u,e}$ be Rademacher (or Gaussian) random variables. Rademacher random variables take the values of $\{-1, +1\}$ with equal probability. If the task is transductive, then $\sigma_{u,e}$ is also obtained by sampling without replacement.

Definition (Empirical Rademacher complexity)

$$R_n(\mathcal{F}) = \sup_f \left| \frac{1}{n} \sum_{(u,e)\in\mathcal{D}} \sigma_{u,e} f(u,e) \right|$$

- There are many *contraction* and *norm-bound* properties that allow us to derive the upper bound of $R_n(\mathcal{F})$ using the structures of $\mathcal{F}$;
- for instance, if $\mathcal{F}$ consists of factorization models, then we can consult results from *random-matrix theory*.
- if $\mathcal{F}$ consists of recursively-defined model, e.g. $\sigma(W^{(q)}\sigma(\ldots W^{(1)}X))$, then we can use the *peeling technique* that greatly simplifies the problem.

General methodology to study generalization

Domain discrepancy is often characterized by either:

- **integral probability metric (IPM)**: $D_{\mathcal{F}}(P\|Q) = \sup_{f\in\mathcal{F}} \left| \int f dP - \int f dQ \right|$ (Wasserstein distance, Maximum Mean Discrepancy),
- **f-divergence**: $D_f(P\|Q) = \int f\left(\frac{dP}{dQ}\right) dQ$ (KL-divergence)

While f-divergence directly uses likelihood ratio, it requires $supp(Q) \subset supp(P)$ that may not be satisfied in IR. IPM may better leverage the geometry of the data (we will revisit IPM later).
The *generalization error bound* usually has the formulation of:

$$\mathbb{E}_{P_{\textbf{test}}} \ell(f) \leq \hat{\mathbb{E}}_{P_{\textbf{trn}}} \ell(f) + \text{func}\big(D(P_{\text{test}}\|P_{\text{trn}}), R_n(\mathcal{F})\big) + \text{slack},$$

where $\hat{\mathbb{E}}$ represents the empirical average, $D(\cdot\|\cdot)$ is some discrepency term depending on the problem, and $\text{func}(\cdot)$ is a simple function.

Domain topic: MCF v.s. NCF

Ready to rock and roll? Let's see how our pyramid of model design and analysis lead to the cutting edge research.

To resolve the ongoing debate of *matrix collaborative filtering* (MCF) and *neural collaborative filtering* (NCF), we start from the NTK analysis. Xu et al.[2] shows that:

$$\lim_{t\to\infty}\lim_{d\to\infty} F\left(\frac{\theta}{\|\theta\|_2}\right) \overset{\text{stationary point}}{\rightarrow} \left\{\arg\min_f \|f\|_{K_{CF}} \ s.t. \ y_{u,e}f(u,e)\geq 1\right\},$$

with $K_{CF}(u, e; u', e') = a1[u = u'] + b1[e, e'] + c$. How does K_{CF} reflects the CF principle?

Here, the constants a, b, c are determined by the *model structure* and *initialization*. They essentially decide the *relative weights* and *global intercept* of CF.

[2] Revisiting NCF v.s. MCF: the theoretical perspectives. ICML'21

Domain topic: MCF v.s. NCF

Under GD optimization, Xu et al. also shows that:

$$\lim_{t\to\infty} \frac{\theta^{(t)}}{\|\theta\|_2} \overset{\text{stationary point}}{\longrightarrow} \begin{cases} \min \|\theta\|_2 \ s.t.\ y_{u,e}f(u,e) \geq 1, \text{ (for NCF)} \\ \min \|\theta\|_{\text{nuc}} \ s.t.\ y_{u,e}f(u,e) \geq 1, \text{ (for MCF)} \end{cases}$$

The key take away is that the solutions have different geometries induced by the norm constraints. We denote the layer-wise ℓ_2 norm of the resulting q-layer NCF as $\{\lambda_i\}_{i=1}^{q}$, and the nuclear norm of the result MCF as λ_{nuc}.
The difference in solution geometry leads to the different generalization behaviors. For **transductive generalization**:

$$\mathbb{E}_{P_{\textbf{test}}}\ell(f) \leq \hat{\mathbb{E}}_{P_{\textbf{trn}}}\ell(f) + \text{slack} + \begin{cases} \mathcal{O}\left(\sqrt{q}\prod_{i=1}^{q}\lambda_i/\sqrt{n}\right) \text{ (for NCF)} \\ \mathcal{O}\left(\sqrt{\log|\mathcal{U}|}\sqrt{|\mathcal{I}|}\cdot\lambda_{\text{nuc}}/n\right) \text{ (for MCF)} \end{cases}$$

MCF has better rate at transductive tasks! Intuitive explaination? We save inductive task for later.

Domain topic: importance weighting for IR

Reweighting is critical for many IR applications: correcting bias, fairness, etc. But handling missing data? Not so quick...

In most cases, people consider $w(u,e) = \frac{P_{test}(u,e)}{P_{trn}(u,e)}$ to correct for the domain shift. Our analysis will help answer three questions:

1. how does it affect optimization?
2. how does it affect generalization?
3. why it may not handle missing data?

It is shown in Xu et al.[3] that If $\mathcal{D}$ is separable by $f(\theta;\cdot)$, then reweighting only affects the convergence speed to θ^* – which is the KKT point of the hard-margin SVM as discussed before:

$$\left| \frac{\theta^{(t)}(w)}{\|\theta^{(t)}(w)\|_2} - \theta^* \right| \lesssim \frac{\log n + D_{KL}(p^* \| w)}{\log t},$$

where p^* is the dual optimum of the hard-margin SVM.

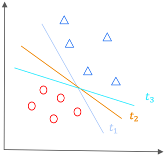
[3] Rethinking the role of importance weighting for deep learning. ICLR'21

Domain topic: importance weighting for IR

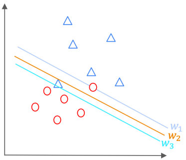
If $\mathcal{D}$ is not separable, we can decompose it into $\mathcal{D}_{\text{sep}} \bigcup \mathcal{D}_{\text{non-sep}}$. Let Π be the orthogonal projection, then: $\theta^{(t)}(w) = \Pi_{\text{sep}}\theta^{(t)}(w) + \Pi_{\text{non-sep}}\theta^{(t)}(w)$.
The story is the same on the separable part.
On the non-separable part, we have $\tilde{\theta}(w) = \arg\min R_{\text{non-sep}}(\theta; w)$, which is uniquely defined by the weights, and that:

$$\left|\Pi_{\text{non-sep}}\theta^{(t)}(w) - \tilde{\theta}(w)\right| \lesssim \frac{\mathcal{O}(1) + \log^2 t}{t}.$$

Therefore, reweighting only shifts the decision boundary on the non-separable subspace, e.g. by inducing different intercepts.



Separable case: weights only affect the convergence speed to the only optimal solution.



Non-separable case: weights affect the intercept (i.e. shift) on the space defined by the non-separable subset.

Domain topic: importance weighting for IR

For generalizing to a different but overlapped domain, we apply reweighting to the previous NCF vs. MCF example. The cross-domain generalization bounds are:

$$\mathbb{E}_Q \ell(f) \leq \hat{\mathbb{E}}_{P_w} \ell(f) + \text{slack} + \begin{cases} \mathcal{O}\Big(D_1(P\|Q) \cdot \sqrt{q} \prod_{i=1}^{q} \lambda_i / \sqrt{n}\Big) \text{ (NCF)} \\ \mathcal{O}\Big(D_2(P\|Q) \cdot \sqrt{(|\mathcal{U}| + |\mathcal{I}|) \log n \lambda_{\text{nuc}} / n}\Big) \text{ (MCF)} \end{cases}$$

For inductive tasks, NCF has a better rate, and the domain difference plays a multiplicative factor on the model complexity terms! Intuitive explanations?

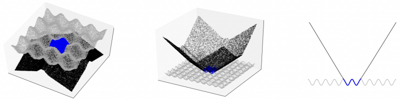
In general, reweighting can achieve domain generalization at the cost of sample efficiency. But we have assumed support overlap, which is not practical. What if we need to generalize to unseen domain?

Domain topic: generalizing to unseen domain

Again, according to the preivous NTK analysis, over-parameterized NN behave like **linear predictors** outside training domain. In particular, Xu et al.[4] shows for two-layer MLP that outside the training domain, for any direction v, there exists a linear model coeficient β_v such that:

$$\lim_{t\to\infty}\left|\frac{f\big(\theta^{(t)};X_{u,e}+\delta v\big)-f\big(\theta^{(t)};X_{u,e}\big)}{\delta}-\beta_v\right|<\mathcal{O}(\frac{1}{t}).$$

Hence, for reparameterized models, reweighting at most changes the *slope* outside the training domain, but not the nature of the linear extrapolation!



[4] How Neural Networks Extrapolate, ICLR'21

Domain topic: generalizing to unseen domain

For IR pattern recognition, the linear extrapolation means the hardness of learning under *insufficient* support overlap.
In particular, Xu et al.[5] show from a **PAC-Bayes learning** perspective that even with reweighting, the generalization errors express as:

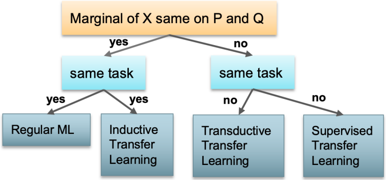
$$\text{test err} \leq \text{trn err} + \text{slack} + \text{complexity} + D(P\|Q) + \mathbb{E}_{Q/P, f\sim\tau} R(f),$$

where $D(P\|Q)$ is the discrepancy on the overlapped region, $\mathbb{E}_{Q/P, f\sim\tau}$ is taken on the non-overlapped region and τ is *any* post-training distribution on the model space.
The result suggests we can at most expect an average out-of-domain performance from any modelling and reweighting efforts. A recent line of research tries to use representation learning to achieve domain generalization, which we will discuss next.

[5] From Intervention to Domain Transportation: a New Perspective for Recommendation, ICLR'22

Domain generalization

Possible sources: covariate shift, target shift, concept change, selection bias, etc.



Seminal results for generalizing across domain:

$$\text{test err} \leq \text{trn err} + \text{slack} + disc(P, Q) + \inf_{f \in \mathcal{F}} \{R_P(f) + R_Q(f)\}.$$

Earlier work uses the $H\Delta H$ terminology to characterize the discrepancy term $disc(P, Q)$. Recent work extends it to using IPM such as the Wasserstein's distance $D_W(P\|Q)$.
However, the optimal joint risk term $\inf_{f \in \mathcal{F}} \{R_P(f) + R_Q(f)\}$ eventually decides if cross-domain generalization is truly feasible.

Domain generalization

The most common practice of using representation learning to achieve domain generalization is to find an "invariant" representation mapping ϕ such that the domain discrepancy on the representation-induced domains $P(\phi)$ and $Q(\phi)$ are small.
The following objective is commonly used:

$$\min_{f,\phi} \mathbb{E}_P \ell(f \circ \phi) + D_W(P(\phi) \| Q(\phi)),$$

where the hope is that ϕ achieves both good training performance while minimizing domain discrepancy.
However, it is often ignored that the presumption for this to work is that the optimal joint risk $\inf_{f \in \mathcal{F}} \left\{ R_P(f) + R_Q(f) \right\}$ can be made small. This is not always the case especially for IR. In particular, Xu et al.[6] shows for **item2vec** (which we discussed earlier) that:

$$\inf_{\phi} \left[\mathbb{E}_P \ell(\phi) + \mathbb{E}_Q \ell(\phi) \right] \gtrsim \sum_{(u,e) \in \mathcal{D}} D_{KL}(Y_{u,e}(P) \| Y_{u,e}(Q)),$$

[6] From Intervention to Domain Transportation: a New Perspective for Recommendation, ICLR'22

Domain generalization

- The previous slide suggests that if P and Q are not well-aligned in terms of the co-occurrence statistics $Y_{u,e}$, then domain generalization is hard!
- Does the intuition from item2vec hold for general representation learning in IR? Comparing IR with CV or NLP, there often lacks a rich enough feature space from which meaningful joint representations can be constructed.
- If perfect cross-domain generalization in IR cannot be achieved anyway, is it possible to find some instruments that help us improve the performance?