Towards a Foundation of Multitask Learning

Hongyang R. Zhang

Northeastern University, Boston

Ph.D. in CS, Stanford University

Overarching goal: Towards a more practical foundation of neural networks

Research interest over the last couple of years: Understand the workings of neural networks

A central topic in recent literature is generalization: **Modern neural networks don't overfit as badly as one would have thought** [ZBH+21]

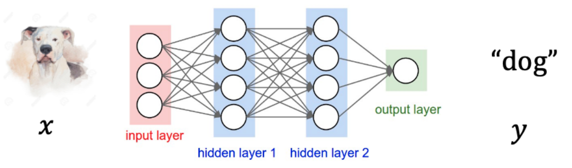


Figure 1: A three-layer neural network trying to recognize a dog image

- $\mathcal{D}$: A data distribution supported on feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- f_W: A neural net with weight variable W (edge weights of Figure 1)
- $\ell(f_W(x), y)$: a loss function that measures the performance of f_W
- **SGD easily minimizes empirical risk even when** y **is random** [ZBH+21]

Overarching goal: Towards a more practical foundation of neural networks

Research interest over the last couple of years: Understand the workings of neural networks

A central topic in recent literature is generalization: **Modern neural networks don't overfit as badly as one would have thought** [ZBH+21]

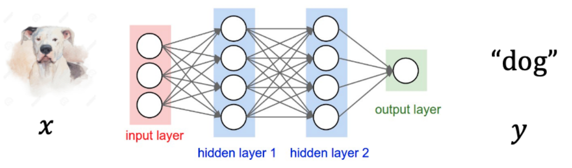


Figure 1: A three-layer neural network trying to recognize a dog image

- $\mathcal{D}$: A data distribution supported on feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- f_W: A neural net with weight variable W (edge weights of Figure 1)
- $\ell(f_W(x), y)$: a loss function that measures the performance of f_W
- **SGD easily minimizes empirical risk even when y is random** [ZBH+21]

Overarching goal: Towards a more practical foundation of neural networks

Research interest over the last couple of years: Understand the workings of neural networks

A central topic in recent literature is generalization: **Modern neural networks don't overfit as badly as one would have thought** [ZBH+21]

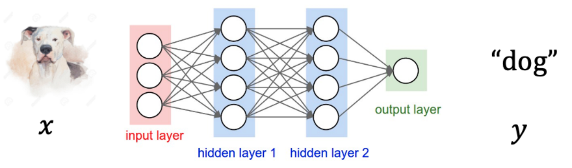


Figure 1: A three-layer neural network trying to recognize a dog image

- $\mathcal{D}$: A data distribution supported on feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- f_W: A neural net with weight variable W (edge weights of Figure 1)
- $\ell(f_W(x), y)$: a loss function that measures the performance of f_W
- **SGD easily minimizes empirical risk even when y is random** [ZBH+21]

Overarching goal: Towards a more practical foundation of neural networks

Research interest over the last couple of years: Understand the workings of neural networks

A central topic in recent literature is generalization: **Modern neural networks don't overfit as badly as one would have thought** [ZBH+21]

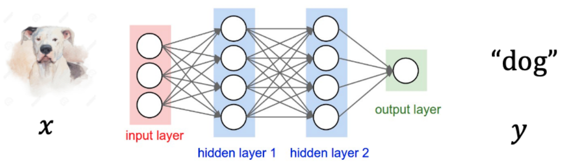


Figure 1: A three-layer neural network trying to recognize a dog image

- $\mathcal{D}$: A data distribution supported on feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- f_W: A neural net with weight variable W (edge weights of Figure 1)
- $\ell(f_W(x), y)$: a loss function that measures the performance of f_W
- **SGD easily minimizes empirical risk even when y is random** [ZBH+21]

Overarching goal: Towards a more practical foundation of neural networks

Research interest over the last couple of years: Understand the workings of neural networks

A central topic in recent literature is generalization: **Modern neural networks don't overfit as badly as one would have thought** [ZBH+21]

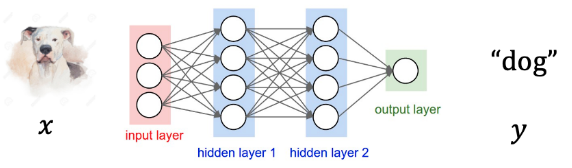


Figure 1: A three-layer neural network trying to recognize a dog image

- $\mathcal{D}$: A data distribution supported on feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- f_W: A neural net with weight variable W (edge weights of Figure 1)
- $\ell(f_W(x), y)$: a loss function that measures the performance of f_W
- **SGD easily minimizes empirical risk even when y is random** [ZBH+21]

Overarching goal: Towards a more practical foundation of neural networks

Research interest over the last couple of years: Understand the workings of neural networks

A central topic in recent literature is generalization: **Modern neural networks don't overfit as badly as one would have thought** [ZBH+21]

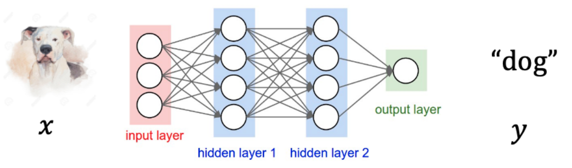


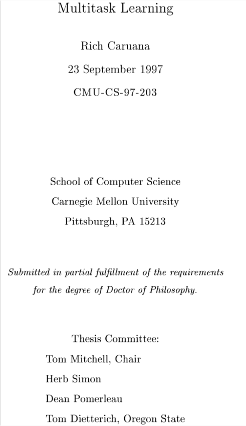
Figure 1: A three-layer neural network trying to recognize a dog image

- $\mathcal{D}$: A data distribution supported on feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- f_W: A neural net with weight variable W (edge weights of Figure 1)
- $\ell(f_W(x), y)$: a loss function that measures the performance of f_W
- **SGD easily minimizes empirical risk even when y is random** [ZBH+21]

This talk: Multitask learning

Given k tasks (or samples from data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$), can we build neural networks to predict all the tasks simultaneously?

Multitask learning is studied since early days of ML w. many applications


Multitask Learning

Rich Caruana

23 September 1997

CMU-CS-97-203

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

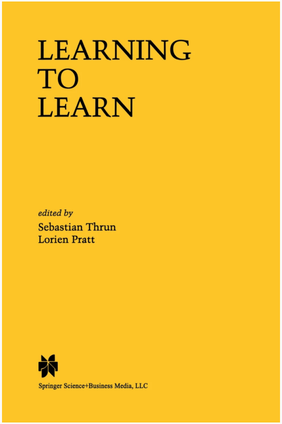
Thesis Committee:

Tom Mitchell, Chair

Herb Simon

Dean Pomerleau

Tom Dietterich, Oregon State

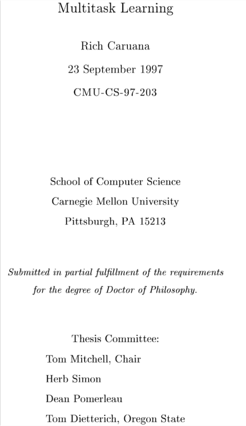


Particularly relevant today thanks to foundation models, federated learning, LLMs, . . .

This talk: Multitask learning

Given k tasks (or samples from data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$), can we build neural networks to predict all the tasks simultaneously?

Multitask learning is studied since early days of ML w. many applications



Multitask Learning

Rich Caruana

23 September 1997

CMU-CS-97-203

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

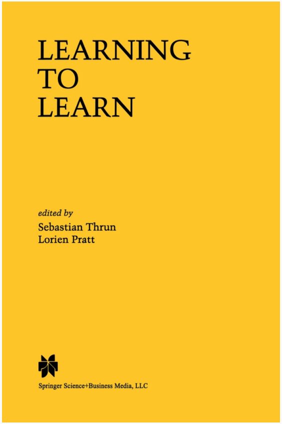
Thesis Committee:

Tom Mitchell, Chair

Herb Simon

Dean Pomerleau

Tom Dietterich, Oregon State

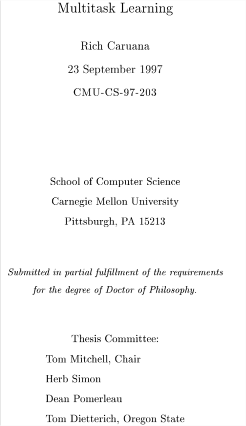


Particularly relevant today thanks to foundation models, federated learning, LLMs, . . .

This talk: Multitask learning

Given k tasks (or samples from data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$), can we build neural networks to predict all the tasks simultaneously?

Multitask learning is studied since early days of ML w. many applications


Multitask Learning

Rich Caruana

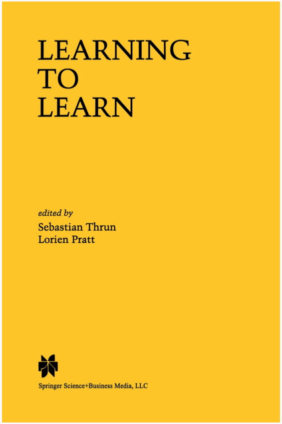
23 September 1997

CMU-CS-97-203

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Tom Mitchell, Chair
Herb Simon
Dean Pomerleau
Tom Dietterich, Oregon State



Particularly relevant today thanks to foundation models, federated learning, LLMs, ...

Motivation: Negative transfer

If different features are required for making predictions from data s to data t (Figure 2), "negative" transfers—meaning that combining datasets s, t performs worse than learning with target data t alone—are likely to occur

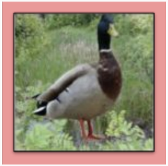


Figure 2: A red bird on water vs. A waterbird on land

Fundamental questions

1. Can we identify when negative transfers would happen?
2. Can we design algorithms to account for negative transfers?

Motivation: Negative transfer

If different features are required for making predictions from data s to data t (Figure 2), "negative" transfers—meaning that combining datasets s, t performs worse than learning with target data t alone—are likely to occur

Fundamental questions

1. Can we identify when negative transfers would happen?
2. Can we design algorithms to account for negative transfers?

Motivation: Negative transfer

If different features are required for making predictions from data s to data t (Figure 2), "negative" transfers—meaning that combining datasets s, t performs worse than learning with target data t alone—are likely to occur

Fundamental questions

1. Can we identify when negative transfers would happen?
2. Can we design algorithms to account for negative transfers?

This talk

We will analyze transfer in neural networks from the lens of deep learning theory. Then we design boosting algorithms to maximize positive transfers

High-level overview

1. **Provide an analysis of transfer in multi-headed neural networks**
   - Observe a connection to two-layer neural networks [WZR20]
   - Case study of linear regression under distribution shifts [YZW+20]
   - Identification using linear surrogate models [LNZ23]
2. **Design boosting algorithms for multitask learning**
   - Introduce a notion called higher-order task affinity [LJS+23]
   - Design gradient-based estimation of task affinity [Working Paper]

This talk

We will analyze transfer in neural networks from the lens of deep learning theory. Then we design boosting algorithms to maximize positive transfers

High-level overview

1. **Provide an analysis of transfer in multi-headed neural networks**
    - Observe a connection to two-layer neural networks [WZR20]
    - Case study of linear regression under distribution shifts [YZW+20]
    - Identification using linear surrogate models [LNZ23]
2. **Design boosting algorithms for multitask learning**
    - Introduce a notion called higher-order task affinity [LJS+23]
    - Design gradient-based estimation of task affinity [Working Paper]

We will analyze transfer in neural networks from the lens of deep learning theory. Then we design boosting algorithms to maximize positive transfers

High-level overview

1. **Provide an analysis of transfer in multi-headed neural networks**
    - Observe a connection to two-layer neural networks [WZR20]
    - Case study of linear regression under distribution shifts [YZW+20]
    - Identification using linear surrogate models [LNZ23]
2. **Design boosting algorithms for multitask learning**
    - Introduce a notion called higher-order task affinity [LJS+23]
    - Design gradient-based estimation of task affinity [Working Paper]

This talk

We will analyze transfer in neural networks from the lens of deep learning theory. Then we design boosting algorithms to maximize positive transfers

High-level overview

1. **Provide an analysis of transfer in multi-headed neural networks**
    - Observe a connection to two-layer neural networks [WZR20]
    - Case study of linear regression under distribution shifts [YZW+20]
    - Identification using linear surrogate models [LNZ23]
2. **Design boosting algorithms for multitask learning**
    - Introduce a notion called higher-order task affinity [LJS+23]
    - Design gradient-based estimation of task affinity [Working Paper]

This talk

We will analyze transfer in neural networks from the lens of deep learning theory. Then we design boosting algorithms to maximize positive transfers

High-level overview

1. **Provide an analysis of transfer in multi-headed neural networks**
    - Observe a connection to two-layer neural networks [WZR20]
    - Case study of linear regression under distribution shifts [YZW+20]
    - Identification using linear surrogate models [LNZ23]
2. **Design boosting algorithms for multitask learning**
    - Introduce a notion called higher-order task affinity [LJS+23]
    - Design gradient-based estimation of task affinity [Working Paper]

This talk

We will analyze transfer in neural networks from the lens of deep learning theory. Then we design boosting algorithms to maximize positive transfers

High-level overview

1. **Provide an analysis of transfer in multi-headed neural networks**
    - Observe a connection to two-layer neural networks [WZR20]
    - Case study of linear regression under distribution shifts [YZW+20]
    - Identification using linear surrogate models [LNZ23]
2. **Design boosting algorithms for multitask learning**
    - Introduce a notion called higher-order task affinity [LJS+23]
    - Design gradient-based estimation of task affinity [Working Paper]

Related literature: Generalization in neural networks

- Norm/margin generalization bounds: given a hypothesis space of neural networks $\mathcal{H}$, bound the (worst-case) generalization gap over this hypothesis space $\mathcal{H}$
  - **Rademacher complexity bounds** [BFT17] (uniform convergence)
  - **PAC-Bayesian bounds** [AGN+18] (data dependent, e.g., Hessian-based [JLZ22])
- GD/SGD in nonconvex optimization
  - **Neural tangent kernels** [ADH+19]: analysis of over-parameterization
  - **Implicit regularization** [LMZ18]: in certain cases of using SGD, no need for an explicit regularizer
- High-dimensional statistical analysis
  - **Precise asymptotics** via random matrix theory [SC19]
  - **Benign overfitting** [BLL+20]

Related literature: Generalization in neural networks

- Norm/margin generalization bounds: given a hypothesis space of neural networks $\mathcal{H}$, bound the (worst-case) generalization gap over this hypothesis space $\mathcal{H}$
  - **Rademacher complexity bounds** [BFT17] (uniform convergence)
  - **PAC-Bayesian bounds** [AGN+18] (data dependent, e.g., Hessian-based [JLZ22])
- GD/SGD in nonconvex optimization
  - **Neural tangent kernels** [ADH+19]: analysis of over-parameterization
  - **Implicit regularization** [LMZ18]: in certain cases of using SGD, no need for an explicit regularizer
- High-dimensional statistical analysis
  - **Precise asymptotics** via random matrix theory [SC19]
  - **Benign overfitting** [BLL+20]

Related literature: Generalization in neural networks

- Norm/margin generalization bounds: given a hypothesis space of neural networks $\mathcal{H}$, bound the (worst-case) generalization gap over this hypothesis space $\mathcal{H}$
  - **Rademacher complexity bounds** [BFT17] (uniform convergence)
  - **PAC-Bayesian bounds** [AGN+18] (data dependent, e.g., Hessian-based [JLZ22])
- GD/SGD in nonconvex optimization
  - **Neural tangent kernels** [ADH+19]: analysis of over-parameterization
  - **Implicit regularization** [LMZ18]: in certain cases of using SGD, no need for an explicit regularizer
- High-dimensional statistical analysis
  - **Precise asymptotics** via random matrix theory [SC19]
  - **Benign overfitting** [BLL+20]

Related literature: Generalization in neural networks

- Norm/margin generalization bounds: given a hypothesis space of neural networks $\mathcal{H}$, bound the (worst-case) generalization gap over this hypothesis space $\mathcal{H}$
  - **Rademacher complexity bounds** [BFT17] (uniform convergence)
  - **PAC-Bayesian bounds** [AGN+18] (data dependent, e.g., Hessian-based [JLZ22])
- GD/SGD in nonconvex optimization
  - **Neural tangent kernels** [ADH+19]: analysis of over-parameterization
  - **Implicit regularization** [LMZ18]: in certain cases of using SGD, no need for an explicit regularizer
- High-dimensional statistical analysis
  - **Precise asymptotics** via random matrix theory [SC19]
  - **Benign overfitting** [BLL+20]

Related literature: Generalization in neural networks

- Norm/margin generalization bounds: given a hypothesis space of neural networks $\mathcal{H}$, bound the (worst-case) generalization gap over this hypothesis space $\mathcal{H}$
  - **Rademacher complexity bounds** [BFT17] (uniform convergence)
  - **PAC-Bayesian bounds** [AGN+18] (data dependent, e.g., Hessian-based [JLZ22])
- GD/SGD in nonconvex optimization
  - **Neural tangent kernels** [ADH+19]: analysis of over-parameterization
  - **Implicit regularization** [LMZ18]: in certain cases of using SGD, no need for an explicit regularizer
- High-dimensional statistical analysis
  - **Precise asymptotics** via random matrix theory [SC19]
  - **Benign overfitting** [BLL+20]

Related literature: Generalization in neural networks

- Norm/margin generalization bounds: given a hypothesis space of neural networks $\mathcal{H}$, bound the (worst-case) generalization gap over this hypothesis space $\mathcal{H}$
  - **Rademacher complexity bounds** [BFT17] (uniform convergence)
  - **PAC-Bayesian bounds** [AGN+18] (data dependent, e.g., Hessian-based [JLZ22])
- GD/SGD in nonconvex optimization
  - **Neural tangent kernels** [ADH+19]: analysis of over-parameterization
  - **Implicit regularization** [LMZ18]: in certain cases of using SGD, no need for an explicit regularizer
- High-dimensional statistical analysis
  - **Precise asymptotics** via random matrix theory [SC19]
  - **Benign overfitting** [BLL+20]

Outline

Setup

Multitask learning is usually conducted using a multi-headed neural network [Car97]

- B: shared feature layer for all tasks
- A_i: output prediction layer for task i, for $i = 1, 2, \ldots, k$

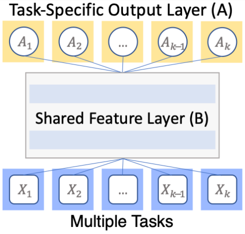


Figure 2: A multi-headed neural network for training k tasks

Observation: When B is a single layer and the activation is a linear map, this architecture is a two-layer neural network, whose generalization properties have been extensively studied

Setup

Multitask learning is usually conducted using a multi-headed neural network [Car97]

- B: shared feature layer for all tasks
- A_i: output prediction layer for task i, for $i = 1, 2, \ldots, k$

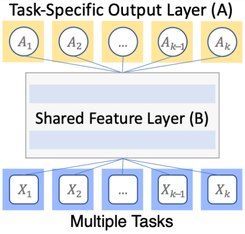


Figure 2: A multi-headed neural network for training k tasks

Observation: When B is a single layer and the activation is a linear map, this architecture is a two-layer neural network, whose generalization properties have been extensively studied

Implication: How model capacity affects transfer

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \,\mathrm{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \,\mathrm{Id})$, $i = 1, \dots, n_t$

Question: How does combining tasks s, t with the MTL network compare with learning with target task t alone?

Proposition [WZR20]

If the width of layer B is at least 2, then combining s, t has the same performance as learning t alone (contrary to beliefs in DL that more parameters are better)

- Proof is via regression analysis when activation function σ is linear: $\sigma(x) = x$; also generalize to nonlinear activation (e.g., ReLU)
- Capacity too large $\rightarrow$ no interference
- Next: Capacity too small $\rightarrow$ destructive interference

Implication: How model capacity affects transfer

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_t$

Question: How does combining tasks s, t with the MTL network compare with learning with target task t alone?

Proposition [WZR20]

If the width of layer B is at least 2, then combining s, t has the same performance as learning t alone (contrary to beliefs in DL that more parameters are better)

- Proof is via regression analysis when activation function σ is linear: $\sigma(x) = x$; also generalize to nonlinear activation (e.g., ReLU)
- Capacity too large → no interference
- Next: Capacity too small → destructive interference

Implication: How model capacity affects transfer

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_t$

Question: How does combining tasks s, t with the MTL network compare with learning with target task t alone?

Proposition [WZR20]

If the width of layer B is at least 2, then combining s, t has the same performance as learning t alone (contrary to beliefs in DL that more parameters are better)

- Proof is via regression analysis when activation function σ is linear: $\sigma(x) = x$; also generalize to nonlinear activation (e.g., ReLU)
- Capacity too large $\rightarrow$ no interference
- Next: Capacity too small $\rightarrow$ destructive interference

Implication: How model capacity affects transfer

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \, \mathrm{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \, \mathrm{Id})$, $i = 1, \dots, n_t$

Question: How does combining tasks s, t with the MTL network compare with learning with target task t alone?

Proposition [WZR20]

If the width of layer B is at least 2, then combining s, t has the same performance as learning t alone (contrary to beliefs in DL that more parameters are better)

- Proof is via regression analysis when activation function σ is linear: $\sigma(x) = x$; also generalize to nonlinear activation (e.g., ReLU)
- Capacity too large $\rightarrow$ no interference
- Next: Capacity too small $\rightarrow$ destructive interference

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \,\mathrm{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \,\mathrm{Id})$, $i = 1, \dots, n_t$

Types of data set shifts

- Covariate shift: $\Sigma_s \neq \Sigma_t$
- Model shift: $\beta^{(s)} \neq \beta^{(t)}$
- Data size imbalance: $n_s \neq n_t$

Theorem [YZW+20]

We characterize the transfer effect of data set pooling for various combinations of these shifts

Data set shifts

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \, \mathrm{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \, \mathrm{Id})$, $i = 1, \dots, n_t$

Types of data set shifts

- Covariate shift: $\Sigma_s \neq \Sigma_t$
- Model shift: $\beta^{(s)} \neq \beta^{(t)}$
- Data size imbalance: $n_s \neq n_t$

Theorem [YZW+20]

We characterize the transfer effect of data set pooling for various combinations of these shifts

Data set shifts

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_t$

Types of data set shifts

- Covariate shift: $\Sigma_s \neq \Sigma_t$
- Model shift: $\beta^{(s)} \neq \beta^{(t)}$
- Data size imbalance: $n_s \neq n_t$

Theorem [YZW+20]

We characterize the transfer effect of data set pooling for various combinations of these shifts

Data set shifts

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \operatorname{Id})$, $i = 1, \dots, n_t$

Types of data set shifts

- Covariate shift: $\Sigma_s \neq \Sigma_t$
- Model shift: $\beta^{(s)} \neq \beta^{(t)}$
- Data size imbalance: $n_s \neq n_t$

Theorem [YZW+20]

We characterize the transfer effect of data set pooling for various combinations of these shifts

Data set shifts

Case study of two linear regression tasks

- Task s: $y_i = x_i^\top \beta^{(s)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_s)$, $\epsilon_i \sim N(0, \sigma^2 \,\mathrm{Id})$, $i = 1, \ldots, n_s$
- Task t: $y_i = x_i^\top \beta^{(t)} + \epsilon_i$, for $x_i \sim N(0, \Sigma_t)$, $\epsilon_i \sim N(0, \sigma^2 \,\mathrm{Id})$, $i = 1, \ldots, n_t$

Types of data set shifts

- Covariate shift: $\Sigma_s \neq \Sigma_t$
- Model shift: $\beta^{(s)} \neq \beta^{(t)}$
- Data size imbalance: $n_s \neq n_t$

Theorem [YZW+20]

We characterize the transfer effect of data set pooling for various combinations of these shifts

Covariate shift can either help or hurt, depending on whether n_s is greater than n_t or not

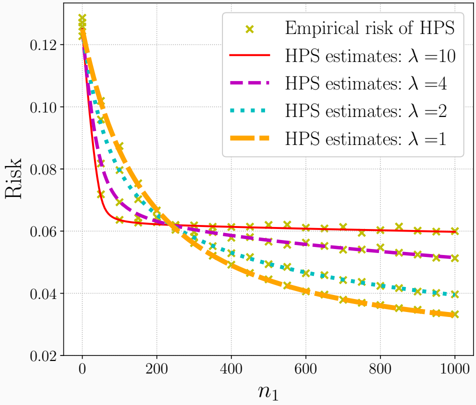


Figure 3: When $n_s \leq n_t$, having covariate shift helps; When $n_s > n_t$, having it hurts. λ refers to the degree of covariate shift (higher is more severe, n_s is n_1 in the figure)

Vignette II: Model shift vs. data set sizes

Depending on n_t and the extent of model shift, the transfer effect can always be positive (irrespective of n_s), be positive for a restricted range of n_s, always be negative

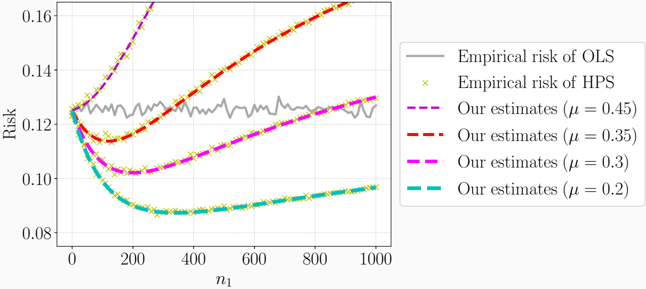


Figure 4: The gray line refers to the empirical excess risk of Ordinary Least Squares on t. Any point above the gray link represents negative transfer, while any point below represents positive transfer. μ controls the model shift so that $\left\|\beta^{(s)} - \beta^{(t)}\right\|^2 = 2\mu^2$.

Beyond linear regression

What can we say beyond the case study?

- "Discrepancy" notions from the *learning theory literature* measure the distance between two tasks
- $\mathcal{H}$-divergence [BBC+10] between two distributions D and D' on domain $\mathcal{X}$

$$d_{\mathcal{H}}(D, D') = 2 \sup_{h \in \mathcal{H}} \left| \Pr_{D}[I(h)] - \Pr_{D'}[I(h')] \right|$$

  where $\mathcal{H}$ is a hypothesis class on domain X, and $I(h)$ is the characteristic function satisfying $x \in I(h) \Leftrightarrow h(x) = 1$
- Challenge: Difficult to implement $\mathcal{H}$-divergence within neural networks

Beyond linear regression

What can we say beyond the case study?

- "Discrepancy" notions from the *learning theory literature* measure the distance between two tasks
- $\mathcal{H}$-divergence [BBC+10] between two distributions D and D' on domain $\mathcal{X}$

$$d_{\mathcal{H}}(D, D') = 2 \sup_{h \in \mathcal{H}} \left| \Pr_{D}[I(h)] - \Pr_{D'}[I(h')] \right|$$

  where $\mathcal{H}$ is a hypothesis class on domain X, and $I(h)$ is the characteristic function satisfying $x \in I(h) \Leftrightarrow h(x) = 1$

- Challenge: Difficult to implement $\mathcal{H}$-divergence within neural networks

Beyond linear regression

What can we say beyond the case study?

- "Discrepancy" notions from the *learning theory literature* measure the distance between two tasks
- $\mathcal{H}$-divergence [BBC+10] between two distributions D and D' on domain $\mathcal{X}$

$$d_{\mathcal{H}}(D, D') = 2 \sup_{h \in \mathcal{H}} \left| \Pr_{D}[I(h)] - \Pr_{D'}[I(h')] \right|$$

  where $\mathcal{H}$ is a hypothesis class on domain X, and $I(h)$ is the characteristic function satisfying $x \in I(h) \Leftrightarrow h(x) = 1$

- Challenge: Difficult to implement $\mathcal{H}$-divergence within neural networks

Our approach

Define a value function to quantify the "discrepancy" between source and target tasks

- Accommodates any number of source tasks
- Scales to a large number of source tasks

Definition [LNZ23]

- There are k source tasks: $\{1, 2, \cdots, k\}$, and a target task t
- For any $S \subseteq \{1, 2, \cdots, k\}$, a neural net is trained by combining S and t
- The **out-of-sample error** of this NN on the target task is defined as the **value function** of S, denoted as $f(S) \in \mathbb{R}$, for every S

Note: If we could evaluate all possible $f(S)$, then we can find the best subset of source tasks that minimizes $f(S)$

Computationally expensive due to 2^k subsets

Question: Find the subset that minimizes the value function without exhaustive search?

Our approach

Define a value function to quantify the "discrepancy" between source and target tasks

- Accommodates any number of source tasks
- Scales to a large number of source tasks

Definition [LNZ23]

- There are k source tasks: $\{1, 2, \cdots, k\}$, and a target task t
- For any $S \subseteq \{1, 2, \cdots, k\}$, a neural net is trained by combining S and t
- The **out-of-sample error** of this NN on the target task is defined as the **value function** of S, denoted as $f(S) \in \mathbb{R}$, for every S

Note: If we could evaluate all possible $f(S)$, then we can find the best subset of source tasks that minimizes $f(S)$

Computationally expensive due to 2^k subsets

Question: Find the subset that minimizes the value function without exhaustive search?

Our approach

Define a value function to quantify the "discrepancy" between source and target tasks

- Accommodates any number of source tasks
- Scales to a large number of source tasks

Definition [LNZ23]

- There are k source tasks: $\{1, 2, \cdots, k\}$, and a target task t
- For any $S \subseteq \{1, 2, \cdots, k\}$, a neural net is trained by combining S and t
- The **out-of-sample error** of this NN on the target task is defined as the **value function** of S, denoted as $f(S) \in \mathbb{R}$, for every S

Note: If we could evaluate all possible $f(S)$, then we can find the best subset of source tasks that minimizes $f(S)$

Computationally expensive due to 2^k subsets

Question: Find the subset that minimizes the value function without exhaustive search?

Our approach

Define a value function to quantify the "discrepancy" between source and target tasks

- Accommodates any number of source tasks
- Scales to a large number of source tasks

Definition [LNZ23]

- There are k source tasks: $\{1, 2, \cdots, k\}$, and a target task t
- For any $S \subseteq \{1, 2, \cdots, k\}$, a neural net is trained by combining S and t
- The **out-of-sample error** of this NN on the target task is defined as the **value function** of S, denoted as $f(S) \in \mathbb{R}$, for every S

Note: If we could evaluate all possible $f(S)$, then we can find the best subset of source tasks that minimizes $f(S)$

Computationally expensive due to 2^k subsets

Question: Find the subset that minimizes the value function without exhaustive search?

Our approach

Define a value function to quantify the "discrepancy" between source and target tasks

- Accommodates any number of source tasks
- Scales to a large number of source tasks

Definition [LNZ23]

- There are k source tasks: $\{1, 2, \cdots, k\}$, and a target task t
- For any $S \subseteq \{1, 2, \cdots, k\}$, a neural net is trained by combining S and t
- The **out-of-sample error** of this NN on the target task is defined as the **value function** of S, denoted as $f(S) \in \mathbb{R}$, for every S

Note: If we could evaluate all possible $f(S)$, then we can find the best subset of source tasks that minimizes $f(S)$

Computationally expensive due to 2^k subsets

Question: Find the subset that minimizes the value function without exhaustive search?

Our approach

Define a value function to quantify the "discrepancy" between source and target tasks

- Accommodates any number of source tasks
- Scales to a large number of source tasks

Definition [LNZ23]

- There are k source tasks: $\{1, 2, \cdots, k\}$, and a target task t
- For any $S \subseteq \{1, 2, \cdots, k\}$, a neural net is trained by combining S and t
- The **out-of-sample error** of this NN on the target task is defined as the **value function** of S, denoted as $f(S) \in \mathbb{R}$, for every S

Note: If we could evaluate all possible $f(S)$, then we can find the best subset of source tasks that minimizes $f(S)$

Computationally expensive due to 2^k subsets

Question: Find the subset that minimizes the value function without exhaustive search?

Surrogate models

Estimate a surrogate model to approximate the value function

- $g_{\boldsymbol{\theta}}(S)$: A surrogate model parameterized by $\boldsymbol{\theta}$
- Linear surrogate model: $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_k) \in \mathbb{R}^k$ and

$$g_{\boldsymbol{\theta}}(S) = \sum_{s \in S} \theta_s$$

- $\theta_s \in \mathbb{R}$ for every source task s: smaller $\theta_s \Rightarrow s$ is more relevant to t

Remark: Inspired by Datamodels [IPE+22], which uses linear models to approximate the performance of NNs

Estimation

- Sample m subsets: $S_1, S_2, \ldots, S_m$, compute $f(S_i)$
- Estimate $\hat{\theta}$ by minimizing

$$\hat{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(f(S_i) - g_{\boldsymbol{\theta}}(S_i)\right)^2$$

Surrogate models

Estimate a surrogate model to approximate the value function

- $g_{\boldsymbol{\theta}}(S)$: A surrogate model parameterized by $\boldsymbol{\theta}$
- Linear surrogate model: $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_k) \in \mathbb{R}^k$ and

$$g_{\boldsymbol{\theta}}(S) = \sum_{s \in S} \theta_s$$

- $\theta_s \in \mathbb{R}$ for every source task s: smaller $\theta_s \Rightarrow s$ is more relevant to t

Remark: Inspired by Datamodels [IPE+22], which uses linear models to approximate the performance of NNs

Estimation

- Sample m subsets: $S_1, S_2, \ldots, S_m$, compute $f(S_i)$
- Estimate $\hat{\theta}$ by minimizing

$$\hat{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(f(S_i) - g_{\boldsymbol{\theta}}(S_i)\right)^2$$

Surrogate models

Estimate a surrogate model to approximate the value function

- $g_{\boldsymbol{\theta}}(S)$: A surrogate model parameterized by $\boldsymbol{\theta}$
- Linear surrogate model: $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_k) \in \mathbb{R}^k$ and

$$g_{\boldsymbol{\theta}}(S) = \sum_{s \in S} \theta_s$$

- $\theta_s \in \mathbb{R}$ for every source task s: smaller $\theta_s \Rightarrow s$ is more relevant to t

Remark: Inspired by Datamodels [IPE+22], which uses linear models to approximate the performance of NNs

Estimation

- Sample m subsets: $S_1, S_2, \ldots, S_m$, compute $f(S_i)$
- Estimate $\hat{\theta}$ by minimizing

$$\hat{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(f(S_i) - g_{\boldsymbol{\theta}}(S_i)\right)^2$$

Surrogate models

Estimate a surrogate model to approximate the value function

- $g_{\boldsymbol{\theta}}(S)$: A surrogate model parameterized by $\boldsymbol{\theta}$
- Linear surrogate model: $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_k) \in \mathbb{R}^k$ and

$$g_{\boldsymbol{\theta}}(S) = \sum_{s \in S} \theta_s$$

- $\theta_s \in \mathbb{R}$ for every source task s: smaller $\theta_s \Rightarrow s$ is more relevant to t

Remark: Inspired by Datamodels [IPE+22], which uses linear models to approximate the performance of NNs

Estimation

- Sample m subsets: $S_1, S_2, \ldots, S_m$, compute $f(S_i)$
- Estimate $\hat{\theta}$ by minimizing

$$\hat{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(f(S_i) - g_{\boldsymbol{\theta}}(S_i)\right)^2$$

Surrogate models

Estimate a surrogate model to approximate the value function

- $g_{\boldsymbol{\theta}}(S)$: A surrogate model parameterized by $\boldsymbol{\theta}$
- Linear surrogate model: $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_k) \in \mathbb{R}^k$ and

$$g_{\boldsymbol{\theta}}(S) = \sum_{s \in S} \theta_s$$

- $\theta_s \in \mathbb{R}$ for every source task s: smaller $\theta_s \Rightarrow s$ is more relevant to t

Remark: Inspired by Datamodels [IPE+22], which uses linear models to approximate the performance of NNs

Estimation

- Sample m subsets: $S_1, S_2, \ldots, S_m$, compute $f(S_i)$
- Estimate $\hat{\boldsymbol{\theta}}$ by minimizing

$$\hat{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(f(S_i) - g_{\boldsymbol{\theta}}(S_i)\right)^2$$

Best subset selection using surrogate models

Task modeling [LNZ23]

Goal: find the subset S that has the smallest $f(S)$

- **Fit a surrogate model**: estimate $\hat{\boldsymbol{\theta}}$ in the linear surrogate model $g_{\boldsymbol{\theta}}(S)$
- **Select S**: a source task is selected if $\hat{\theta}_s < \gamma$ for a threshold γ, and $S^* = \{s : \hat{\theta}_s < \gamma\}$

Best subset selection using surrogate models

Task modeling [LNZ23]

Goal: find the subset S that has the smallest $f(S)$

- **Fit a surrogate model**: estimate $\hat{\theta}$ in the linear surrogate model $g_{\boldsymbol{\theta}}(S)$
- **Select S**: a source task is selected if $\hat{\theta}_s < \gamma$ for a threshold γ, and $S^* = \{s : \hat{\theta}_s < \gamma\}$

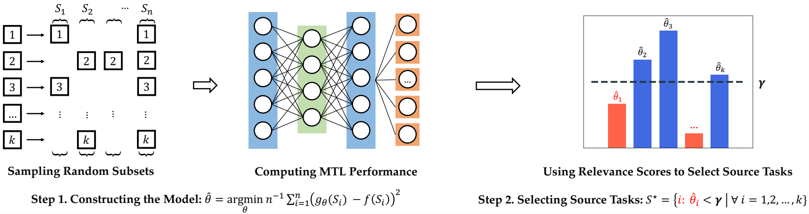


Figure 5: Illustration of two-step procedure in task modeling

Theoretical results: linear sample complexity and running time to estimate $\boldsymbol{\theta}$ in the number of source tasks k

Experimental results: experiments on weak supervision, GLUE, and FolkTables (multi-group fairness)

- > 0.8 F_1-score for predicting negative transfers on **unseen subsets**
- $> 4\%$ performance improvement in weak supervision data sets over existing methods

Outline

Higher-order task relationships

Motivation: The effect of adding a source task depends on what other tasks are in S

Example: Source task 1 is a negative task, while source task 2 is a positive task to target task t

- The OOS error of $S = \{1\}$ is higher than that of $S = \{\emptyset\}$
- The OOS error of $S = \{2\}$ is lower than that of $S = \{\emptyset\}$
- The OOS error of $S = \{1, 2\}$ is higher than that of $S = \{1\}$

Higher-order task relationships

Motivation: The effect of adding a source task depends on what other tasks are in S

Example: Source task 1 is a negative task, while source task 2 is a positive task to target task t

- The OOS error of $S = \{1\}$ is higher than that of $S = \{\emptyset\}$
- The OOS error of $S = \{2\}$ is lower than that of $S = \{\emptyset\}$
- The OOS error of $S = \{1, 2\}$ is higher than that of $S = \{1\}$

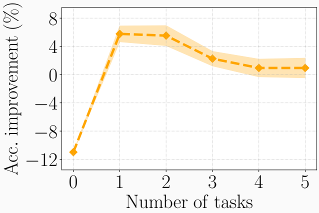


Figure 6: Accuracy improvement for different number of tasks: 1 indicates $S = \{\emptyset\}$ (only trained on target task), 2 indicates $S = \{1\}$ and 3 indicates $S = \{1, 2\}$.

Higher-order task affinity

Define a higher-order value function to quantify "discrepancy" between source task s and target task t accounting for presence of other source tasks

Definition

Let $\bar{f}(s)$ be the average **out-of-sample error** on target task t over all possible subsets of source tasks $S \subseteq \{1, 2, \cdots, k-1\}$ that contain task s

$$\bar{f}(s) = \frac{1}{2^{k-2}} \sum_{S \in \{1, \cdots, k-1\} : s \in S} f(S),$$

where 2^{k-2} is the number of subsets of $\{1, \cdots, k-1\}$ that includes the source task s.

Estimation: Randomly sample m subsets $S_1, S_2, \ldots, S_m$, average over the subsets that include s, t, for every s and t in $1, 2, \ldots, k$

Higher-order task affinity

Define a higher-order value function to quantify "discrepancy" between source task s and target task t accounting for presence of other source tasks

Definition

Let $\bar{f}(s)$ be the average **out-of-sample error** on target task t over all possible subsets of source tasks $S \subseteq \{1, 2, \cdots, k-1\}$ that contain task s

$$\bar{f}(s) = \frac{1}{2^{k-2}} \sum_{S \in \{1, \cdots, k-1\} : s \in S} f(S),$$

where 2^{k-2} is the number of subsets of $\{1, \cdots, k-1\}$ that includes the source task s.

Estimation: Randomly sample m subsets $S_1, S_2, \ldots, S_m$, average over the subsets that include s, t, for every s and t in $1, 2, \ldots, k$

Higher-order task affinity

Define a higher-order value function to quantify "discrepancy" between source task s and target task t accounting for presence of other source tasks

Definition

Let $\bar{f}(s)$ be the average **out-of-sample error** on target task t over all possible subsets of source tasks $S \subseteq \{1, 2, \cdots, k-1\}$ that contain task s

$$\bar{f}(s) = \frac{1}{2^{k-2}} \sum_{S \in \{1, \cdots, k-1\} : s \in S} f(S),$$

where 2^{k-2} is the number of subsets of $\{1, \cdots, k-1\}$ that includes the source task s.

Estimation: Randomly sample m subsets $S_1, S_2, \ldots, S_m$, average over the subsets that include s, t, for every s and t in $1, 2, \ldots, k$

Task grouping

Boosting procedure [LJS+23]

1. Estimate a k by k higher-order task affinity matrix, denoted as $\hat{M}$
2. Find a clustering of $1, 2, \ldots, k$ into $S_1, S_2, \ldots, S_{|C|}$ by maximizing the average density within each cluster
3. For each cluster S_i, train a separate NN for tasks within that subset

Choices of clustering

- Spectral clustering
- Lloyd's algorithm
- Semi-definite programming relaxation

Conceptually similar to boosting (bagging more precisely)

Task grouping

Boosting procedure [LJS+23]

1. Estimate a k by k higher-order task affinity matrix, denoted as $\hat{M}$
2. Find a clustering of $1, 2, \ldots, k$ into $S_1, S_2, \ldots, S_{|C|}$ by maximizing the average density within each cluster
3. For each cluster S_i, train a separate NN for tasks within that subset

Choices of clustering

- Spectral clustering
- Lloyd's algorithm
- Semi-definite programming relaxation

Conceptually similar to boosting (bagging more precisely)

Task grouping

Boosting procedure [LJS+23]

1. Estimate a k by k higher-order task affinity matrix, denoted as $\hat{M}$
2. Find a clustering of $1, 2, \ldots, k$ into $S_1, S_2, \ldots, S_{|C|}$ by maximizing the average density within each cluster
3. For each cluster S_i, train a separate NN for tasks within that subset

Choices of clustering

- Spectral clustering
- Lloyd's algorithm
- Semi-definite programming relaxation

Conceptually similar to boosting (bagging more precisely)

Experimental results

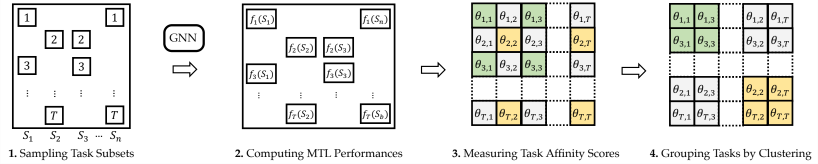


Figure 7: Illustration of the boosting procedure

We find that this boosting procedure can outperform naive multitask learning by $\sim$ 4% and existing task grouping methods by $>$ 2%

Efficient algorithms for computing task affinity

The boosting procedure requires repeatedly training many multitask models, which is still computationally expensive

Examples

- Higher-order task affinity: $m = O(k)$ random subsets $S_1, S_2, \ldots, S_m$, each of size α
- Pairwise task affinity: $\binom{k}{2}$ subsets, including $\{1,2\}, \{1,3\}, \ldots, \{1,k\}$, $\{2,3\}, \ldots, \{2,k\}, \ldots, \{k-1,k\}$
- Forward selection: $\binom{k}{2}$ subsets, including $\{1\}, \{1,2\}, \ldots, \{1,k\}$; if i_1 is selected, then $\{1,i_1,2\}, \{1,i_1,3\}, \ldots, \{1,i_i,k\}$; and so on

Can we enable this computation without this repeated multitask model training?

Efficient algorithms for computing task affinity

The boosting procedure requires repeatedly training many multitask models, which is still computationally expensive

Examples

- Higher-order task affinity: $m = O(k)$ random subsets $S_1, S_2, \ldots, S_m$, each of size α
- Pairwise task affinity: $\binom{k}{2}$ subsets, including $\{1,2\}, \{1,3\}, \ldots, \{1,k\}$, $\{2,3\}, \ldots, \{2,k\}, \ldots, \{k-1,k\}$
- Forward selection: $\binom{k}{2}$ subsets, including $\{1\}, \{1,2\}, \ldots, \{1,k\}$; if i_1 is selected, then $\{1, i_1, 2\}, \{1, i_1, 3\}, \ldots, \{1, i_i, k\}$; and so on

Can we enable this computation without this repeated multitask model training?

Efficient algorithms for computing task affinity

The boosting procedure requires repeatedly training many multitask models, which is still computationally expensive

Examples

- Higher-order task affinity: $m = O(k)$ random subsets $S_1, S_2, \ldots, S_m$, each of size α
- Pairwise task affinity: $\binom{k}{2}$ subsets, including $\{1,2\}, \{1,3\}, \ldots, \{1,k\}$, $\{2,3\}, \ldots, \{2,k\}, \ldots, \{k-1,k\}$
- Forward selection: $\binom{k}{2}$ subsets, including $\{1\}, \{1,2\}, \ldots, \{1,k\}$; if i_1 is selected, then $\{1,i_1,2\}, \{1,i_1,3\}, \ldots, \{1,i_i,k\}$; and so on

Can we enable this computation without this repeated multitask model training?

Gradient-based estimation

First train a meta-initialization on all tasks, then, estimate the fine-tuned model by aligning the gradients at the initialization to the task labels

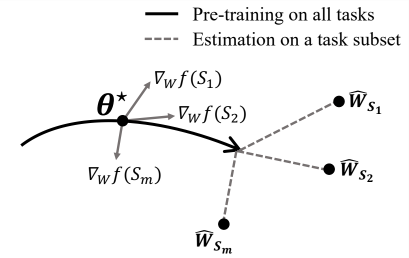


Figure 8: We replace multitask training with a regression-based estimation of model parameters fine-tuned on a particular subset of tasks.

If W (the fine-tuned model parameter) is close to $\theta^\star$ (the meta-initialization model parameter), $f_W(x, y)$ can be approximated by

$$f_W(x, y) \approx f_{\theta^\star}(x, y) + \nabla_W f_{\theta^\star}(x, y)^\top (W - \theta^\star) + \epsilon. \tag{1}$$

Linearization of fine-tuned models

If W (the fine-tuned model parameter) is close to $\theta^\star$ (the meta-initialization model parameter), $f_W(x, y)$ can be approximated by

$$f_W(x, y) \approx f_{\theta^\star}(x, y) + \nabla_W f_{\theta^\star}(x, y)^\top (W - \theta^\star) + \epsilon. \quad (1)$$

Table 1: Measuring ϵ for models fine-tuned from an initialization pre-trained on all tasks. The results are averaged over 100 random task subsets.

GNN		BERT		T5	
Distance	RSS	Distance	RSS	Distance	RSS
1%	4.2×10^{-4}	1%	3.6×10^{-6}	1%	3.8×10^{-6}
2%	9.5×10^{-4}	2%	5.4×10^{-6}	2%	6.0×10^{-5}
3%	1.1×10^{-3}	3%	3.0×10^{-5}	3%	3.2×10^{-5}
4%	2.5×10^{-3}	4%	1.5×10^{-4}	4%	2.6×10^{-4}
5%	6.8×10^{-3}	5%	2.2×10^{-4}	5%	6.3×10^{-4}
6%	7.5×10^{-3}	6%	5.7×10^{-4}	6%	8.4×10^{-4}
7%	9.0×10^{-3}	7%	9.9×10^{-4}	7%	1.4×10^{-3}
8%	9.3×10^{-3}	8%	9.0×10^{-4}	8%	2.5×10^{-3}
9%	1.2×10^{-2}	9%	2.2×10^{-3}	9%	3.3×10^{-3}
10%	3.4×10^{-2}	10%	5.1×10^{-3}	10%	4.1×10^{-3}

Task affinity estimation

Algorithm [Working Paper]

1. Approximate NN output with Taylor's expansion (1), ignoring the ϵ errors
2. Estimate $\hat{W}_{S_i}$ by fitting a logistic regression from $f_W(x, y)$ to y, for every subset S_i

Proposition [Working Paper]

Provided ϵ is small, this algorithm will recover the true loss function accurately

Algorithm [Working Paper]

1. Approximate NN output with Taylor's expansion (1), ignoring the ϵ errors
2. Estimate $\hat{W}_{S_i}$ by fitting a logistic regression from $f_W(x, y)$ to y, for every subset S_i

Proposition [Working Paper]

Provided ϵ is small, this algorithm will recover the true loss function accurately

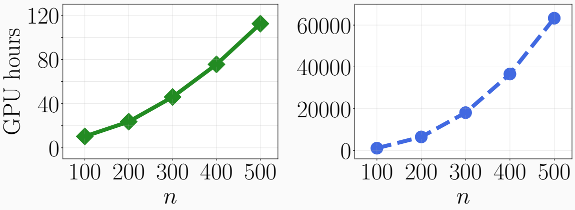
Task affinity estimation

Algorithm [Working Paper]

1. Approximate NN output with Taylor's expansion (1), ignoring the ϵ errors
2. Estimate $\hat{W}_{S_i}$ by fitting a logistic regression from $f_W(x, y)$ to y, for every subset S_i

Proposition [Working Paper]

Provided ϵ is small, this algorithm will recover the true loss function accurately



(a) Our estimation (b) Full training cost

Figure 9: The number of GPU hours vs. the number of tasks to compute pairwise affinity, evaluated on a graph with 21M edges and 500 labeling tasks.

Discussion and open questions

A large part of this work is recent, so there are many opportunities for future work

- Can we better flesh out the connection between boosting and multitask learning? Note there is extensive literature on boosting algorithms for supervised learning
- Apply the gradient-based estimation to large-scale data sets in foundation models?
- Efficiently computing influence functions to capture higher-order correlation in foundation models?

Applications

- Model personalization
- Data privacy

Discussion and open questions

A large part of this work is recent, so there are many opportunities for future work

- Can we better flesh out the connection between boosting and multitask learning? Note there is extensive literature on boosting algorithms for supervised learning
- Apply the gradient-based estimation to large-scale data sets in foundation models?
- Efficiently computing influence functions to capture higher-order correlation in foundation models?

Applications

- Model personalization
- Data privacy

Discussion and open questions

A large part of this work is recent, so there are many opportunities for future work

- Can we better flesh out the connection between boosting and multitask learning? Note there is extensive literature on boosting algorithms for supervised learning
- Apply the gradient-based estimation to large-scale data sets in foundation models?
- Efficiently computing influence functions to capture higher-order correlation in foundation models?

Applications

- Model personalization
- Data privacy

Discussion and open questions

A large part of this work is recent, so there are many opportunities for future work

- Can we better flesh out the connection between boosting and multitask learning? Note there is extensive literature on boosting algorithms for supervised learning
- Apply the gradient-based estimation to large-scale data sets in foundation models?
- Efficiently computing influence functions to capture higher-order correlation in foundation models?

Applications

- Model personalization
- Data privacy

Conclusion

This talk covers our work to develop the algorithmic foundations of multitask learning. Key takeaways:

- Using linear surrogate models can accurately identify negative transfers
- Boosting helps multitask learning performance when task relationships are highly complex

Many open directions and deeper connections to deep learning theory, boosting, influence functions, differential privacy for future work

Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. (2019). **"Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks".** In: *International Conference on Machine Learning*. PMLR, pp. 322–332 (20–25).

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018). **"Stronger generalization bounds for deep nets via a compression approach".** In: *International Conference on Machine Learning*. PMLR, pp. 254–263 (20–25).

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). **"Spectrally-normalized margin bounds for neural networks".** In: *Advances in neural information processing systems* 30 (20–25).

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020). **"Benign overfitting in linear regression".** In: *Proceedings of the National Academy of Sciences* 117.48, pp. 30063–30070 (20–25).

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). **"A theory of learning from different domains".** In: *Machine learning* 79, pp. 151–175 (40–42).

Caruana, R. (1997). **"Multitask learning".** In: *Machine learning* 28, pp. 41–75 (27, 28).

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. (2022). **"Datamodels: Predicting predictions from training data"**. In: *ICML* (49–53).

Ju, H., Li, D., and Zhang, H. R. (2022). **"Robust fine-tuning of deep neural networks with hessian-based generalization guarantees"**. In: *International Conference on Machine Learning*. PMLR, pp. 10431–10461 (20–25).

Li, D., Ju, H., Sharma, A., and Zhang, H. R. (2023a). **"Boosting Multitask Learning on Graphs through Higher-Order Task Affinities"**. In: *Proceedings of the 29nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1213–1222 (14–19, 63–65).

Li, D., Nguyen, H. L., and Zhang, H. R. (2023b). **"Identification of Negative Transfers in Multitask Learning using Surrogate Models"**. In: *Transactions on Machine Learning Research (***Featured Certification***)*. URL: https://openreview.net/forum?id=KgfFAI9f3E (14–19, 43–48, 54, 55).

Li, Y., Ma, T., and Zhang, H. (2018). **"Algorithmic Regularization in Over-parameterized Matrix Sensing and Neural Networks with Quadratic Activations"**. In: *Conference On Learning Theory* (20–25).

Sur, P. and Candès, E. J. (2019). **"A modern maximum-likelihood theory for high-dimensional logistic regression".** In: *Proceedings of the National Academy of Sciences* 116.29, pp. 14516–14525 (20–25).

Wu, S., Zhang, H. R., and Ré, C. (2020). **"Understanding and Improving Information Transfer in Multi-Task Learning".** In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SylzhkBtDB (14–19, 29–32).

Yang, F., Zhang, H. R., Wu, S., Ré, C., and Su, W. J. (2020). **"Precise High-dimensional Asymptotics for Quantifying Heterogeneous Transfers".** In: *arXiv preprint arXiv:2010.11750* (14–19, 33–37).

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). **"Understanding deep learning (still) requires rethinking generalization".** In: *Communications of the ACM* 64.3, pp. 107–115 (2–7).