Kernels, Random Embeddings and Deep Learning

Vikas Sindhwani

Google

Tuesday 26th April, 2016

Topics

- ORF523: Computational Tractability, Convex Optimization, SDPs, Polynomial Optimization and SoS

Topics

- ORF523: Computational Tractability, Convex Optimization, SDPs, Polynomial Optimization and SoS
- Machine Learning: rich source of optimization problems, but with a twist.

Topics

- ORF523: Computational Tractability, Convex Optimization, SDPs, Polynomial Optimization and SoS
- Machine Learning: rich source of optimization problems, but with a twist.
- Nonlinear techniques at the opposite ends of tractability.
  - **Kernel Methods**: Convex Optimization, Learning polynomial functions and beyond
  - **Deep Learning**: Current dominant ML practice from an industry perspective.
    - TensorFlow
    - CNNs and RNNs
  - **Random embeddings**: approximate kernel methods or approximate neural networks.

Setting

Estimate an unknown dependency,

$$f : \mathcal{X} \mapsto \mathcal{Y},$$

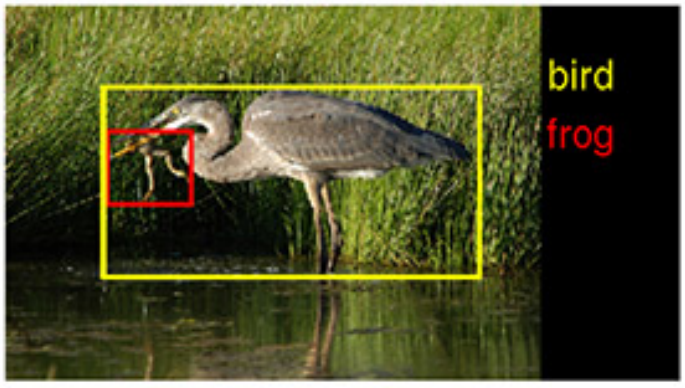
given (manually) labeled data, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l} \sim p$, $\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}$.

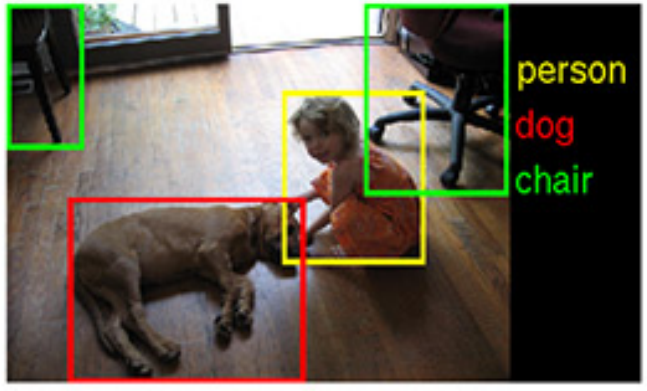
Setting

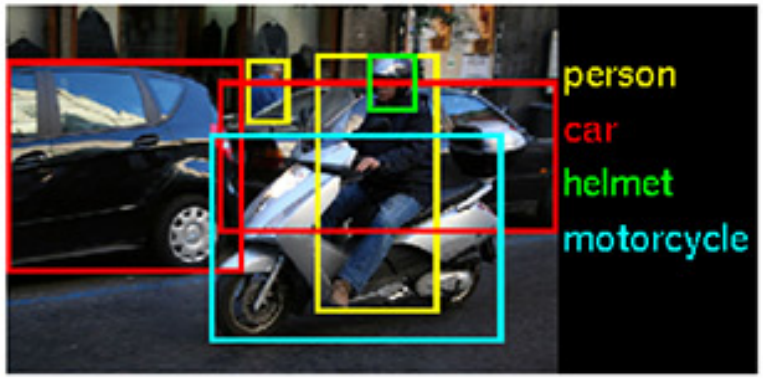
Estimate an unknown dependency,

$$f : \mathcal{X} \mapsto \mathcal{Y},$$

given (manually) labeled data, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l} \sim p$, $\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}$.







Setting

Estimate an unknown dependency,

$$f : \mathcal{X} \mapsto \mathcal{Y},$$

given (manually) labeled data, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l} \sim p$, $\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}$.

A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

A skateboarder does a trick on a ramp.

A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

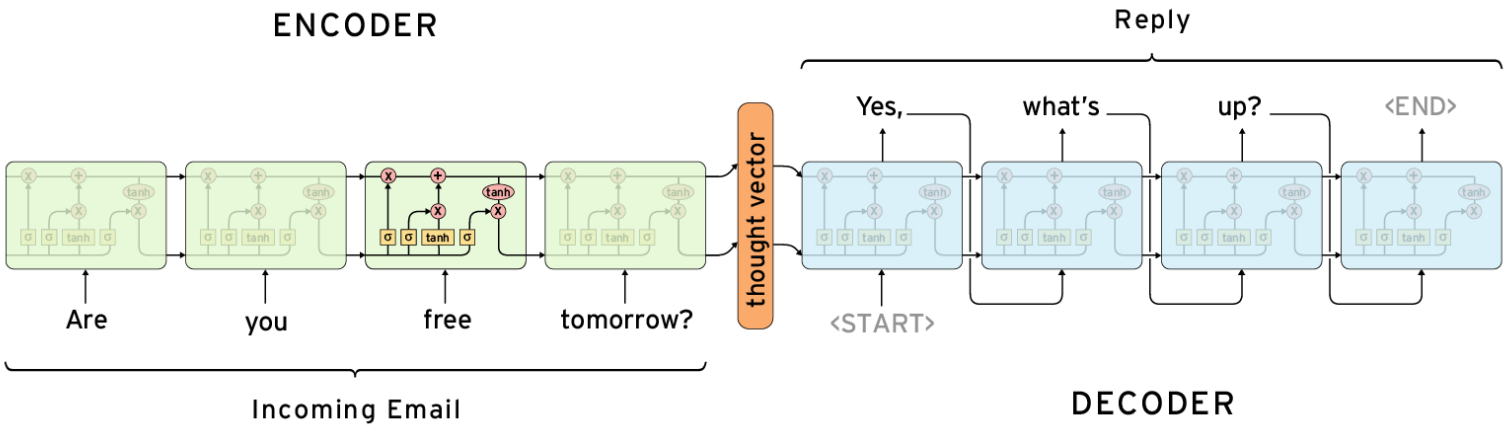
A little girl in a pink hat is blowing bubbles.

Setting

Estimate an unknown dependency,

$$f : \mathcal{X} \mapsto \mathcal{Y},$$

given (manually) labeled data, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l} \sim p$, $\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}$.

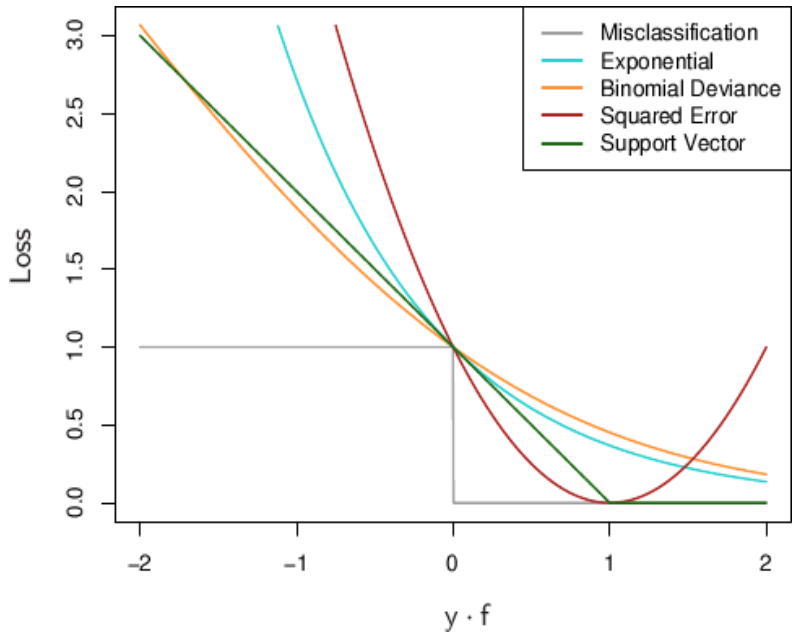


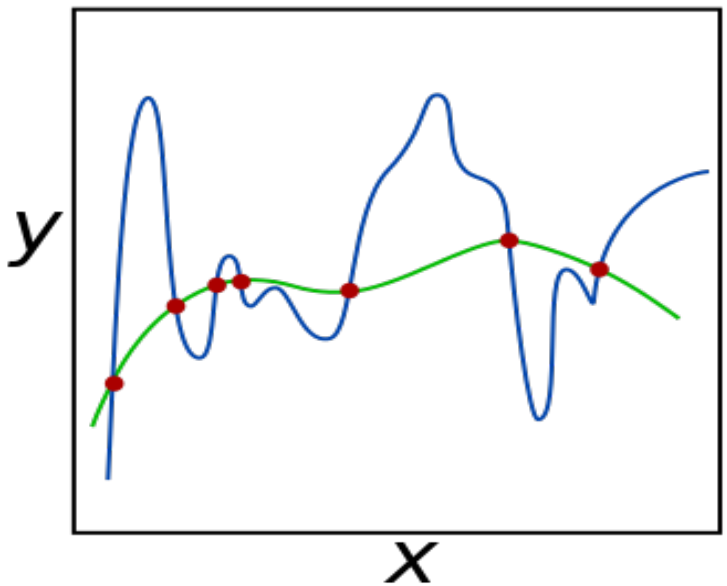
`google.com/inbox`

Regularized Loss Minimization

- Regularized Loss Minimization in a suitable hypothesis space $\mathcal{H}$,

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^{l} V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$





Optimization in Learning

- **Optimal predictor**

$$f^\star = \arg\min_f \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

Optimization in Learning

- **Optimal predictor**

$$f^\star = \arg\min_f \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Approximation Error**: due to finite domain knowledge

$$f^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

Optimization in Learning

- **Optimal predictor**

$$f^\star = \arg\min_f \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Approximation Error**: due to finite domain knowledge

$$f^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Estimation Error**: due to finite data

$$\hat{f}^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \sum_{i=1}^{l} V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

Optimization in Learning

- **Optimal predictor**

$$f^\star = \arg\min_f \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Approximation Error**: due to finite domain knowledge

$$f^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Estimation Error**: due to finite data

$$\hat{f}^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \sum_{i=1}^{l} V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

- **Optimization Error**: finite computation

$$\text{return:} \quad \hat{f}^{(i)\star}_{\mathcal{H}}$$

Optimization in Learning

- **Optimal predictor**

$$f^\star = \arg\min_f \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Approximation Error**: due to finite domain knowledge

$$f^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} V(f(\mathbf{x}), \mathbf{y})$$

- **Estimation Error**: due to finite data

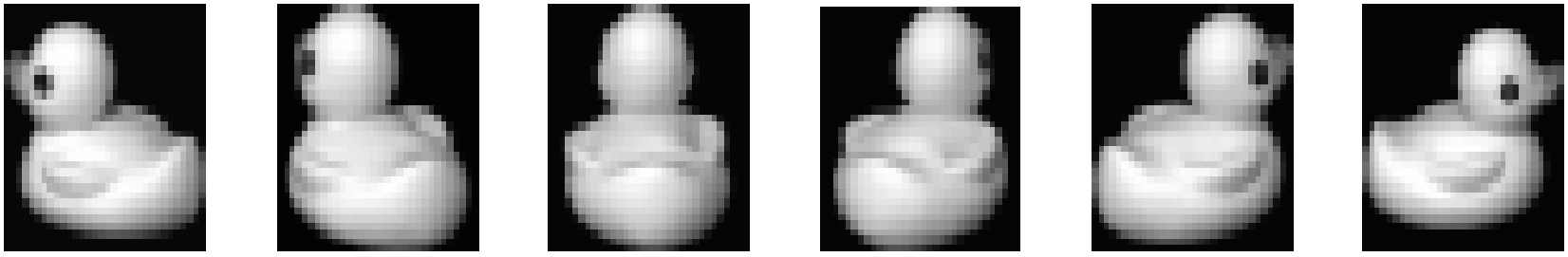
$$\hat{f}^\star_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \sum_{i=1}^{l} V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

- **Optimization Error**: finite computation

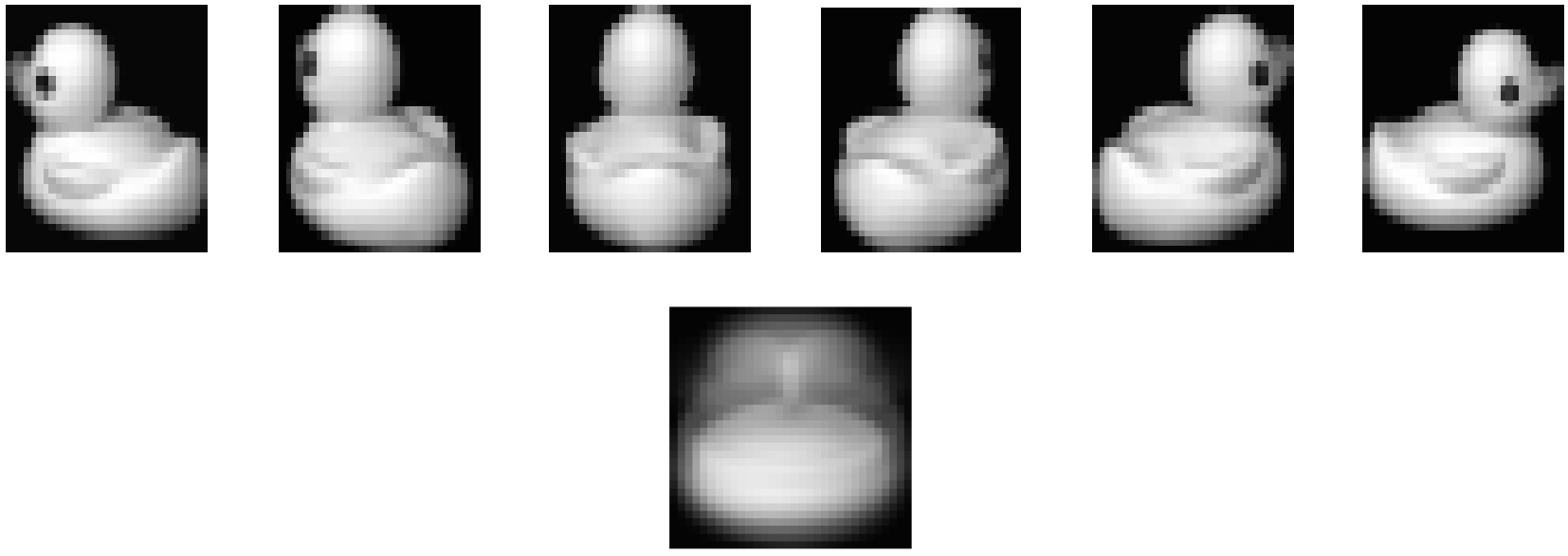
$$\text{return:} \quad \hat{f}^{(i)\star}_{\mathcal{H}}$$

Tractable but bad model, or intractable but good model?

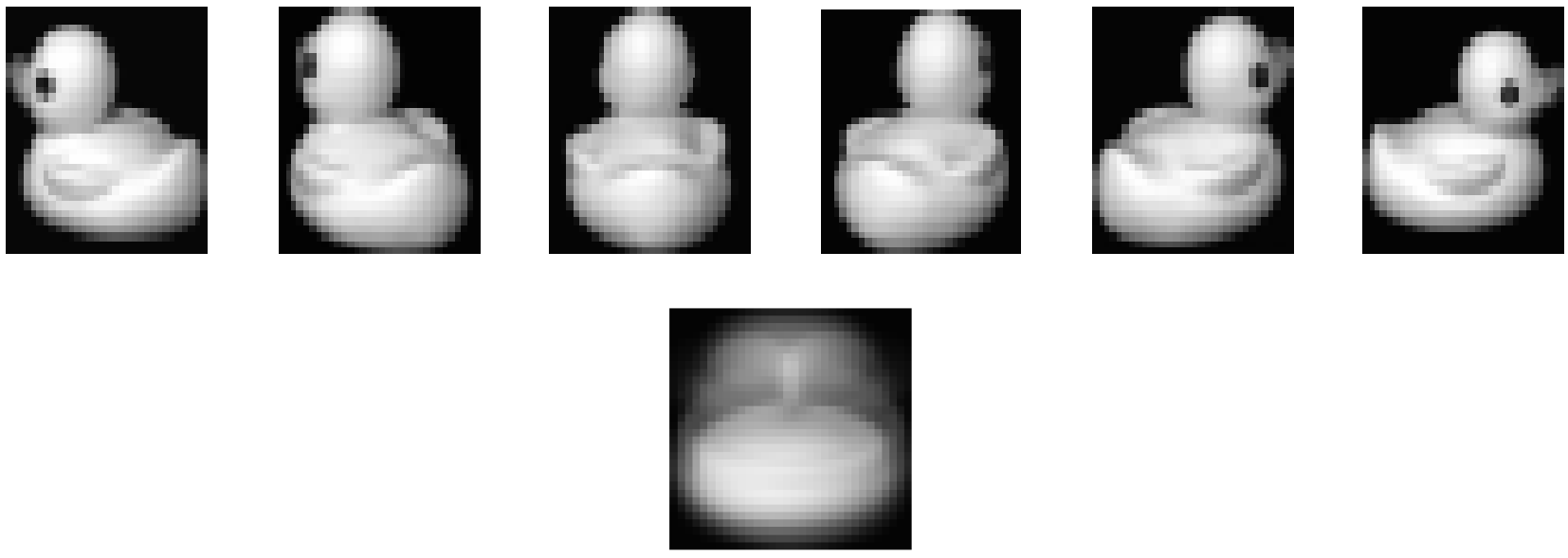
Nonlinearities Everywhere!



Nonlinearities Everywhere!

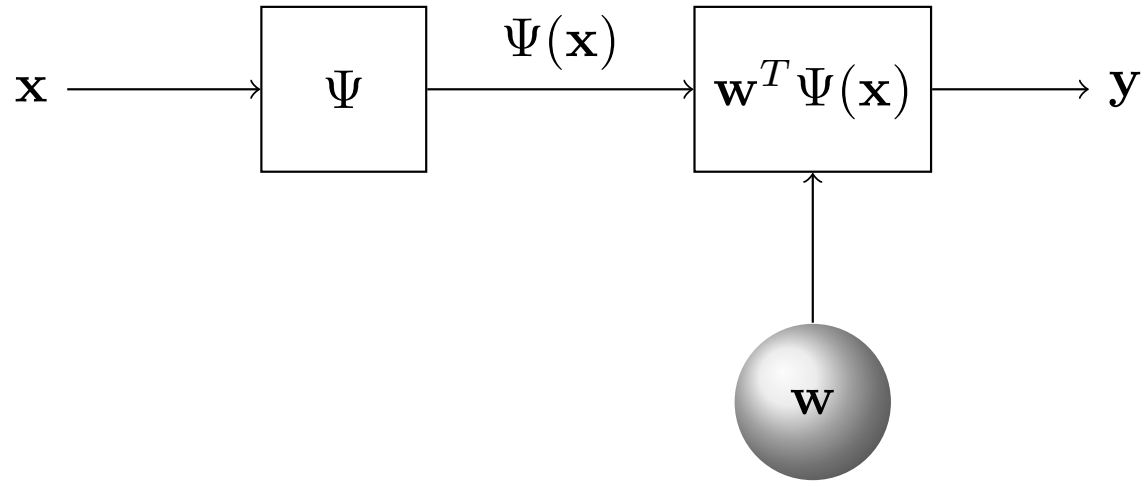


Nonlinearities Everywhere!



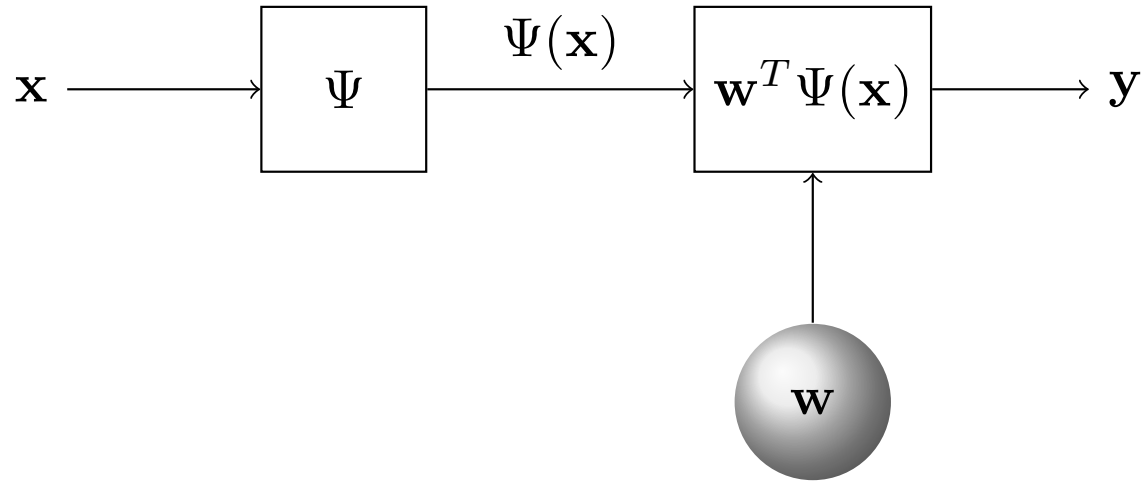
Large $l \implies$ Big models: $\mathcal{H}$ "rich" /non-parametric/nonlinear.

Choice of Nonlinear Hypothesis Space: Kernels



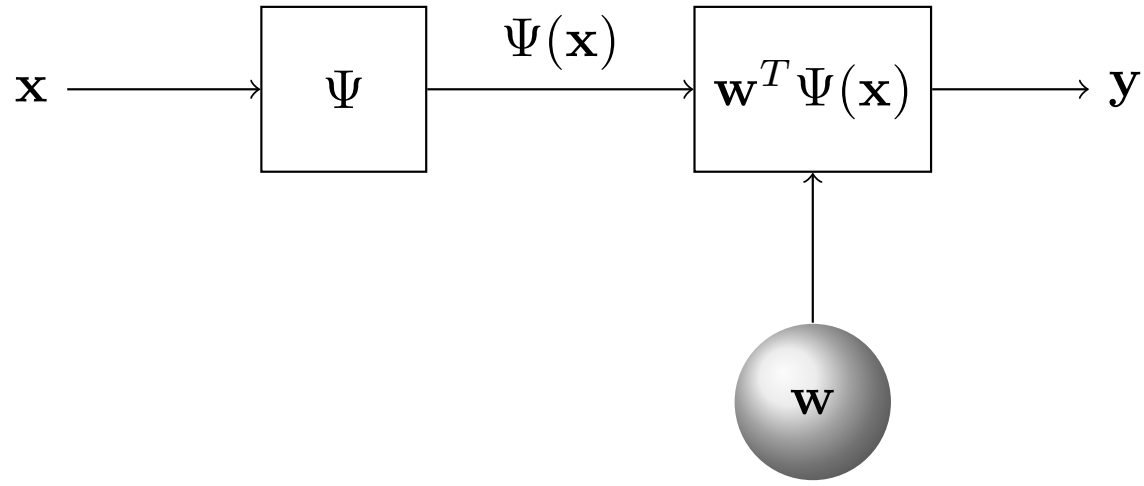
- Very high-dimensional (infinite) nonlinear embeddings + linear models. Fully non-parameteric.

Choice of Nonlinear Hypothesis Space: Kernels



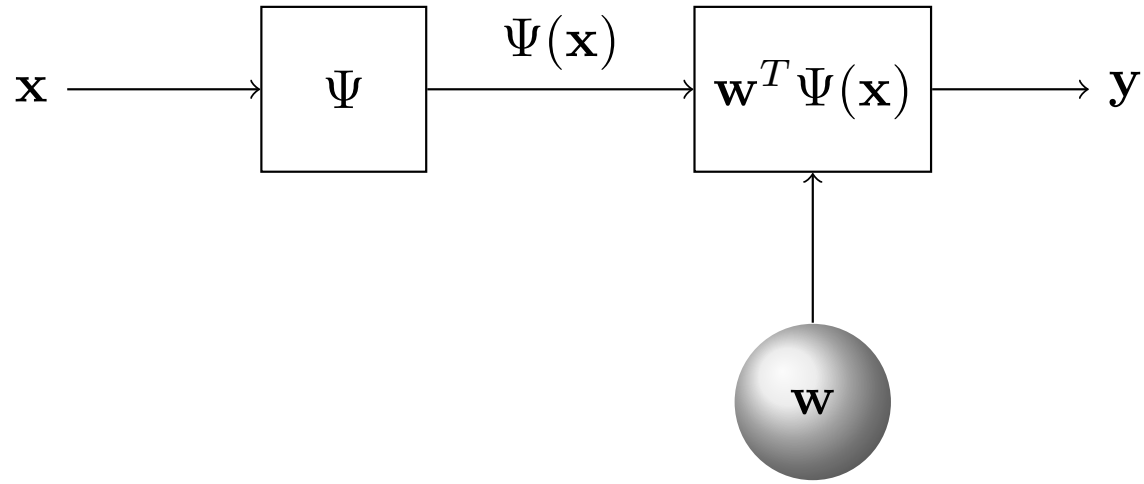
- Very high-dimensional (infinite) nonlinear embeddings + linear models. Fully non-parameteric.
- Functional Analysis (Aronszajn, Bergman (1950s)); Statistics (Parzen (1960s)); PDEs; Numerical Analysis. . .

Choice of Nonlinear Hypothesis Space: Kernels



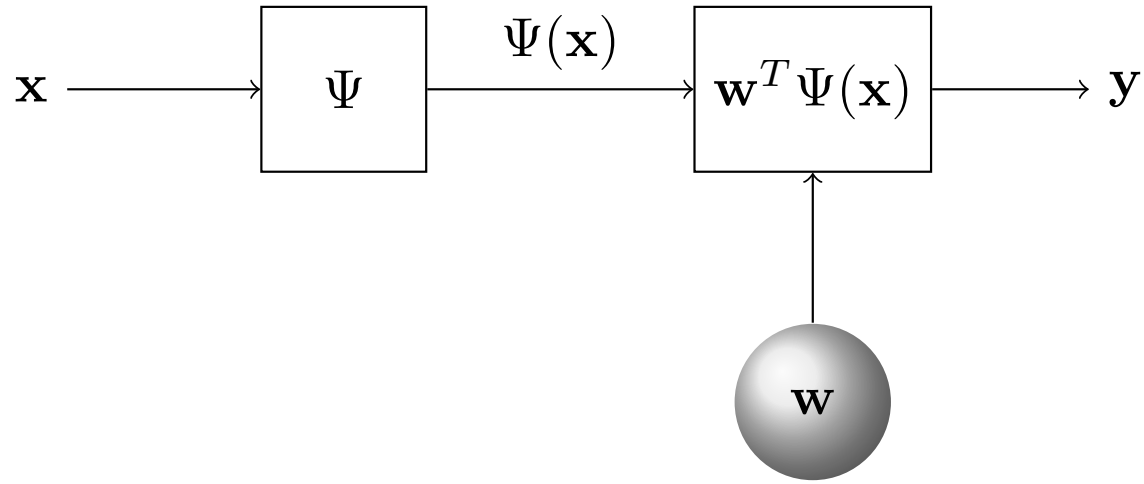
- Very high-dimensional (infinite) nonlinear embeddings + linear models. Fully non-parameteric.
- Functional Analysis (Aronszajn, Bergman (1950s)); Statistics (Parzen (1960s)); PDEs; Numerical Analysis. . .
- ML in 1990's: Nonlinear classification, regression, clustering, time-series analysis, dynamical systems, hypothesis testing, causal modeling, . . .

Choice of Nonlinear Hypothesis Space: Kernels



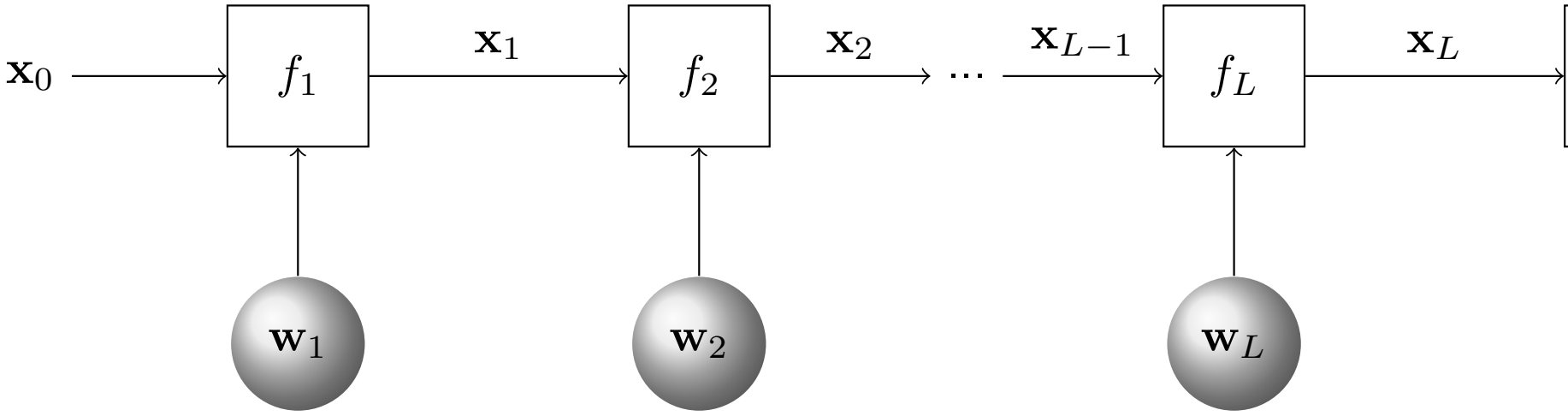
- Very high-dimensional (infinite) nonlinear embeddings + linear models. Fully non-parameteric.
- Functional Analysis (Aronszajn, Bergman (1950s)); Statistics (Parzen (1960s)); PDEs; Numerical Analysis. . .
- ML in 1990's: Nonlinear classification, regression, clustering, time-series analysis, dynamical systems, hypothesis testing, causal modeling, . . .
- Convex Optimization.

Choice of Nonlinear Hypothesis Space: Kernels



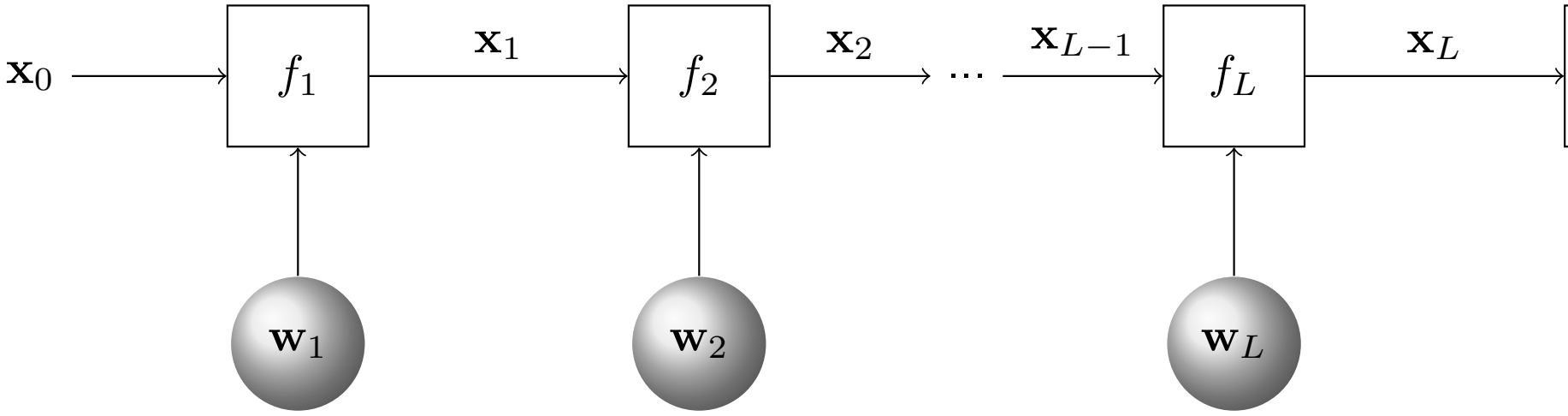
- Very high-dimensional (infinite) nonlinear embeddings + linear models. Fully non-parameteric.
- Functional Analysis (Aronszajn, Bergman (1950s)); Statistics (Parzen (1960s)); PDEs; Numerical Analysis. . .
- ML in 1990's: Nonlinear classification, regression, clustering, time-series analysis, dynamical systems, hypothesis testing, causal modeling, . . .
- Convex Optimization.

Choice of Nonlinear Hypothesis Space: Deep Learning



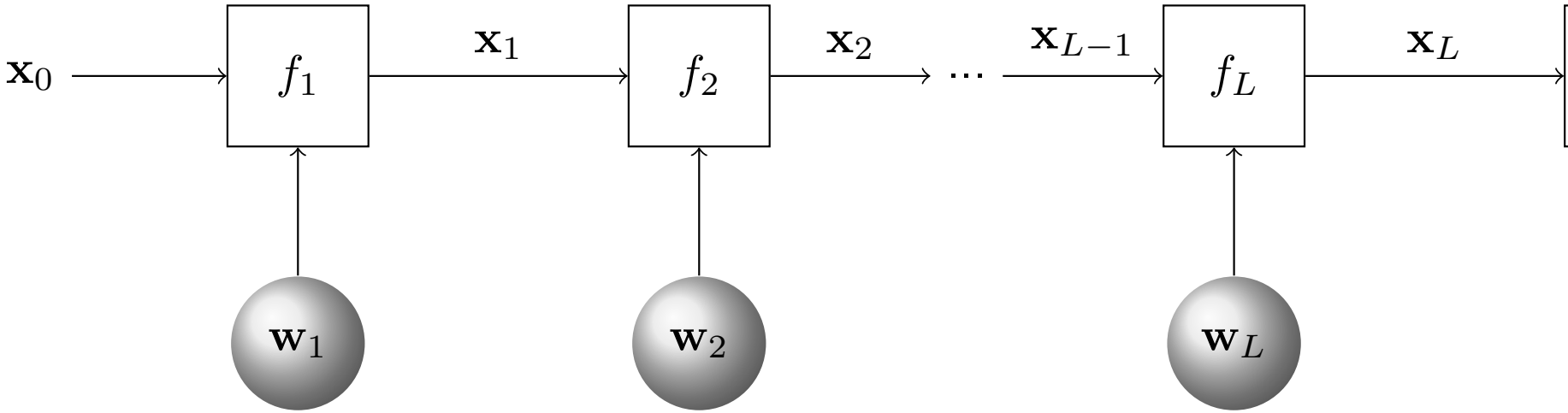
- Compositions: Raw data to higher abstractions (representation learning)

Choice of Nonlinear Hypothesis Space: Deep Learning



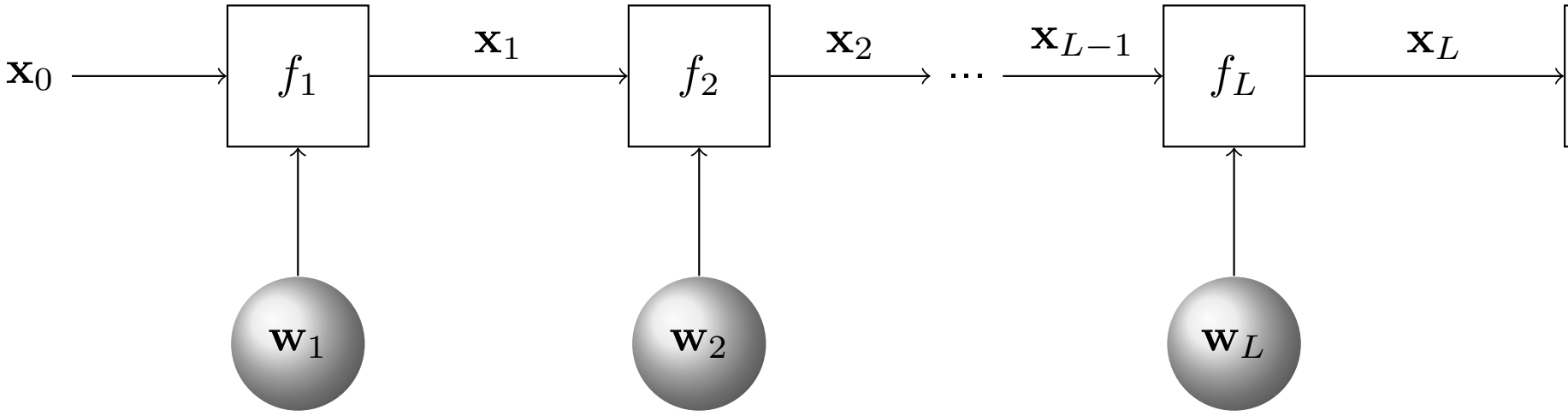
- Compositions: Raw data to higher abstractions (representation learning)
- Multilayer Perceptrons: $f_l(\mathbf{x}, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x})$ (sgmd, ReLU - stat/opt)

Choice of Nonlinear Hypothesis Space: Deep Learning



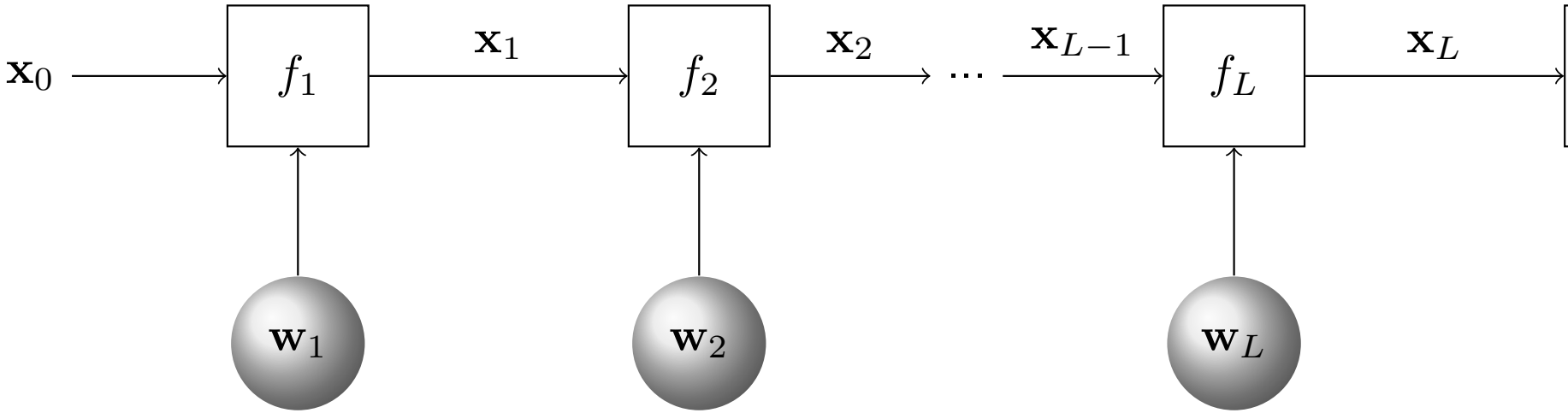
- Compositions: Raw data to higher abstractions (representation learning)
- Multilayer Perceptrons: $f_l(\mathbf{x}, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x})$ (sgmd, ReLU - stat/opt)
- Rosenblatt (1957), Perceptron (Minski and Pappert, 1969), Backprop (1986), Reverse-mode differentiation (1960s-70s), CNNs (1980s, 2012-)

Choice of Nonlinear Hypothesis Space: Deep Learning



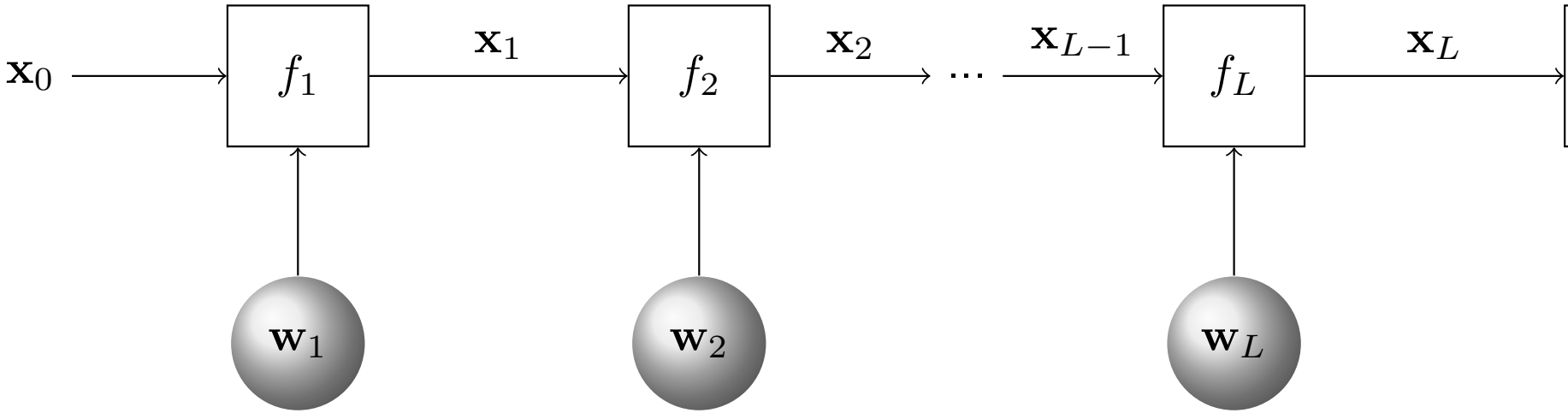
- Compositions: Raw data to higher abstractions (representation learning)
- Multilayer Perceptrons: $f_l(\mathbf{x}, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x})$ (sgmd, ReLU - stat/opt)
- Rosenblatt (1957), Perceptron (Minski and Pappert, 1969), Backprop (1986), Reverse-mode differentiation (1960s-70s), CNNs (1980s, 2012-)
- Turing (1950): In considering the functions of the mind or the brain we find certain operations which we can explain in purely mechanical terms. This we say does not correspond to the real mind: it is a sort of skin which we must strip off if we are to find the real mind. Proceeding in this way do we ever come to the "real" mind, or do we eventually come to the skin which has nothing in it? In the latter case the whole mind is mechanical.

Choice of Nonlinear Hypothesis Space: Deep Learning



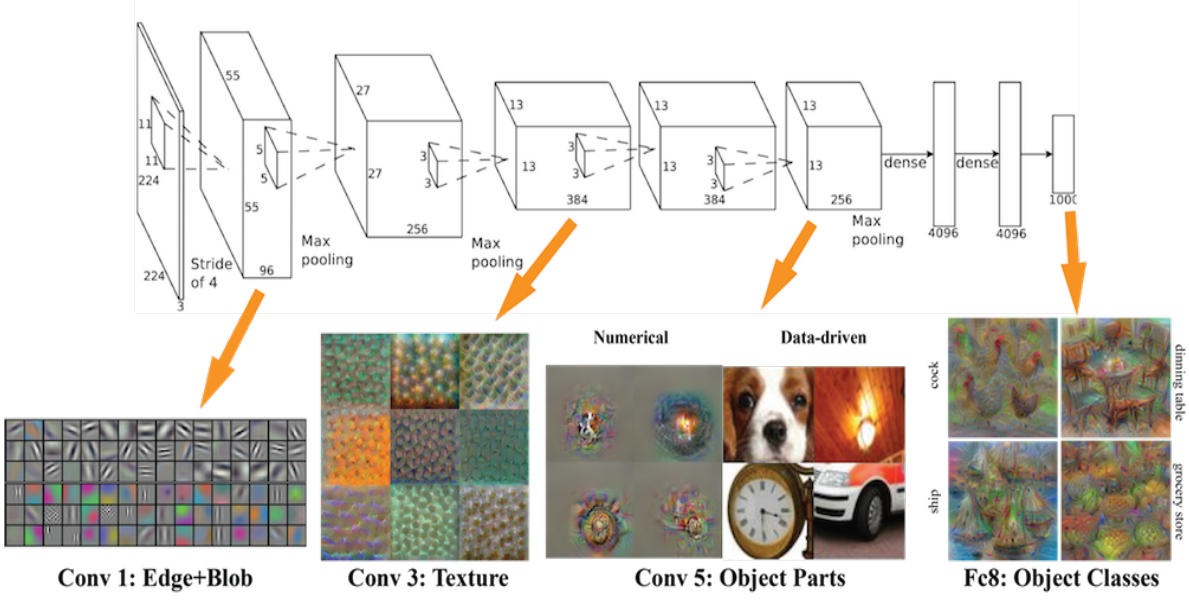
- Compositions: Raw data to higher abstractions (representation learning)
- Multilayer Perceptrons: $f_l(\mathbf{x}, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x})$ (sgmd, ReLU - stat/opt)
- Rosenblatt (1957), Perceptron (Minski and Pappert, 1969), Backprop (1986), Reverse-mode differentiation (1960s-70s), CNNs (1980s, 2012-)
- Turing (1950): In considering the functions of the mind or the brain we find certain operations which we can explain in purely mechanical terms. This we say does not correspond to the real mind: it is a sort of skin which we must strip off if we are to find the real mind. Proceeding in this way do we ever come to the "real" mind, or do we eventually come to the skin which has nothing in it? In the latter case the whole mind is mechanical.
- Non-convex Optimization. Domain-agnostic recipe.

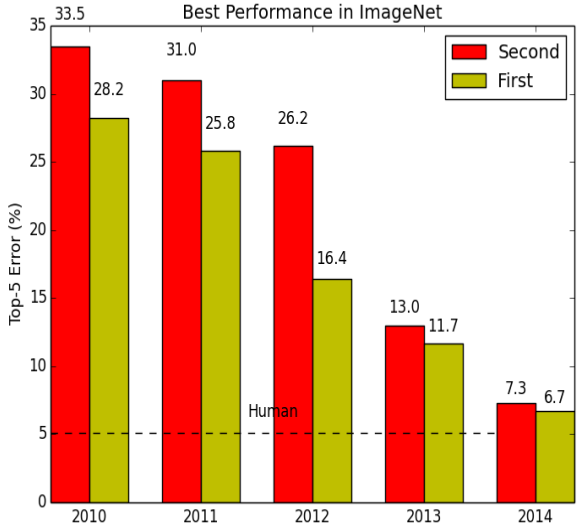
Choice of Nonlinear Hypothesis Space: Deep Learning



- Compositions: Raw data to higher abstractions (representation learning)
- Multilayer Perceptrons: $f_l(\mathbf{x}, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x})$ (sgmd, ReLU - stat/opt)
- Rosenblatt (1957), Perceptron (Minski and Pappert, 1969), Backprop (1986), Reverse-mode differentiation (1960s-70s), CNNs (1980s, 2012-)
- Turing (1950): In considering the functions of the mind or the brain we find certain operations which we can explain in purely mechanical terms. This we say does not correspond to the real mind: it is a sort of skin which we must strip off if we are to find the real mind. Proceeding in this way do we ever come to the "real" mind, or do we eventually come to the skin which has nothing in it? In the latter case the whole mind is mechanical.
- Non-convex Optimization. Domain-agnostic recipe.

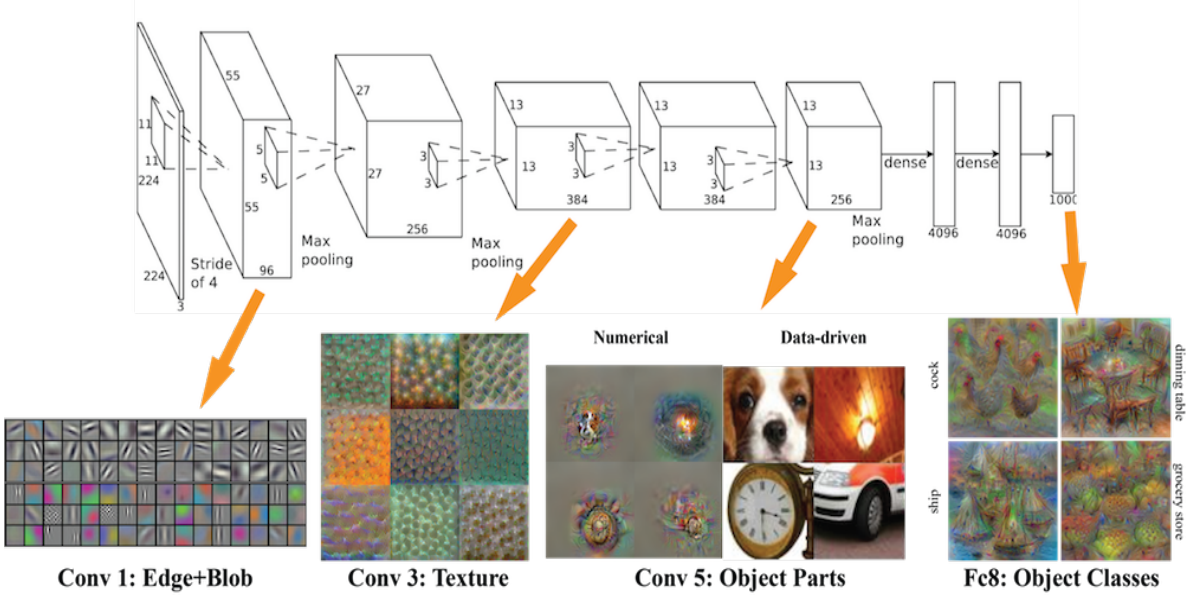
Imagenet, 2012

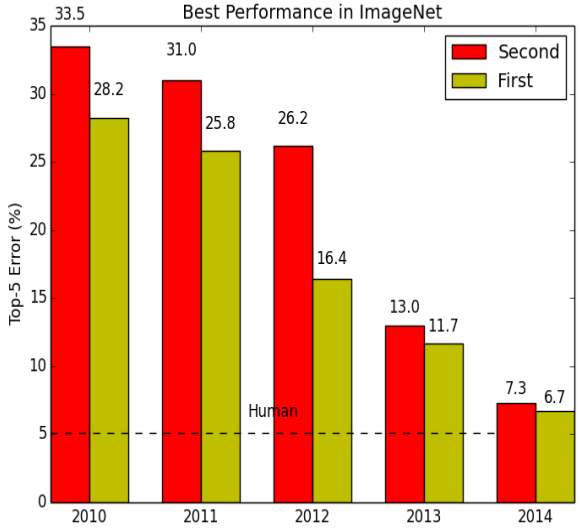




- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)

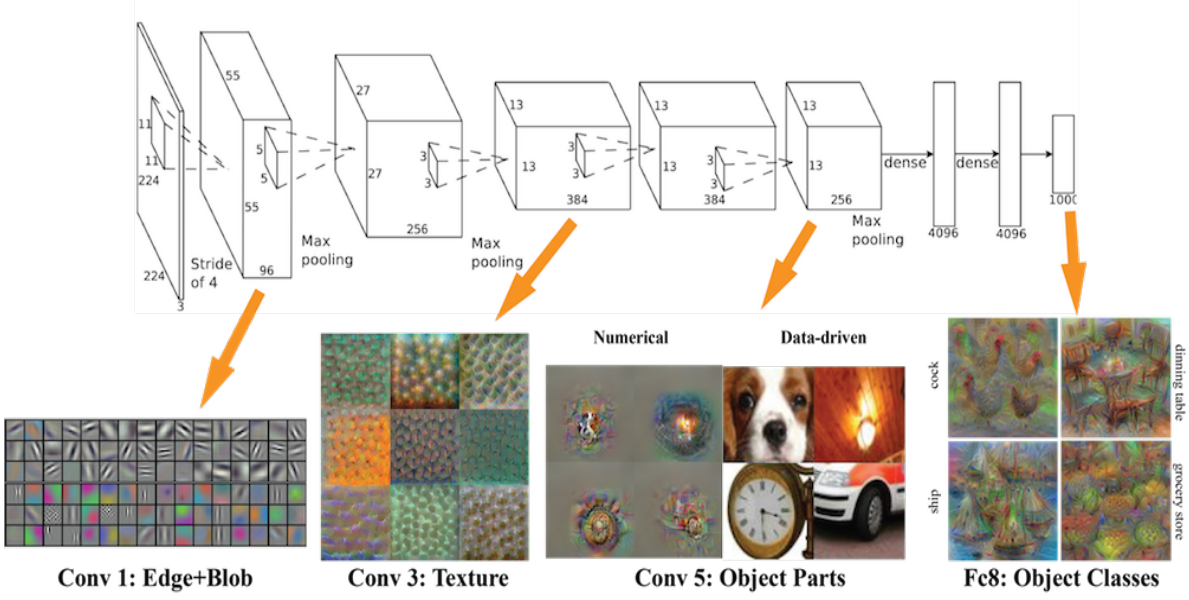
Imagenet, 2012

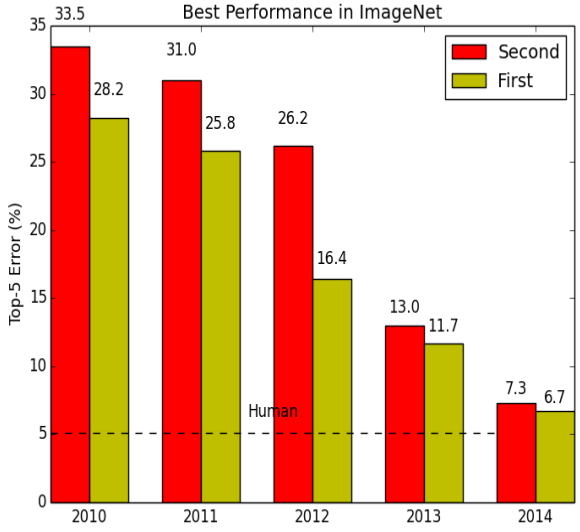




- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)

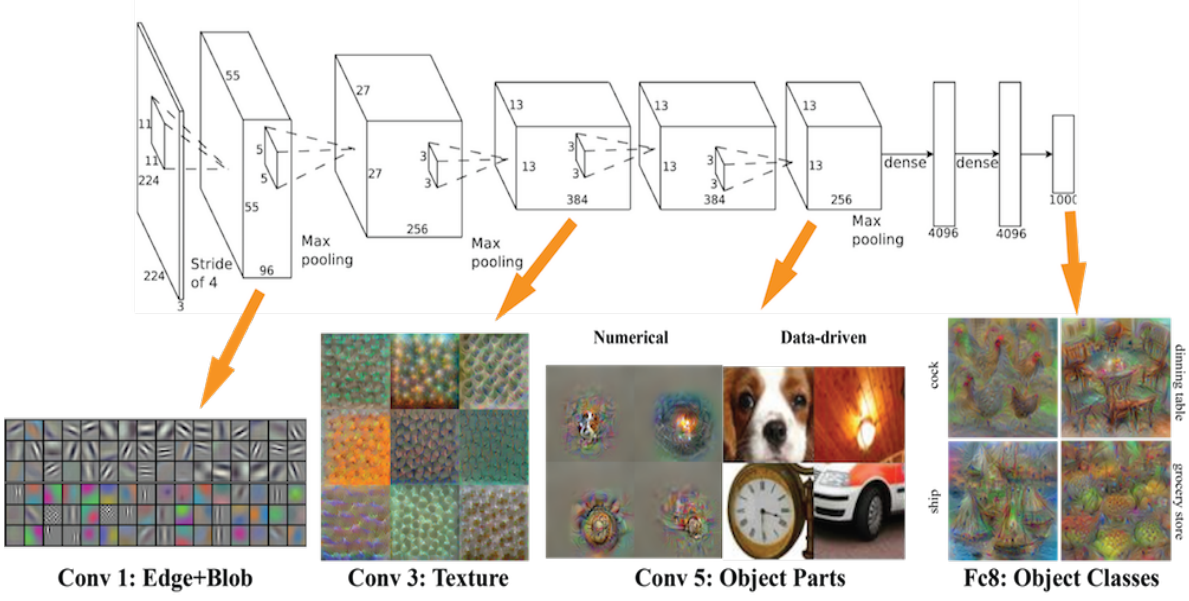
Imagenet, 2012

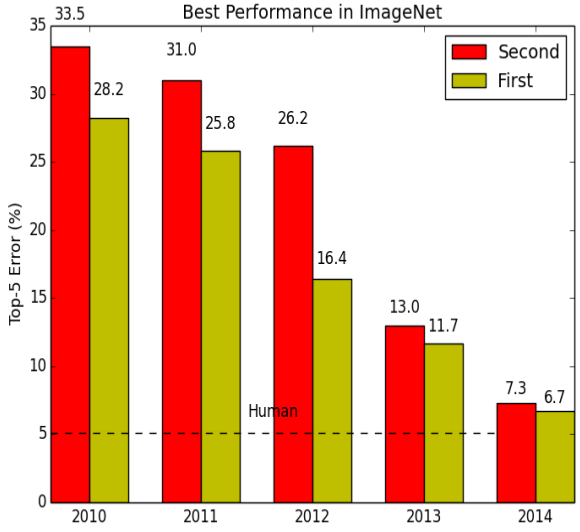




- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)
  - Distributed computation

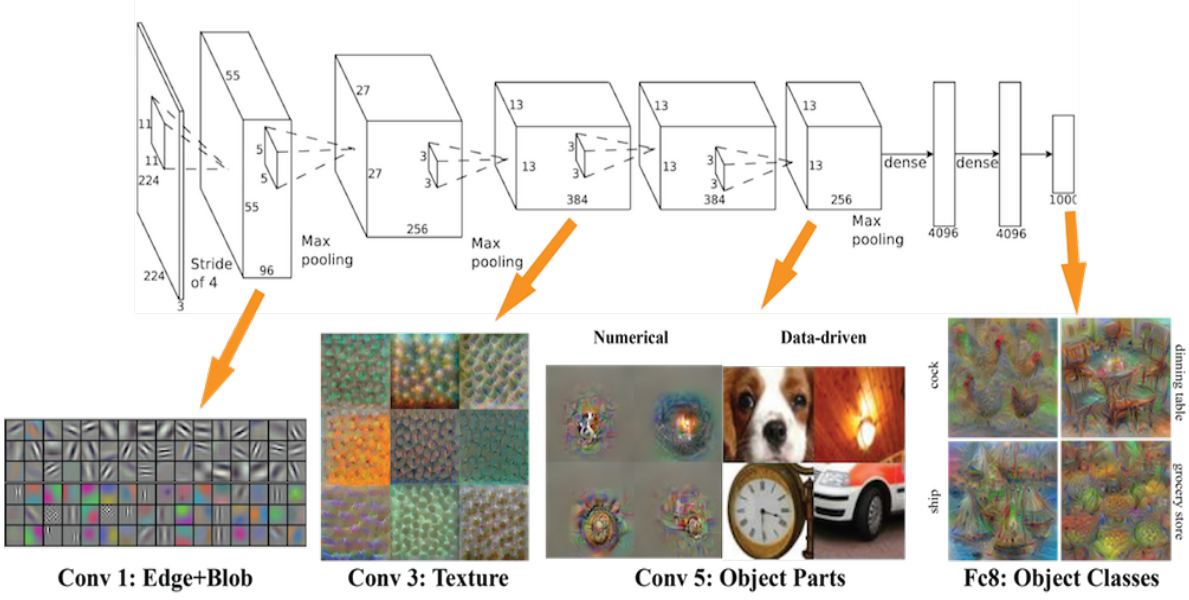
Imagenet, 2012

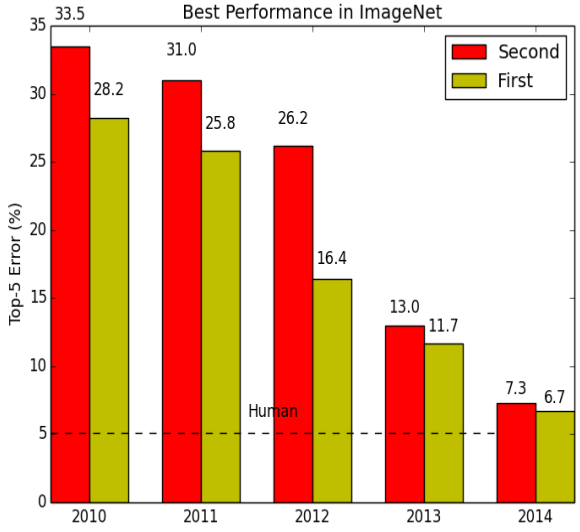




- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)
  - Distributed computation
  - Depth, Invariant feature learning (transferrable to other tasks)

Imagenet, 2012





- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)
  - Distributed computation
  - Depth, Invariant feature learning (transferrable to other tasks)
  - Engineering: Dropout, ReLU ...
- Very active area in Speech and Natural Language Processing.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing
  - Handle non-vectorial data; high-dimensional data
  - Easier model selection via continuous optimization.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing
  - Handle non-vectorial data; high-dimensional data
  - Easier model selection via continuous optimization.
  - Matched NN in many cases, although didnt scale wrt n as well.
- So what changed?

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998 (Deep Blue, Google, Email): Train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing
  - Handle non-vectorial data; high-dimensional data
  - Easier model selection via continuous optimization.
  - Matched NN in many cases, although didnt scale wrt n as well.
- So what changed?
  - More data, parallel algorithms, hardware? Better DNN training? ...

Kernel Methods vs Neural Networks (Pre-Google)

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand
quantitatively why big neural nets working on large databases are not so bad.
(Understanding means that there will be clear conditions and bounds)

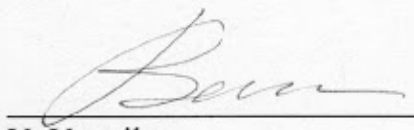
Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

****

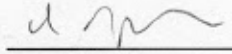
2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong



3/14/95

V. Vapnik



3/14//95

L. Jackel

3/14/95

Witnessed by Y. LeCun

Kernel Methods vs Neural Networks

Geoff Hinton facts meme maintained at
`http://yann.lecun.com/ex/fun/`

Kernel Methods vs Neural Networks

Geoff Hinton facts meme maintained at
`http://yann.lecun.com/ex/fun/`

- All **kernels** that ever dared approaching Geoff Hinton woke up convolved.
- The only **kernel** Geoff Hinton has ever used is a **kernel** of truth.
- If you defy Geoff Hinton, he will maximize your entropy in no time. Your free energy will be gone even before you reach equilibrium.

Kernel Methods vs Neural Networks

Geoff Hinton facts meme maintained at
`http://yann.lecun.com/ex/fun/`

- All **kernels** that ever dared approaching Geoff Hinton woke up convolved.
- The only **kernel** Geoff Hinton has ever used is a **kernel** of truth.
- If you defy Geoff Hinton, he will maximize your entropy in no time. Your free energy will be gone even before you reach equilibrium.

Are there synergies between these fields towards design of even better (faster and more accurate) algorithms?

Linear Hypotheses

- $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$. Assume $\mathcal{Y} \subset \mathbb{R}$ setting.

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{l} (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda\|\mathbf{w}\|_2^2$$

$$\mathbf{w}^\star = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d\right)^{-1}\left(\mathbf{X}^T\mathbf{y}\right)$$

$$\mathbf{X} = \begin{pmatrix} \vdots \\ \mathbf{x}_i^T \\ \vdots \end{pmatrix} \in \mathbb{R}^{l\times n}$$

- $n \times n$ linear system $\implies$ $O(ln^2 + n^3)$ training time assuming no structure (e.g., sparsity).
- $O(n)$ prediction time.
- High Approximation error

Polynomials: The expensive way

- Homogeneous degree-d polynomial

$$f(\mathbf{x}) = \mathbf{w}^T \Psi_{n,d}(\mathbf{x}), \quad \mathbf{w} \in \mathbb{R}^s, \quad s = \binom{d+n-1}{d}$$

$$\Psi_{n,d}(\mathbf{x}) = \begin{pmatrix} \vdots \\ \sqrt{\binom{d}{\alpha}}\mathbf{x}^\alpha \\ \vdots \end{pmatrix} \in \mathbb{R}^s$$

$$\alpha = (\alpha_1 \dots \alpha_n), \sum_i \alpha_i = d, \binom{d}{\alpha} = \frac{d}{\alpha_1! \dots \alpha_n!}$$

$$\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$$

- Construct $\mathbf{Z} \in \mathbb{R}^{n \times s}$ with rows $\Psi_{n,d}(\mathbf{x}_i)$ and solve in $O(s^3)$ (!) time:

$$\mathbf{w}^\star = \left(\mathbf{Z}^T\mathbf{Z} + \lambda \mathbf{I}_d\right)^{-1} \left(\mathbf{Z}^T\mathbf{y}\right) \tag{1}$$

- Note: $n = 100, d = 4 \implies s > 4M$

Polynomials

- Consider the subspace of $\mathbb{R}^s$ spanned by the data,

$$S = \text{span}\left(\Psi(\mathbf{x}_1) \dots \Psi(\mathbf{x}_l)\right) = \{\mathbf{v} \in \mathbb{R}^s : \mathbf{v} = \sum_{i=1}^{l} \alpha_i \Psi(\mathbf{x}_i)\}$$

Polynomials

- Consider the subspace of $\mathbb{R}^s$ spanned by the data,

$$S = \text{span}\left(\Psi(\mathbf{x}_1) \dots \Psi(\mathbf{x}_l)\right) = \{\mathbf{v} \in \mathbb{R}^s : \mathbf{v} = \sum_{i=1}^{l} \alpha_i \Psi(\mathbf{x}_i)\}$$

- So, $\mathbb{R}^s = S + S^{\perp}$. Hence, for any vector $\mathbf{w} \in \mathbb{R}^s$,

$$\mathbf{w} = \mathbf{w}_S + \mathbf{w}_{S^{\perp}}$$

Polynomials

- Consider the subspace of $\mathbb{R}^s$ spanned by the data,

$$S = \operatorname{span}\left(\Psi(\mathbf{x}_1) \dots \Psi(\mathbf{x}_l)\right) = \{\mathbf{v} \in \mathbb{R}^s : \mathbf{v} = \sum_{i=1}^{l} \alpha_i \Psi(\mathbf{x}_i)\}$$

- So, $\mathbb{R}^s = S + S^{\perp}$. Hence, for any vector $\mathbf{w} \in \mathbb{R}^s$,

$$\mathbf{w} = \mathbf{w}_S + \mathbf{w}_{S^{\perp}}$$

- The search of a minimizer can be reduced to S because,

$$\begin{aligned} \sum_{i=1}^{l} V(\mathbf{w}^T \Psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_2^2 &\geq \sum_{i=1}^{l} V(\mathbf{w}_S^T \Psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}_S\|_2^2 + \|\mathbf{w}_S^{\perp}\|_2^2 \\ &\geq \sum_{i=1}^{l} V(\mathbf{w}_S^T \Psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}_S\|_2^2 \qquad (2) \end{aligned}$$

Polynomials

- Consider the subspace of $\mathbb{R}^s$ spanned by the data,

$$S = \text{span}\left(\Psi(\mathbf{x}_1) \dots \Psi(\mathbf{x}_l)\right) = \{\mathbf{v} \in \mathbb{R}^s : \mathbf{v} = \sum_{i=1}^{l} \alpha_i \Psi(\mathbf{x}_i)\}$$

- So, $\mathbb{R}^s = S + S^{\perp}$. Hence, for any vector $\mathbf{w} \in \mathbb{R}^s$,

$$\mathbf{w} = \mathbf{w}_S + \mathbf{w}_{S^{\perp}}$$

- The search of a minimizer can be reduced to S because,

$$\begin{aligned} \sum_{i=1}^{l} V(\mathbf{w}^T \Psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_2^2 &\geq \sum_{i=1}^{l} V(\mathbf{w}_S^T \Psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}_S\|_2^2 + \|\mathbf{w}_S^{\perp}\|_2^2 \\ &\geq \sum_{i=1}^{l} V(\mathbf{w}_S^T \Psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}_S\|_2^2 \qquad (2) \end{aligned}$$

- Argument holds for any loss (convex or non-convex, additive or not), but needs orthogonality (l_2 regularizer) [**Representer Theorem**]

Polynomials

- Hence, $\mathbf{w}^\star = \sum_{i=1}^{l} \beta_i \Psi(\mathbf{x}_i) \in S$ for $\beta \in \mathbb{R}^l$, and so we can solve:

$$\underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min} \qquad \sum_{i=1}^{l} (y_i - \mathbf{w}^T \Psi(\mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|_2^2$$

Polynomials

- Hence, $\mathbf{w}^\star = \sum_{i=1}^{l} \beta_i \Psi(\mathbf{x}_i) \in S$ for $\beta \in \mathbb{R}^l$, and so we can solve:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \quad \sum_{i=1}^{l} (y_i - \mathbf{w}^T \Psi(\mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|_2^2$$

$$\arg\min_{\beta\in\mathbb{R}^l} \quad \sum_{j=1}^{l} \left(y_j - \sum_{i=1}^{l} \beta_i \Psi(\mathbf{x}_i)^T \Psi(\mathbf{x}_j) \right)^2 + \lambda \sum_{i,j=1}^{l} \beta_i \beta_j \Psi(\mathbf{x}_i)^T \Psi(\mathbf{x}_j)$$

Polynomials

- Hence, $\mathbf{w}^\star = \sum_{i=1}^{l} \beta_i \Psi(\mathbf{x}_i) \in S$ for $\beta \in \mathbb{R}^l$, and so we can solve:

$$\begin{aligned}
\arg\min_{\mathbf{w}\in\mathbb{R}^d} \quad & \sum_{i=1}^{l} (y_i - \mathbf{w}^T \Psi(\mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|_2^2 \\
\arg\min_{\beta\in\mathbb{R}^l} \quad & \sum_{j=1}^{l} \left(y_j - \sum_{i=1}^{l} \beta_i \Psi(\mathbf{x}_i)^T \Psi(\mathbf{x}_j) \right)^2 + \lambda \sum_{i,j=1}^{l} \beta_i \beta_j \Psi(\mathbf{x}_i)^T \Psi(\mathbf{x}_j) \\
\arg\min_{\beta\in\mathbb{R}^l} \quad & \sum_{j=1}^{l} \left(y_j - (\mathbf{G}\beta)_j \right)^2 + \lambda \beta^T \mathbf{G} \beta \\
\mathbf{G}_{ij} &= \Psi(\mathbf{x}_i)^T \Psi(\mathbf{x}_j) \\
\beta^\star &= (\mathbf{G} + \lambda \mathbf{I}_d)^{-1} \mathbf{y}
\end{aligned}$$

- $O(l^3 + sl^2)$ training time - $O(s^3)$ cost eliminated.
- Inference time ($O(s)$, or $O(ls)$):

$$f(\mathbf{x}) = \mathbf{w}^T \Psi(\mathbf{x}) = \sum_{i=1}^{l} \beta_i \Psi(\mathbf{x}_i)^T \Psi(\mathbf{x})$$

Polynomials

- Multinomial Theorem

$$(z_1 + z_2 + \ldots + z_n)^d = \sum_{\alpha:|\alpha|=d} \binom{d}{\alpha} \mathbf{z}^\alpha$$

- Implicit computation of inner products

$$\Psi(\mathbf{x})^T \Psi(\mathbf{x}') = \sum_{i=1}^{s} \binom{d}{\alpha} \mathbf{x}^\alpha \mathbf{x}'^\alpha = \sum_{i=1}^{s} \binom{d}{\alpha} x_1^{\alpha_1} \ldots x_n^{\alpha_n} x_1'^{\alpha_1} \ldots x_n'^{\alpha_n}$$

Polynomials

- Multinomial Theorem

$$(z_1 + z_2 + \ldots + z_n)^d = \sum_{\alpha:|\alpha|=d} \binom{d}{\alpha} \mathbf{z}^\alpha$$

- Implicit computation of inner products

$$\begin{aligned}
\Psi(\mathbf{x})^T \Psi(\mathbf{x}') &= \sum_{i=1}^{s} \binom{d}{\alpha} \mathbf{x}^\alpha \mathbf{x}'^\alpha = \sum_{i=1}^{s} \binom{d}{\alpha} x_1^{\alpha_1} \ldots x_n^{\alpha_n} x_1'^{\alpha_1} \ldots x_n'^{\alpha_n} \\
&= \sum_{i=1}^{s} \binom{d}{\alpha} (x_1 x_1')^{\alpha_1} (x_2 x_2')^{\alpha_2} \ldots (x_n x_n')^{\alpha_n}
\end{aligned}$$

Polynomials

- Multinomial Theorem

$$(z_1 + z_2 + \ldots + z_n)^d = \sum_{\alpha:|\alpha|=d} \binom{d}{\alpha} \mathbf{z}^\alpha$$

- Implicit computation of inner products

$$\begin{aligned}
\Psi(\mathbf{x})^T \Psi(\mathbf{x}') &= \sum_{i=1}^{s} \binom{d}{\alpha} \mathbf{x}^\alpha \mathbf{x}'^\alpha = \sum_{i=1}^{s} \binom{d}{\alpha} x_1^{\alpha_1} \ldots x_n^{\alpha_n} x_1'^{\alpha_1} \ldots x_n'^{\alpha_n} \\
&= \sum_{i=1}^{s} \binom{d}{\alpha} (x_1 x_1')^{\alpha_1} (x_2 x_2')^{\alpha_2} \ldots (x_n x_n')^{\alpha_n} \\
&= (x_1 x_1' + \ldots x_n x_n')^d
\end{aligned}$$

Polynomials

- Multinomial Theorem

$$(z_1 + z_2 + \ldots + z_n)^d = \sum_{\alpha:|\alpha|=d} \binom{d}{\alpha} \mathbf{z}^\alpha$$

- Implicit computation of inner products

$$\begin{aligned}
\Psi(\mathbf{x})^T \Psi(\mathbf{x}') &= \sum_{i=1}^{s} \binom{d}{\alpha} \mathbf{x}^\alpha \mathbf{x}'^\alpha = \sum_{i=1}^{s} \binom{d}{\alpha} x_1^{\alpha_1} \ldots x_n^{\alpha_n} x_1'^{\alpha_1} \ldots x_n'^{\alpha_n} \\
&= \sum_{i=1}^{s} \binom{d}{\alpha} (x_1 x_1')^{\alpha_1} (x_2 x_2')^{\alpha_2} \ldots (x_n x_n')^{\alpha_n} \\
&= (x_1 x_1' + \ldots x_n x_n')^d \\
&= (\mathbf{x}^T \mathbf{x}')^d
\end{aligned}$$

Polynomials

- Multinomial Theorem

$$(z_1 + z_2 + \ldots + z_n)^d = \sum_{\alpha:|\alpha|=d} \binom{d}{\alpha} \mathbf{z}^\alpha$$

- Implicit computation of inner products

$$\begin{aligned}
\Psi(\mathbf{x})^T \Psi(\mathbf{x}') &= \sum_{i=1}^{s} \binom{d}{\alpha} \mathbf{x}^\alpha \mathbf{x}'^\alpha = \sum_{i=1}^{s} \binom{d}{\alpha} x_1^{\alpha_1} \ldots x_n^{\alpha_n} x_1'^{\alpha_1} \ldots x_n'^{\alpha_n} \\
&= \sum_{i=1}^{s} \binom{d}{\alpha} (x_1 x_1')^{\alpha_1} (x_2 x_2')^{\alpha_2} \ldots (x_n x_n')^{\alpha_n} \\
&= (x_1 x_1' + \ldots x_n x_n')^d \\
&= (\mathbf{x}^T \mathbf{x}')^d
\end{aligned}$$

- $O(l^3 + l^2 n)$ training and $O(ln)$ predicting speed.
- Complexity coming from s has been completely eliminated (!). [**Kernel Trick**]

Polynomials: Algorithm

- Algorithm
  - Start with $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$
  - Construct Gram matrix: $\mathbf{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ on the training samples.
  - Solve: $\beta^\star = (\mathbf{G} + \lambda \mathbf{I}_d)^{-1} \mathbf{y}$
  - Return $f^\star(\mathbf{x}) = \sum_{i=1}^{l} \beta_i k(\mathbf{x}_i, \mathbf{x})$
- $f^\star$ is the optimal degree-d polynomial solving the learning problem, in complexity independent of d.

Polynomials: Algorithm

- Algorithm
  - Start with $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$
  - Construct Gram matrix: $\mathbf{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ on the training samples.
  - Solve: $\beta^\star = (\mathbf{G} + \lambda \mathbf{I}_d)^{-1} \mathbf{y}$
  - Return $f^\star(\mathbf{x}) = \sum_{i=1}^{l} \beta_i k(\mathbf{x}_i, \mathbf{x})$
- $f^\star$ is the optimal degree-d polynomial solving the learning problem, in complexity independent of d.
- What other forms of k correspond to linear learning in high-dimensional nonlinear embeddings of the data?

Symmetric, positive semi-definite functions

- **Definition:** A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is p.s.d if for any finite collection of points $\mathbf{x}_1 \dots \mathbf{x}_l$, the $l \times l$ Gram matrix

$$\mathbf{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

is positive semi-definite, i.e. for any vector $\beta \in \mathbb{R}^l$

$$\beta^T \mathbf{G} \beta = \sum_{ij} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

Symmetric, positive semi-definite functions

- **Definition:** A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is p.s.d if for any finite collection of points $\mathbf{x}_1 \dots \mathbf{x}_l$, the $l \times l$ Gram matrix

$$\mathbf{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

  is positive semi-definite, i.e. for any vector $\beta \in \mathbb{R}^l$

$$\beta^T \mathbf{G} \beta = \sum_{ij} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

- **Theorem**[Mercer]: If k is symmetric, p.s.d, $\mathcal{X}$ is compact subset of $\mathbb{R}^n$, then it admits an eigenfunction decomposition:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \sum_{i=1}^{N} \lambda_i \phi_j(\mathbf{x}) \phi_j(\mathbf{x}') = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}') \rangle_{l_2} \\ \Psi(\mathbf{x}) &= [\dots, \sqrt{\lambda_j} \phi(\mathbf{x}) \dots]^T \end{aligned} \tag{3}$$

- Feature map associated with a kernel is not unique.
- Functional generalization of positive semi-definite matrices.

Kernels

- Linear

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$$

- Gaussian: $s = \infty$, Universal

$$k(\mathbf{x}, \mathbf{x}') = e^{\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$$

Kernels

- Linear

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$$

- Gaussian: $s = \infty$, Universal

$$k(\mathbf{x}, \mathbf{x}') = e^{\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$$

- Conic combinations:

$$k(\mathbf{x}, \mathbf{x}') = \alpha_1 k_1(\mathbf{x}, \mathbf{x}') + \alpha_2 k_2(\mathbf{x}, \mathbf{x}')$$

Kernels

- Linear
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$
- Polynomial
$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$$
- Gaussian: $s = \infty$, Universal
$$k(\mathbf{x}, \mathbf{x}') = e^{\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$$
- Conic combinations:
$$k(\mathbf{x}, \mathbf{x}') = \alpha_1 k_1(\mathbf{x}, \mathbf{x}') + \alpha_2 k_2(\mathbf{x}, \mathbf{x}')$$
- Elementwise products:
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$$
- Kernels on discrete sets: strings, graphs, sequences, shapes

Gaussian Kernel is Positive Definite

- Exponential

$$e^{\beta \mathbf{x}^T \mathbf{x}'} = 1 + \beta \mathbf{x}^T \mathbf{x}' + \frac{\beta^2}{2!}(\mathbf{x}^T \mathbf{y})^2 + \frac{\beta^3}{3!}(\mathbf{x}^T \mathbf{y})^3 + \ldots$$

Gaussian Kernel is Positive Definite

- Exponential

$$e^{\beta \mathbf{x}^T \mathbf{x}'} = 1 + \beta \mathbf{x}^T \mathbf{x}' + \frac{\beta^2}{2!}(\mathbf{x}^T \mathbf{y})^2 + \frac{\beta^3}{3!}(\mathbf{x}^T \mathbf{y})^3 + \dots$$

- For any function $f : \mathbb{R}^n \mapsto \mathbb{R}$, p.s.d function k, the following kernel is p.s.d.

$$k'(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) \left(k(\mathbf{x}, \mathbf{x}')\right) f(\mathbf{x}')$$

Proof:

$$\sum_{ij} \beta_i \beta_j k'(\mathbf{x}_i, \mathbf{x}_j) = \sum_{ij} \beta_i \beta_j f(\mathbf{x}_i) k(\mathbf{x}_i, \mathbf{x}_j) f(\mathbf{x}_j) = \sum_{ij} \beta_i' \beta_j' k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

Gaussian Kernel is Positive Definite

- Exponential

$$e^{\beta \mathbf{x}^T \mathbf{x}'} = 1 + \beta \mathbf{x}^T \mathbf{x}' + \frac{\beta^2}{2!}(\mathbf{x}^T \mathbf{y})^2 + \frac{\beta^3}{3!}(\mathbf{x}^T \mathbf{y})^3 + \ldots$$

- For any function $f : \mathbb{R}^n \mapsto \mathbb{R}$, p.s.d function k, the following kernel is p.s.d.

$$k'(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})\left(k(\mathbf{x}, \mathbf{x}')\right) f(\mathbf{x}')$$

  Proof:

$$\sum_{ij} \beta_i \beta_j k'(\mathbf{x}_i, \mathbf{x}_j) = \sum_{ij} \beta_i \beta_j f(\mathbf{x}_i) k(\mathbf{x}_i, \mathbf{x}_j) f(\mathbf{x}_j) = \sum_{ij} \beta_i' \beta_j' k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

- Gaussian Kernel:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\sigma^2}} \\ &= e^{-\frac{\|\mathbf{x}\|_2^2}{2\sigma^2}} \left(e^{\frac{2\mathbf{x}^T \mathbf{x}'}{2\sigma^2}}\right) e^{-\frac{\|\mathbf{x}'\|_2^2}{2\sigma^2}} \end{aligned} \tag{4}$$

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)
- **Theorem** *All* nice Hilbert spaces are generated by a symmetric positive definite function (the kernel) $k(\mathbf{x}, \mathbf{x}')$ on $\mathcal{X} \times \mathcal{X}$
- if $f, g \in \mathcal{H}$ close i.e. $\|f - g\|_{\mathcal{H}}$ small, then $f(\mathbf{x}), g(\mathbf{x})$ close $\forall \mathbf{x} \in \mathcal{X}$.

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)
- **Theorem** *All* nice Hilbert spaces are generated by a symmetric positive definite function (the kernel) $k(\mathbf{x}, \mathbf{x}')$ on $\mathcal{X} \times \mathcal{X}$
- if $f, g \in \mathcal{H}$ close i.e. $\|f - g\|_{\mathcal{H}}$ small, then $f(\mathbf{x}), g(\mathbf{x})$ close $\forall \mathbf{x} \in \mathcal{X}$.
- **Reproducing Kernel Hilbert Spaces** (RKHSs): Hilbert space $\mathcal{H}$ that admit a *reproducing kernel*, i.e., a function $k(\mathbf{x}, \mathbf{x}')$ satisfying:

$$(i) \quad \forall \mathbf{x} \in \mathcal{X} : k(\cdot, \mathbf{x}) \in \mathcal{H}$$
$$(ii) \quad \langle f, k(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = f(\mathbf{x})$$

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)
- **Theorem** *All* nice Hilbert spaces are generated by a symmetric positive definite function (the kernel) $k(\mathbf{x}, \mathbf{x}')$ on $\mathcal{X} \times \mathcal{X}$
- if $f, g \in \mathcal{H}$ close i.e. $\|f - g\|_{\mathcal{H}}$ small, then $f(\mathbf{x}), g(\mathbf{x})$ close $\forall \mathbf{x} \in \mathcal{X}$.
- **Reproducing Kernel Hilbert Spaces** (RKHSs): Hilbert space $\mathcal{H}$ that admit a *reproducing kernel*, i.e., a function $k(\mathbf{x}, \mathbf{x}')$ satisfying:

$$(i) \quad \forall \mathbf{x} \in \mathcal{X} : k(\cdot, \mathbf{x}) \in \mathcal{H}$$
$$(ii) \quad \langle f, k(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = f(\mathbf{x})$$

- Reproducing kernels are symmetric, p.s.d:

$$\begin{aligned} \sum_{i,j} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j) &= \langle \sum_i \beta_i k(\cdot, \mathbf{x}_i), \sum_i \beta_i k(\cdot, \mathbf{x}_i) \rangle_{\mathcal{H}} \\ &= \| \sum_i \beta_i k(\cdot, \mathbf{x}_i) \|^2_{\mathcal{H}} \geq 0 \end{aligned} \tag{5}$$

- **(Moore-Aronsjan, 1950)** Symmetric, p.s.d functions are reproducing kernels for some (unique) RKHS $\mathcal{H}_k$.

Kernel Methods: Summary

- Symm. pos. def. function $k(\mathbf{x}, \mathbf{z})$ on input domain $\mathcal{X} \subset \mathbb{R}^d$
- $k \Leftrightarrow$ rich Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ of real-valued functions, with inner product $\langle \cdot, \cdot \rangle_k$ and norm $\| \cdot \|_k$
- Regularized Risk Minimization $\Leftrightarrow$ Linear models in an *implicit* high-dimensional (often infinite-dimensional) feature space.

$$f^\star = \arg\min_{f \in \mathcal{H}_k} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_k}, \ \mathbf{x}_i \in \mathbb{R}^d$$

- **Representer Theorem**: $f^\star(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$

Revisiting the Issue of Scalability

- Regularized Least Squares

$$(\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} = \mathbf{Y} \qquad \begin{array}{cc} O(l^2) & \text{storage} \\ O(l^3 + l^2 n) & \text{training} \\ O(ln) & \text{test speed} \end{array}$$

  Hard to parallelize when working directly with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

- Linear kernels: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}, \quad f^\star(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$, $(\mathbf{w} = \mathbf{X}^T \boldsymbol{\alpha})$

$$\left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}\right) \mathbf{w} = \mathbf{X}^T \mathbf{Y} \quad \begin{array}{cc} O(ln) & \text{storage} \\ O(ln^2) & \text{training} \\ O(n) & \text{test speed} \end{array}$$

Randomized Algorithms: Definitions

- Exact feature maps are extremely high-dimensional, hard to compute.
- Approximate feature maps: $\Psi : \mathbb{R}^d \mapsto \mathbb{C}^s$ such that,

$$k(\mathbf{x}, \mathbf{z}) \approx \langle \hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{z}) \rangle_{\mathbb{C}^s}$$

$$\Rightarrow \left(\mathbf{Z}(\mathbf{X})^T \mathbf{Z}(\mathbf{X}) + \lambda \mathbf{I} \right) \mathbf{w} = \mathbf{Z}(\mathbf{X})^T \mathbf{Y} \Rightarrow \begin{matrix} O(ls) & \text{storage} \\ O(ls^2) & \text{training} \\ O(s) & \text{test speed} \end{matrix}$$

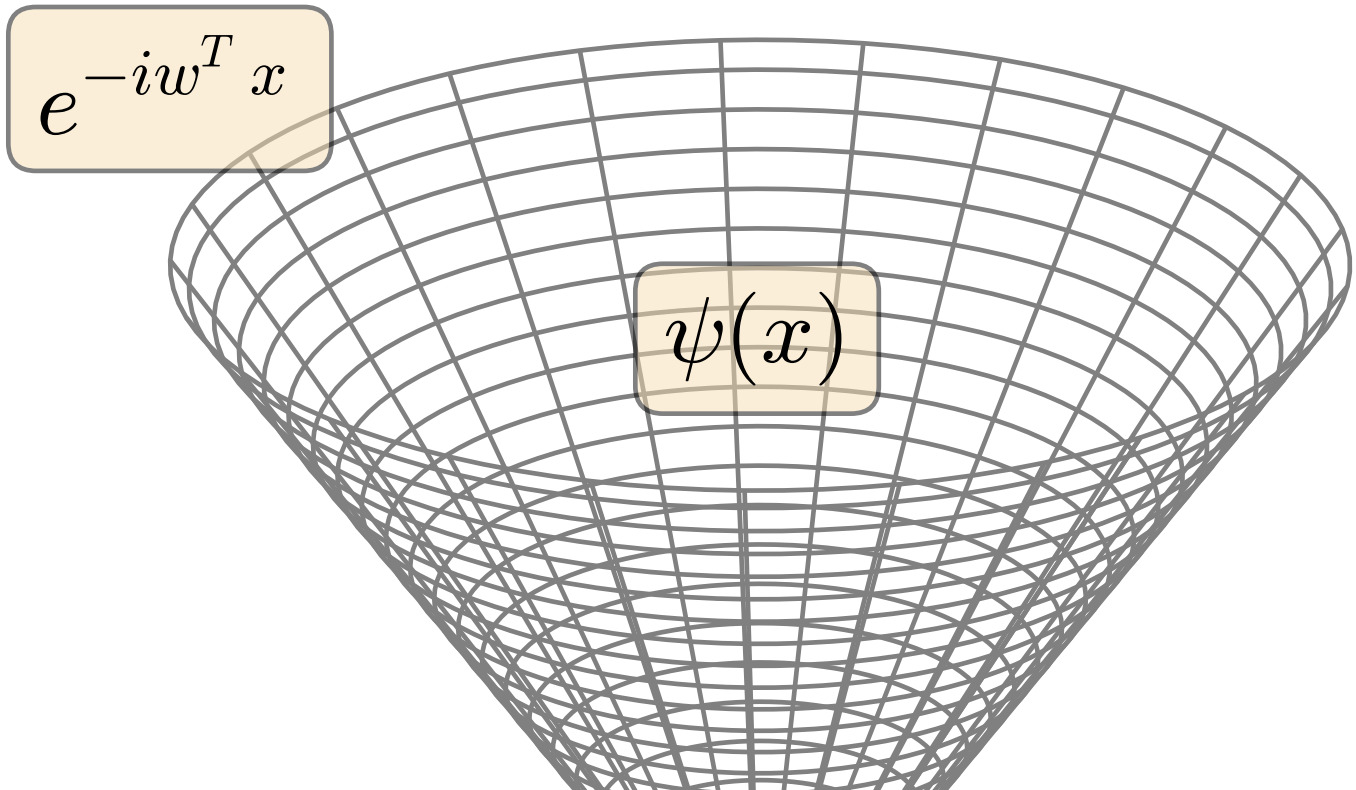
- Shift-Invariant kernels: $k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x} - \mathbf{z})$, for some complex-valued *positive definite function* ψ on $\mathbb{R}^d$
  - Given any set of m points, $\mathbf{z}_1 \ldots \mathbf{z}_m \in \mathbb{R}^d$, the $m \times m$ the matrix $\mathbf{A}_{ij} = \psi(\mathbf{z}_i - \mathbf{z}_j)$ is positive definite.

Randomized Algorithms: Bochner's Theorem

Theorem (Bochner, 1932-33)

A complex-valued function $\psi : \mathbb{R}^d \mapsto \mathbb{C}$ is positive definite if and only if it is the Fourier Transform of a finite non-negative measure μ on $\mathbb{R}^d$, i.e.,

$$\psi(\mathbf{x}) = \hat{\mu}(\mathbf{x}) = \int_{\mathbb{R}^d} e^{-i\mathbf{x}^T\mathbf{w}} d\mu(\mathbf{w}), \qquad \mathbf{x} \in \mathbb{R}^d \ .$$



Random Fourier Features (Rahimi & Recht, 2007)

- One-to-one correspondence between k and density p such that,

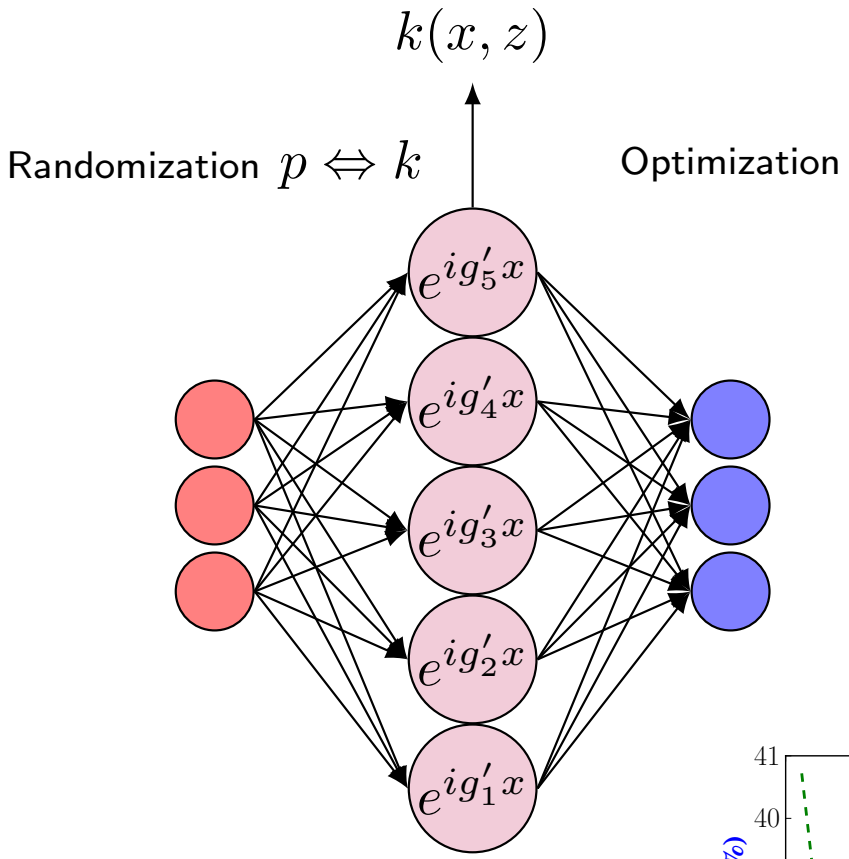
$$k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x} - \mathbf{z}) = \int_{\mathbb{R}^d} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}} p(\mathbf{w}) d\mathbf{w}$$

  Gaussian kernel: $k(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2}} \iff p = \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$
- Monte-Carlo approximation to Integral representation:

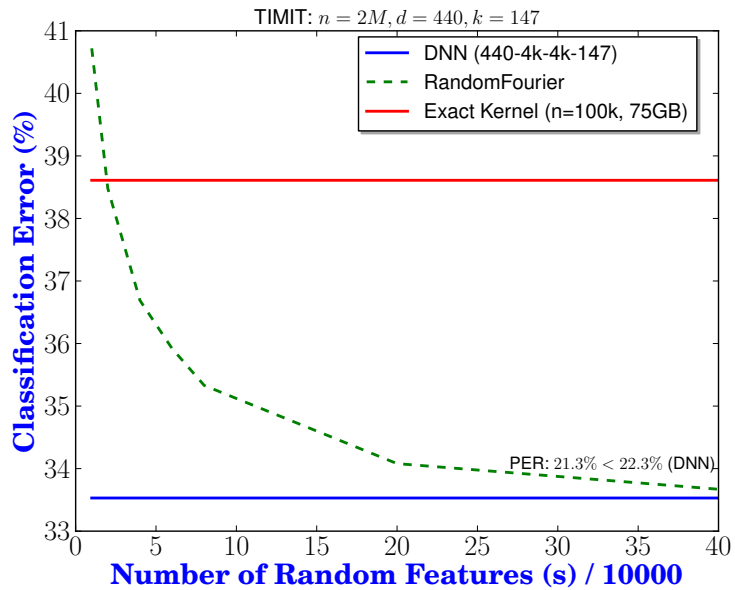
$$k(\mathbf{x}, \mathbf{z}) \approx \frac{1}{s} \sum_{j=1}^{s} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}_j} = \langle \hat{\Psi}_S(\mathbf{x}), \hat{\Psi}_S(\mathbf{z}) \rangle_{\mathbb{C}^s}$$

$$\hat{\Psi}_S(\mathbf{x}) = \frac{1}{\sqrt{s}} \left[e^{-i\mathbf{x}^T \mathbf{w}_1} \dots e^{-i\mathbf{x}^T \mathbf{w}_s} \right] \in \mathbb{C}^s, \quad S = [\mathbf{w}_1 \dots \mathbf{w}_s] \sim p$$

Randomization-vs-Optimization

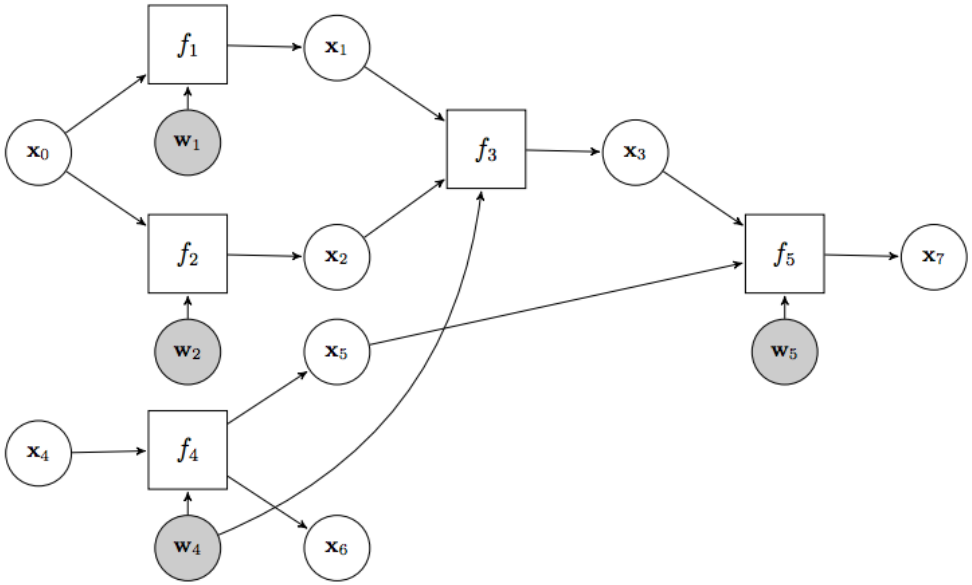


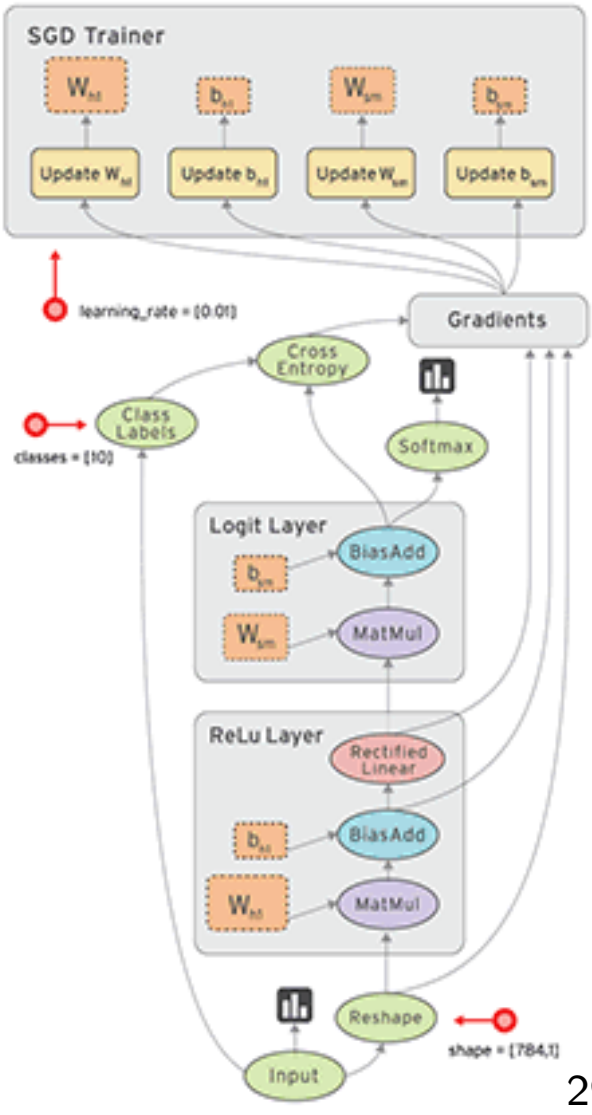

```
G = randn(size(X,1), s);
Z = exp(i*X*G);
I = eye(size(X,2));
C = Z'*Z;
alpha = (C + lambda*I)\(Z'*y(:));
ztest = exp(i*xtest*G)*alpha;
```



Computational Graphs and TensorFlow

tensorflow.org/





- Jacobian of $f : \mathbb{R}^n \mapsto \mathbb{R}^m$:

$$[\frac{df}{dx}]_{ij} = \frac{\partial f_i}{\partial x_j} \in \mathbb{R}^{m \times n}$$

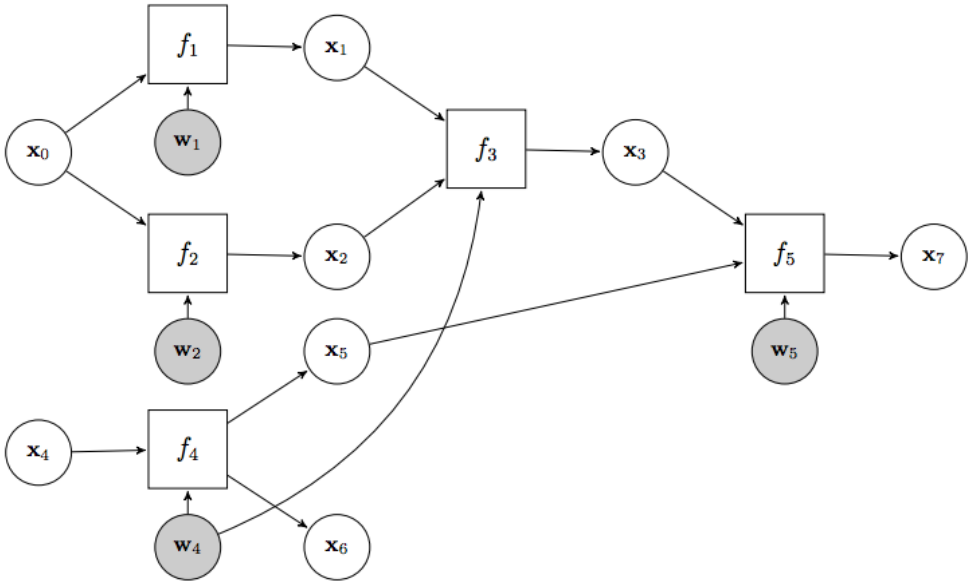
- SGD with minibatches

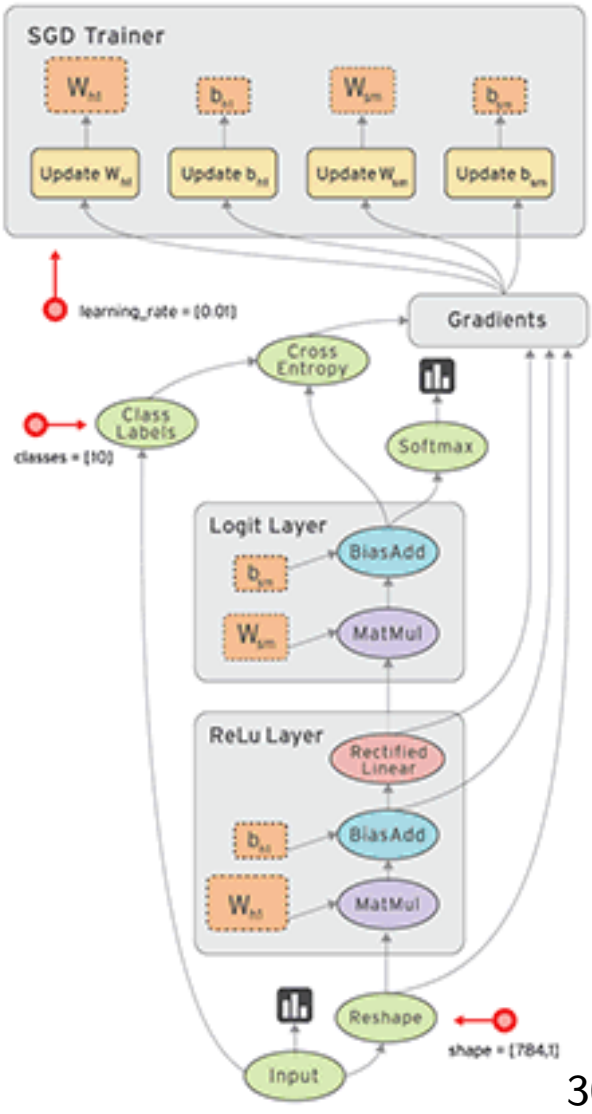
$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \sum_{i=1}^{l} \frac{\partial l_i}{\partial \mathbf{w}}\Big|_{\mathbf{w_t}}$$

.

Computational Graphs and TensorFlow

tensorflow.org/





- Automatic Differentiation: Forward mode

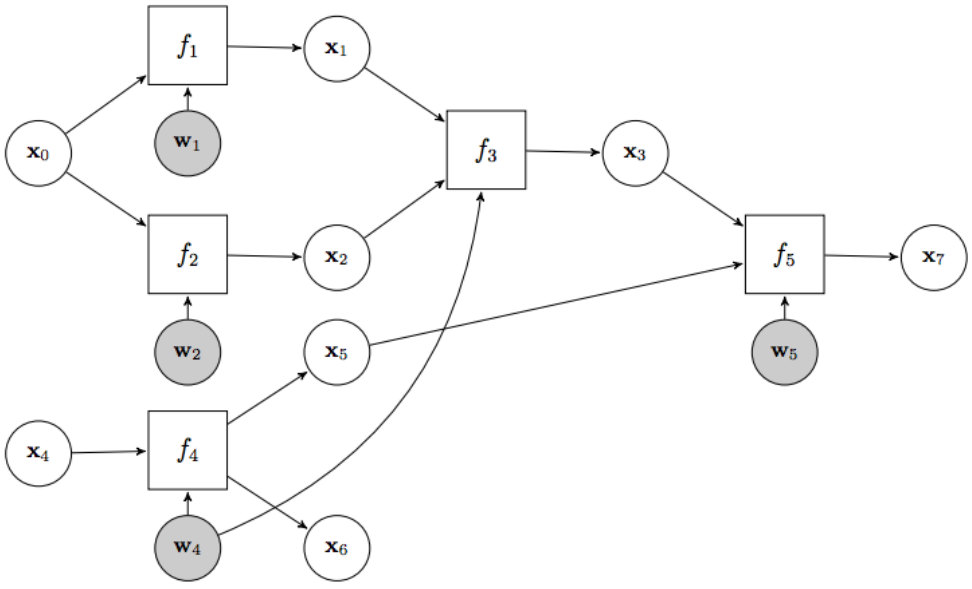
$$\frac{\partial \mathbf{x}_i}{\partial \mathbf{w}} = \sum_{j \to i} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_j} \frac{\partial \mathbf{x}_j}{\partial \mathbf{w}}$$

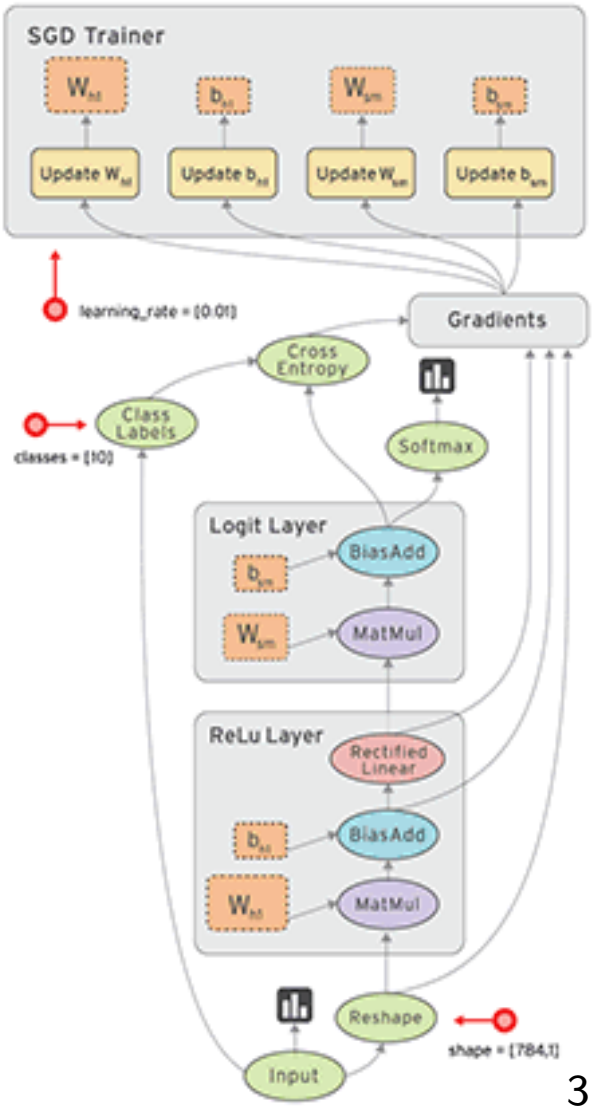
- Automatic Differentiation: Reverse mode

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_i} = \sum_{i \to j} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_j} \frac{\partial \mathbf{x}_j}{\partial \mathbf{x}_i}$$

Computational Graphs and TensorFlow

tensorflow.org/



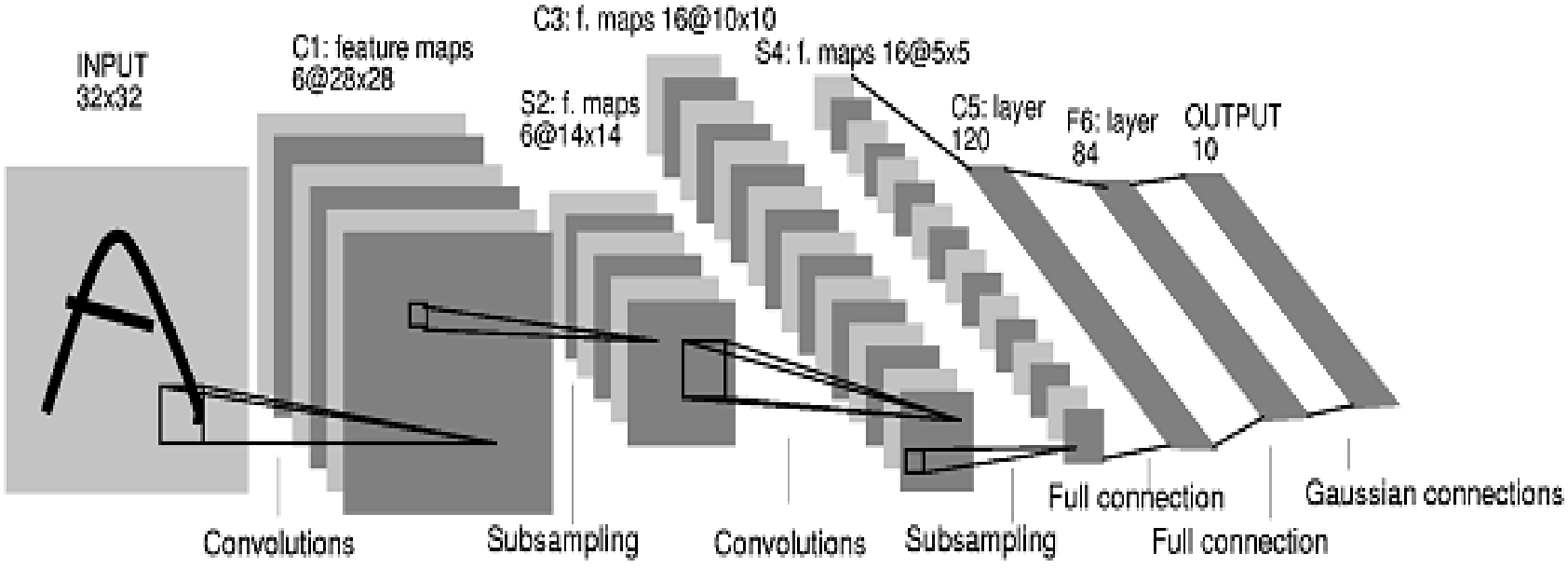


- Tensor-Ops (Python/C++)

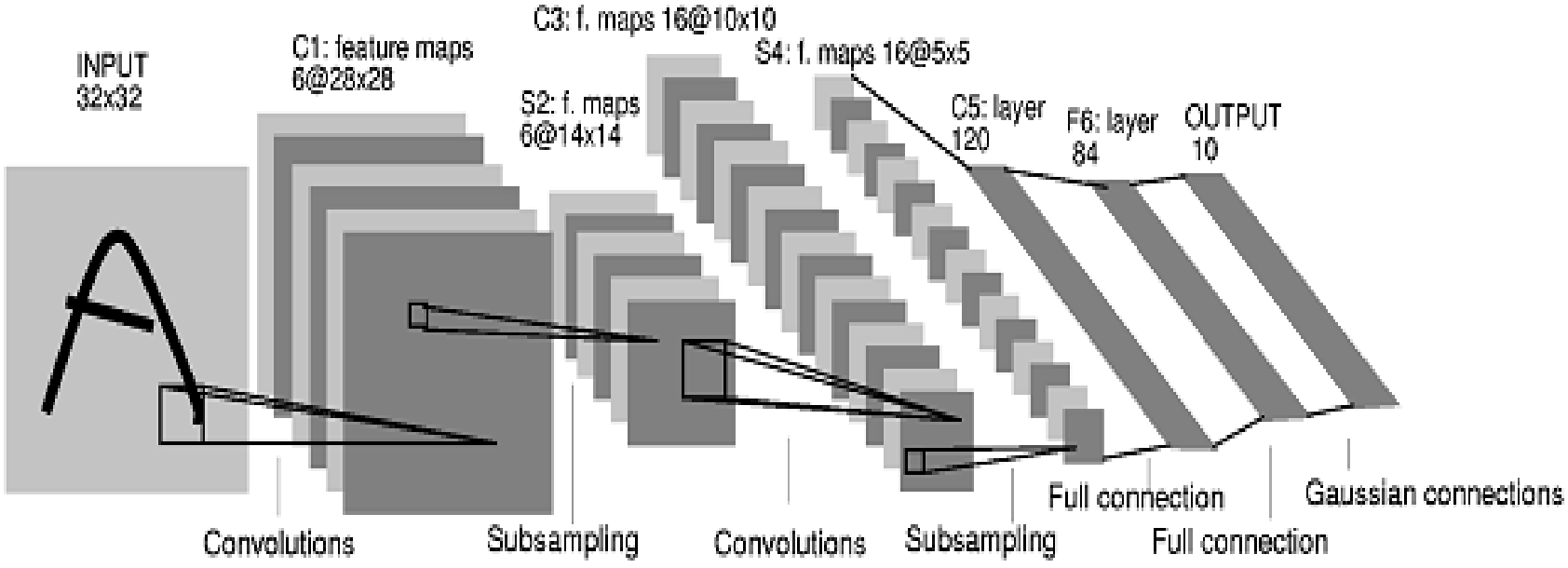
$$f(\mathbf{x}, \mathbf{w}), \left[\frac{\partial f}{\partial \mathbf{x}}\right]^T \star \mathbf{v}, \left[\frac{\partial f}{\partial \mathbf{w}}\right]^T \star \mathbf{v}$$

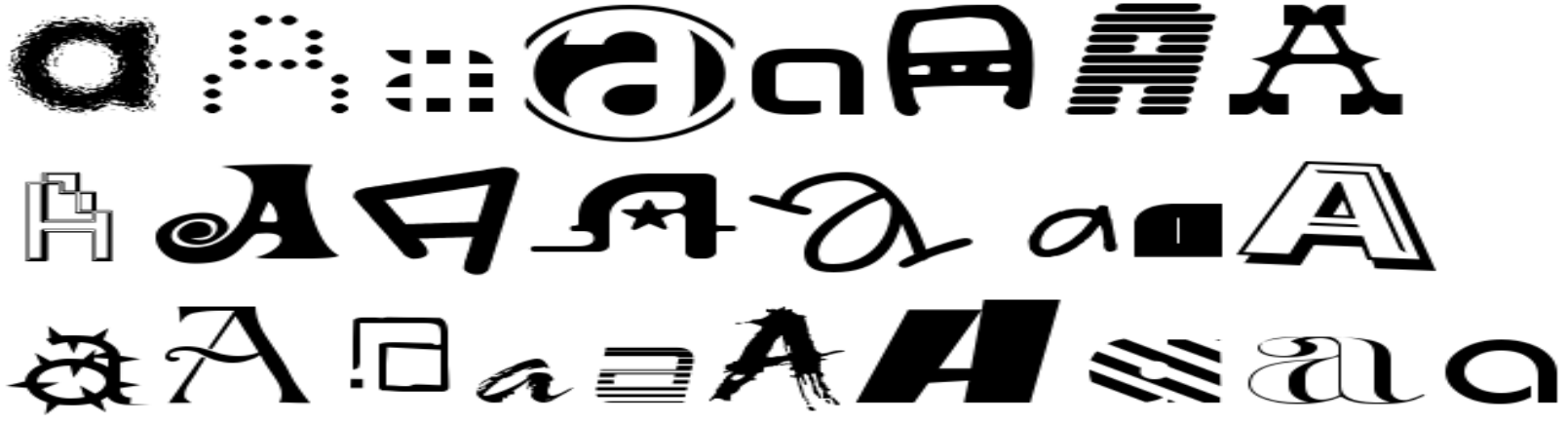
- Model Parallelism: Assign independent paths to threads, GPUs.
- Data Parallelism: create Graph replicas on different machines.

Convolutional Neural Networks

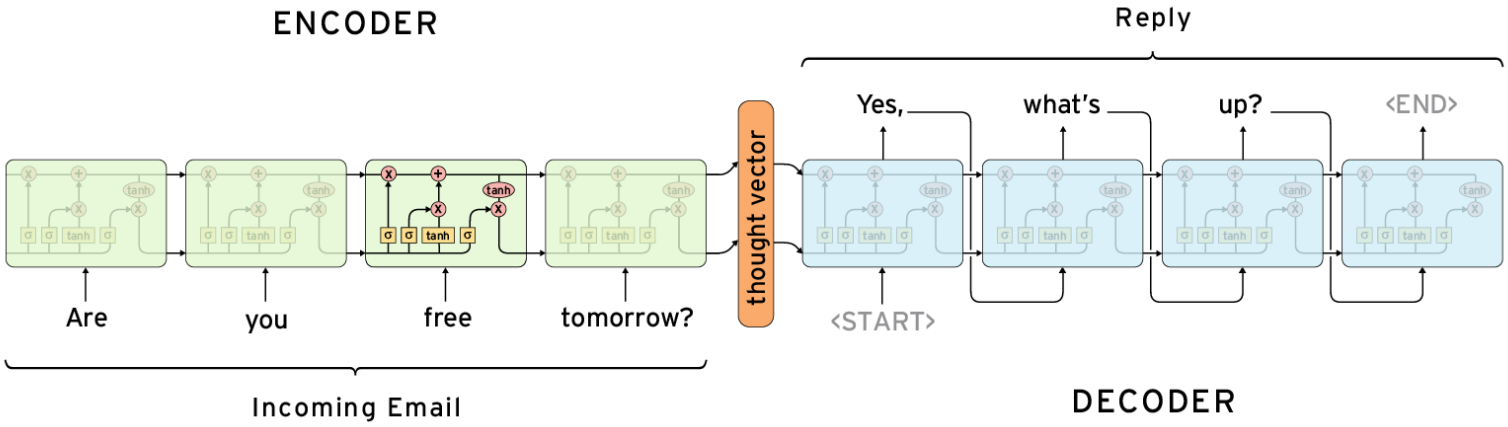


Convolutional Neural Networks



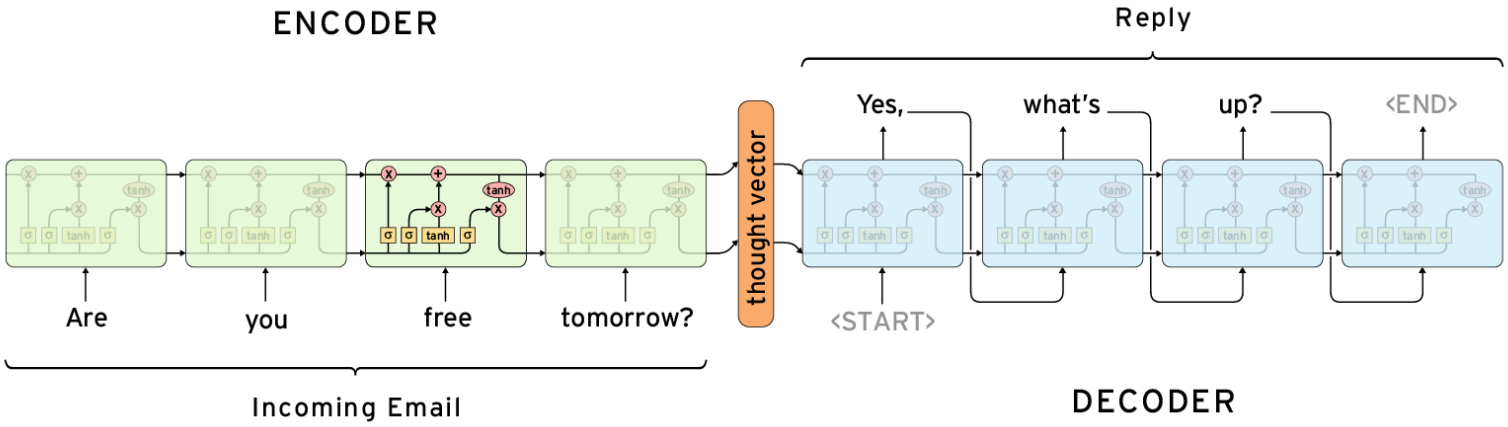


Recurrent NN: SmartReply, Translation, Captioning



- Training data: $\{\left(\mathbf{x}_1^{(i)} \dots \mathbf{x}_p^{(i)}\right), \left(\mathbf{y}_1^{(i)} \dots \mathbf{y}_p^{(i)}\right)\}_{i=1}^{l}$, $\mathbf{x}_t^i, \mathbf{y}_t^i \in \mathbb{R}^V$

Recurrent NN: SmartReply, Translation, Captioning



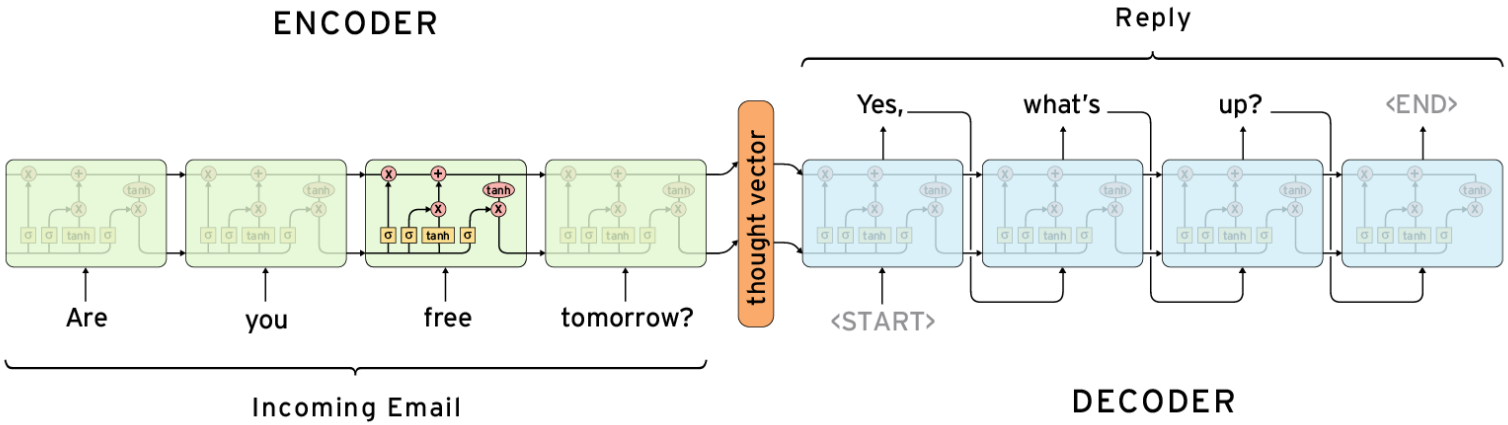
- Training data: $\{\left(\mathbf{x}_1^{(i)} \dots \mathbf{x}_p^{(i)}\right), \left(\mathbf{y}_1^{(i)} \dots \mathbf{y}_p^{(i)}\right)\}_{i=1}^{l}$, $\mathbf{x}_t^i, \mathbf{y}_t^i \in \mathbb{R}^V$
- Parameterized Nonlinear dynamical system:

$$\mathbf{s}_t = f(\mathbf{s}_t, \mathbf{x}_t) = \sigma(\mathbf{A}\mathbf{s}_{t-1} + \mathbf{B}\mathbf{x}_t) \quad (6)$$

$$\mathbf{o}_t = g(\mathbf{s}_t) = \mathbf{W}\mathbf{s}_t \quad (7)$$

- Inference time: sampling and beam search

Recurrent NN: SmartReply, Translation, Captioning



- Training data: $\{\left(\mathbf{x}_1^{(i)} \dots \mathbf{x}_p^{(i)}\right), \left(\mathbf{y}_1^{(i)} \dots \mathbf{y}_p^{(i)}\right)\}_{i=1}^{l}$, $\mathbf{x}_t^i, \mathbf{y}_t^i \in \mathbb{R}^V$
- Parameterized Nonlinear dynamical system:

$$\mathbf{s}_t = f(\mathbf{s}_t, \mathbf{x}_t) = \sigma(\mathbf{A}\mathbf{s}_{t-1} + \mathbf{B}\mathbf{x}_t) \quad (6)$$

$$\mathbf{o}_t = g(\mathbf{s}_t) = \mathbf{W}\mathbf{s}_t \quad (7)$$

- Inference time: sampling and beam search
- Training

$$\arg\min_{\mathbf{A},\mathbf{B},\mathbf{W}} \sum_{i=1}^{l} \sum_{t=1}^{T} V(\mathbf{y}_t^{(i)}, \mathbf{o}_t^{(i)})$$

- End-to-End English-French translation using LSTMs (tensorflow.org)

Challenges and Opportunities

- Architectures
- Nonconvex Optimization: beyond SGD, escaping saddle points, vanishing-exploding gradients.
- Can you win Imagenet with Convex optimization?
- Supervised to Semi-supervised to Unsupervised Learning.
- Compact, efficient, real-time models for new form factors.
- Exciting, emerging world of Robotics, Wearable Computers, Intelligent home devices, Personal digital assistants!

Google Internship Program.