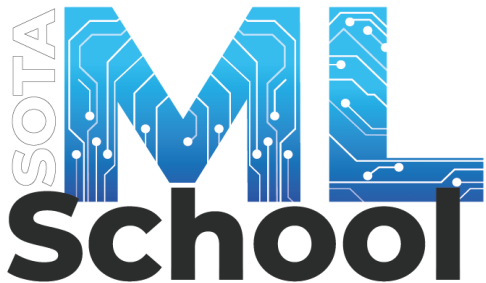
ITMO.Hack

ITMO ML LAB

BOTAN INVESTMENTS



Deep Learning Theory Part II

Semyon Polyakov

Outline

Bias and Variance tradeoff

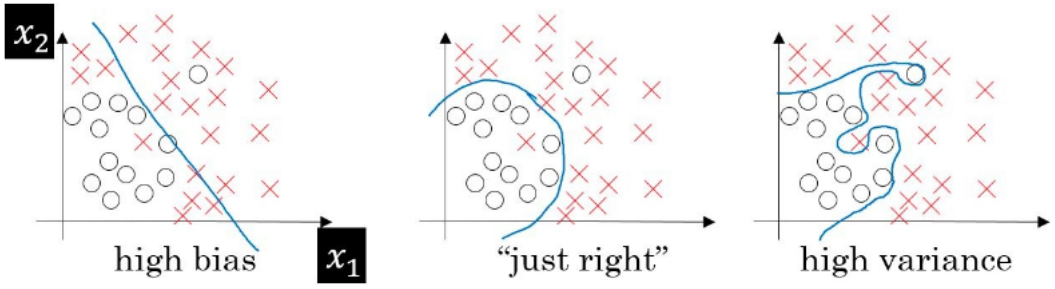


Figure: Bias and variance tradeoff corner cases illustration

Bias and Variance tradeoff

$$E[(y - \tilde{f}(x))^2] = (Bias[\tilde{f}(x)])^2 + Var[\tilde{f}(x)] + \sigma^2$$

$$Bias[\tilde{f}(x)] = E[\tilde{f}(x) - f(x)]$$

$$Var[\tilde{f}(x)] = E[(\tilde{f}(x))^2] - E[(\tilde{f}(x)]^2]$$

Intuition

The bias error is an error from erroneous assumptions in the learning algorithm (underfitting)

The variance is an error from sensitivity to small fluctuations in the training set (overfitting)

Regularization

$$E_D(w) + \lambda E_W(w)$$

$$E_W = \sum_{j=1}^{M} |w_j|^q$$

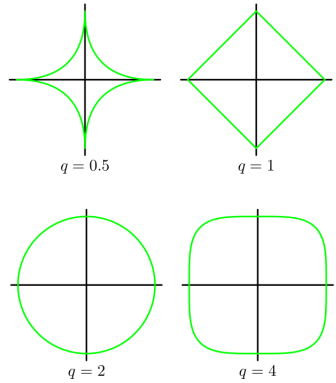


Figure: Polinomial regularizations - L1, L2, L3, L4

Regularization

$$E_D(w) + \lambda E_W(w)$$

$$E_W = \sum_{j=1}^{M} |w_j|^q$$

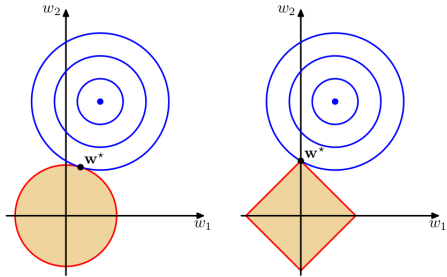


Figure: L1 tends to zeroing least valuable weights

Regularization

Other ways for regularization:

- Dropout (optimal is near 0.5)
- BatchNormalization (+layer, instance and group normalization)

BatchNorm limitations

- Works bad for small batches
- Too computationaly expensive for recurrent networks

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure: BatchNorm algorithm

Regularization

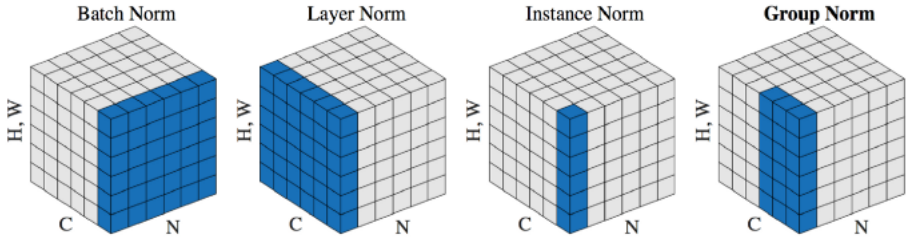


Figure: Different types of normalization in NN

Optimizers, warmup and LR scheduling

Optimizers

Adam

- One of the best optimizers, but often loses to SGD with momentum
- Has 3 hyperparameters
- There are plenty of improved Adam versions (NAdam, RAdam, AdaMod, Lamb, LARS, ..)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Figure: Adam optimizer

Optimizers

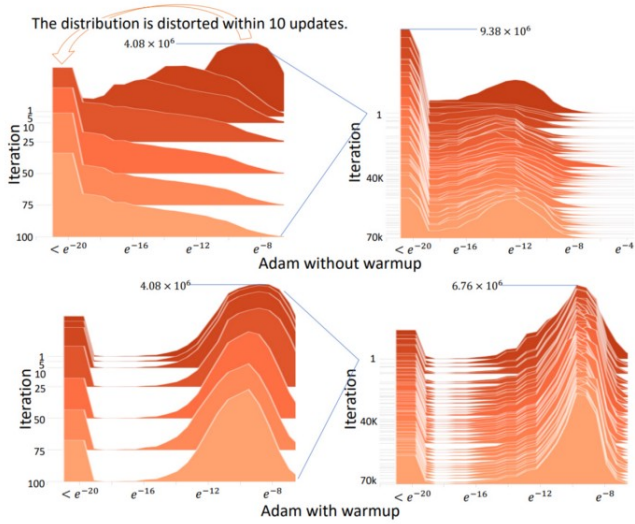


Figure: Adam warmup influence

Optimizers

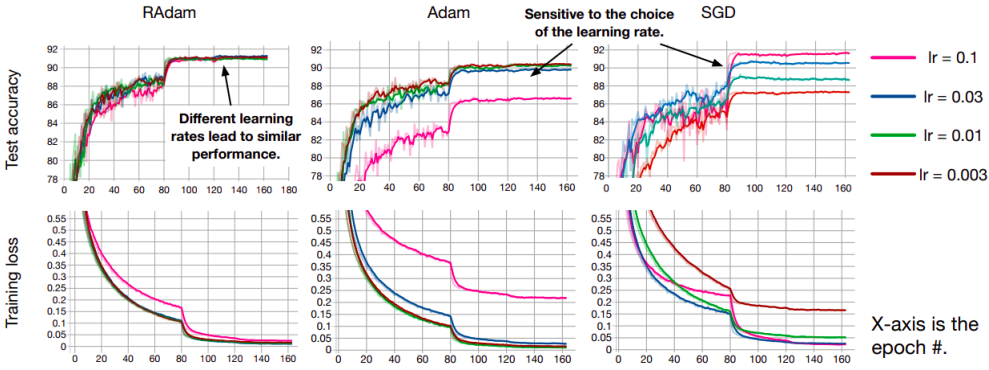


Figure: RAdam performance for CIFAR-10 dataset

Optimizers

Algorithm 2: Rectified Adam. All operations are element-wise.

Input: $\{\alpha_t\}_{t=1}^{T}$: step size, $\{\beta_1, \beta_2\}$: decay rate to calculate moving average and moving 2nd moment, θ_0: initial parameter, $f_t(\theta)$: stochastic objective function.

Output: θ_t: resulting parameters

$m_0, v_0 \leftarrow 0, 0$ (Initialize moving 1st and 2nd moment)

$\rho_\infty \leftarrow 2/(1-\beta_2) - 1$ (Compute the maximum length of the approximated SMA)

while $t = \{1, \cdots, T\}$ **do**

  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Calculate gradients w.r.t. stochastic objective at timestep t)

  $v_t \leftarrow 1/\beta_2 v_{t-1} + (1-\beta_2) g_t^2$ (Update exponential moving 2nd moment)

  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ (Update exponential moving 1st moment)

  $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$ (Compute bias-corrected moving average)

  $\rho_t \leftarrow \rho_\infty - 2t\beta_2^t/(1-\beta_2^t)$(Compute the length of the approximated SMA)

  if *the variance is tractable, i.e.,* $\rho_t > 4$ **then**

    $l_t \leftarrow \sqrt{(1-\beta_2^t)/v_t}$ (Compute adaptive learning rate)

    $r_t \leftarrow \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}}$ (Compute the variance rectification term)

    $\theta_t \leftarrow \theta_{t-1} - \alpha_t r_t \widehat{m_t} l_t$ (Update parameters with adaptive momentum)

  else

    $\theta_t \leftarrow \theta_{t-1} - \alpha_t \widehat{m_t}$ (Update parameters with un-adapted momentum)

return θ_T

Optimizers

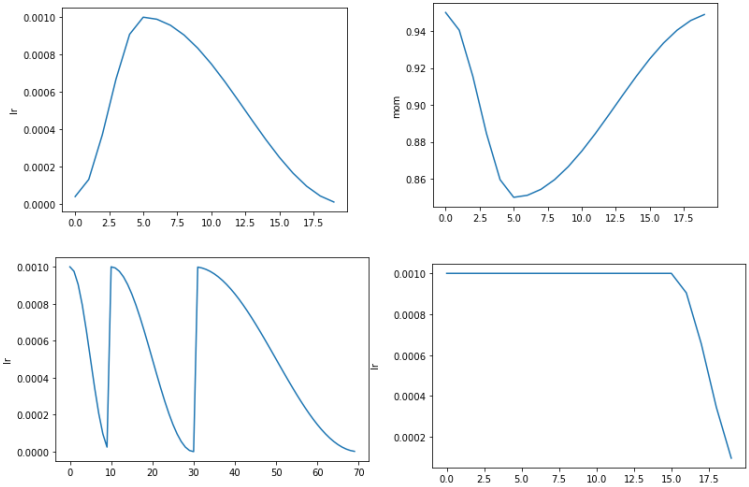


Figure: Learning rate scheduling examples from fastai

Optimizers

Notes:

- No free lunch
- Combinations of ideas can help (Nesterov momentum, expnential weighting, ..)
- Layer-wise learning rate wins global learning rate
- Slow, but effective: SGD with momentum + warmup/LR scheduling + tuning

Multi-tasking

Transfer learning, Multi-tasking, one-shot learning

Preliminaries

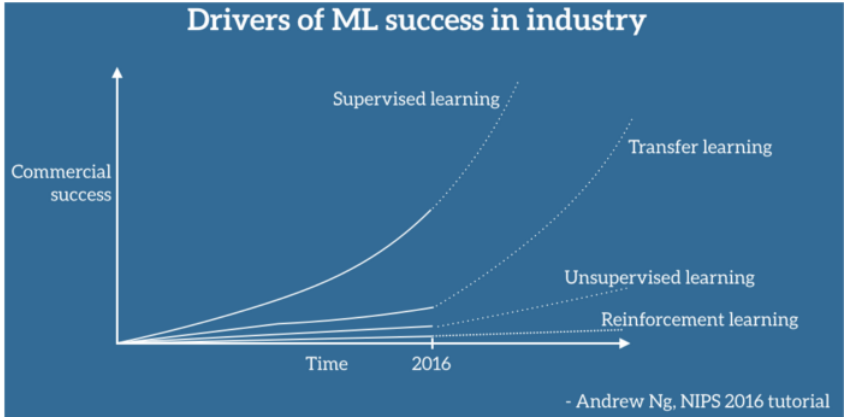


Figure: ML areas success plot by Andrew NG

Transfer learning

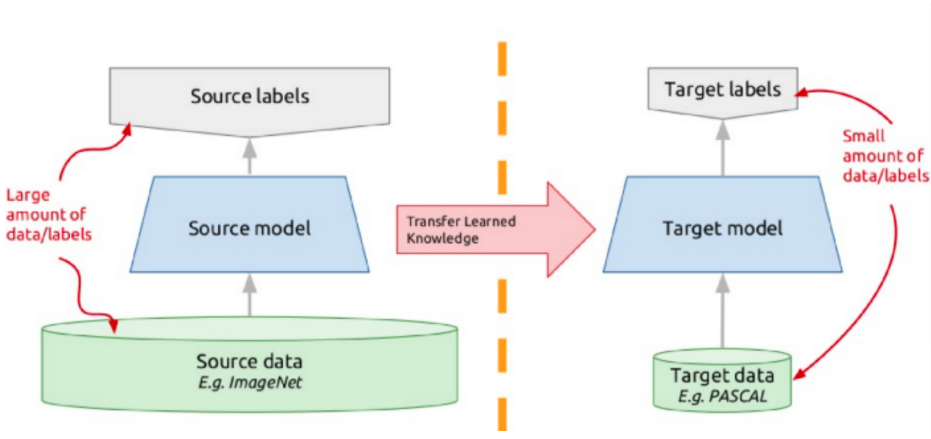


Figure: Transfer learning principal scheme

Multi-tasking

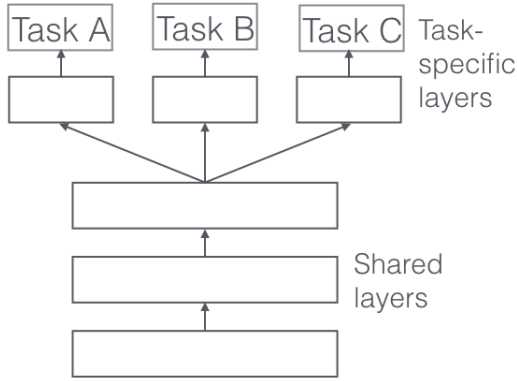


Figure: Shared layers for multi-task learning

Multi-tasking

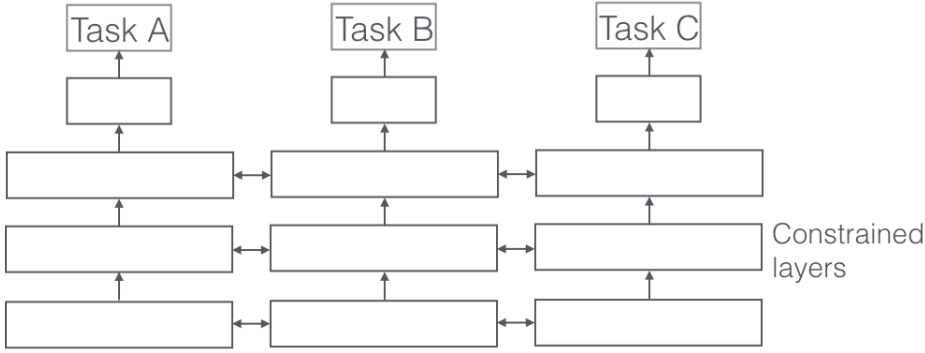


Figure: Weakly shared layers for multi-task learning

Transfer learning

Notes:

- One-shot learning = Metric learning + transfer learning
- Augmentation helps even with 1 example
- Zero-shot learning = Metric learning + (external data)?
- Center loss for better regularization

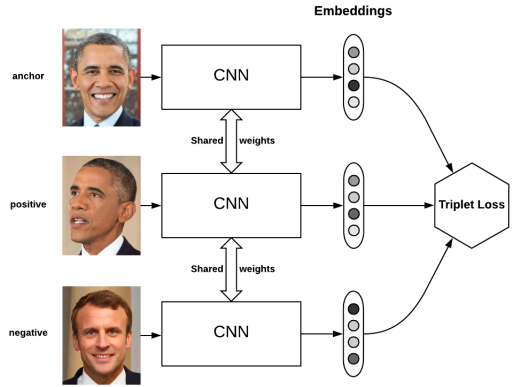


Figure: Triplet loss example

Compression

Neural networks compression and speed up

Compression and speed up

Notes:

- Quantization (16bit, 8bit, ..)
- Factorization
- Pruning (weights, neurons, filters)
- Knowledge distillation

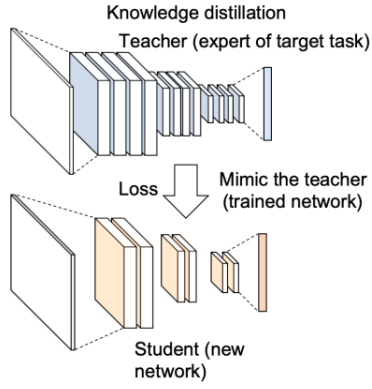


Figure: Triplet loss example

Recent neural networks research

Adversarial attacks

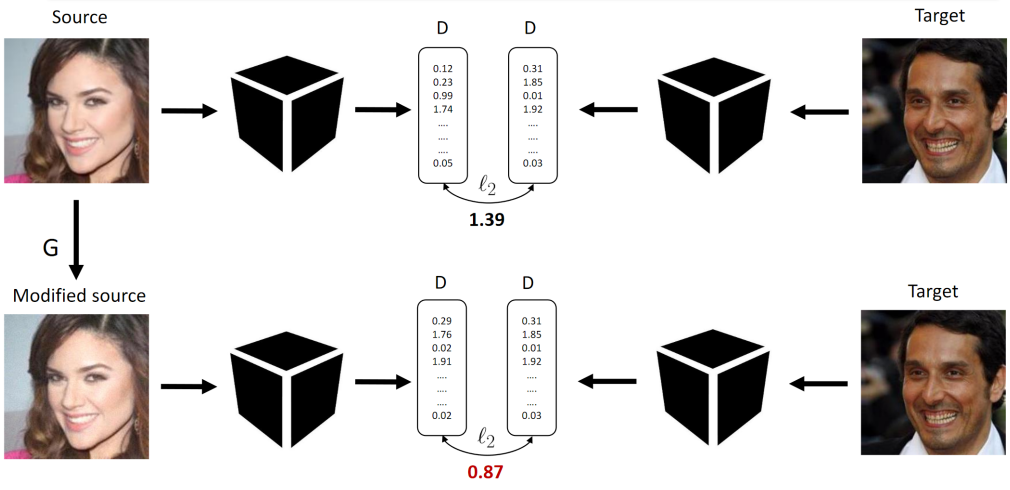


Figure: Adversarial attack example

Neural tangent kernels

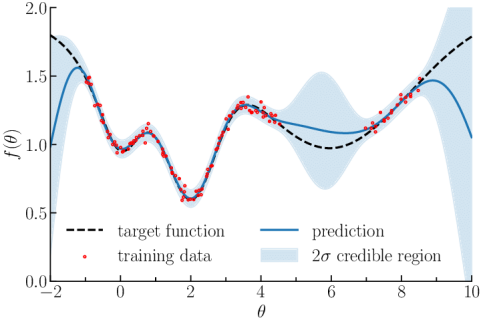


Figure: Gaussian process

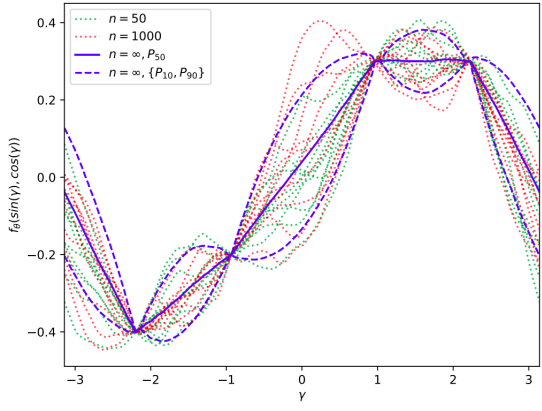


Figure: Neural network asymptotically approximates GP

Flows

$$x \sim P(X)$$
$$z \sim p_\theta(z)$$
$$x = g_\theta(z)$$
$$z = f_\theta(x) = g^{-1}{}_\theta(x)$$
$$f = f_1 \circ f_2 \circ \ldots \circ f_k$$

Figure: Flow-based networks

Figure: Glow: Flow-based face generation network results

Final

During the research, not a single neural network was harmed

Thank you for attention!

SOTA ML School

State-Of-The-Art Machine Learning School