Machine Learning - Modern Times
Large-scale learning at Google

Corinna Cortes
Google Research
corinna@google.com
Outline

- Learning kernels
- Learning similarities
The user is burdened with choosing an appropriate kernel.
Regression Example
Regression Example

Fit with a second degree polynomial?
Regression Example

Fit with a third degree polynomial?
Regression Example

Fit with a linear function?
Demands less commitment from the user: instead of a specific kernel, only requires the definition of a family of kernels.
Kernel Families

- Most frequently used family, linear combinations:

\[ \{ K_\mu = \sum_{k=1}^{p} \mu_k K_k : \mu \in \mathcal{M} \} \]

- Lanckriet et al. (2002)

\[
\min_{\mu} \max_{\alpha} \quad F(\mu, \alpha) = 2\alpha^\top 1 - \alpha^\top Y^\top \left( \sum_{k=1}^{p} \mu_k K_k \right) Y \alpha
\]

subject to \( 0 \leq \alpha \leq C \land \alpha^\top y = 0 \)

\[ \mu \geq 0 \land \sum_{k=1}^{p} \mu_k \text{Tr}(K_k) \leq \Lambda. \]

L1 regularization

\[ \text{single convex optimization (p + m parameters)}. \]
Previous Work

- **Linear Families:**
  - Lanckriet et al., *Learning the kernel matrix with semidefinite programming*. JMLR, 2004
  - Kloft et al., *Efficient and accurate lp-norm multiple kernel learning*. NIPS, 2009

- **Non-Linear Families:**
  - Ong et al., *Learning the kernel with hyperkernels*. JMLR, 2005
Centered Alignment-Based LK

- Two stages:

  - Kernel Selection \( \rightarrow K \rightarrow \) Learning Algorithm \( \rightarrow h \)

- Outperforms uniform baseline and previous algorithms.
- Centered alignment is key: different from notion used by (Cristiannini et al., 2001).

[C. Cortes, M. Mohri, and A. Rostamizadeh: Two-stage learning kernel methods, ICML 2010]
Centered Alignment

- Centered kernels:

\[ K_c(x, x') = (\Phi(x) - \mathbf{E}_x[\Phi])^\top (\Phi(x') - \mathbf{E}_{x'}[\Phi]) \]

- Centered alignment:

\[ \rho(K, K') = \frac{\mathbf{E}[K_cK'_c]}{\sqrt{\mathbf{E}[K_c^2]\mathbf{E}[K'_c^2]}} \]

where expectation is over pairs of points.

- Choose kernel to maximize alignment with the target kernel:

\[ K_Y(x_i, x_j) = y_iy_j. \]
Empirical Results

4,000 rank-1 base kernels.

RMSE

Books

Uniform 1-stage Simple align

Electronics

Uniform 1-stage Simple align

Kitchen

Uniform 1-stage Simple align
Feature Selection Benefits

- Kitchen appliances:
  - Large positive weights:
    great little, great product, is perfect, are great, and looks, beautiful and.
  - Large negative weights:
    a shame, doesn't work, very poor, return it, way too, very disappointed, after just, bother with.
Alignment-Based Kernel Learning

- Centered Alignment.
- Experimental results.
- Theoretical Results:
  - Concentration bound.
  - Existence of good predictors.
- Alignment algorithms:
  - Simple, highly scalable algorithm
  - Quadratic Program based algorithm.
Existence of Good Predictor

**Theorem:** let $h^*$ be the hypothesis defined for all $x$ by

$$h^*(x) = \frac{E_{x'}[y'K_c(x, x')]}{\sqrt{E[K_c^2]}}$$

and assume normalized labels: $E[y^2] = 1$. Then,

$$\text{error}(h^*) = E_x[(h^*(x) - y)^2] \leq 2(1 - \rho(K, K_Y)).$$
Proof

\[ \mathbb{E}_x[h^*(x)^2] = \mathbb{E}_x \left[ \frac{\mathbb{E}_{x'}[y'K_v(x, x')]^2}{\mathbb{E}[K_v^2]} \right] \]
\[ \leq \mathbb{E}_x \left[ \frac{\mathbb{E}_{x'}[y'^2]\mathbb{E}_{x'}[K_v^2(x, x')]}{\mathbb{E}[K_v^2]} \right] \quad \text{(Cauchy-Schwarz)} \]
\[ = \frac{\mathbb{E}_{x'}[y'^2]\mathbb{E}_{x', x'}[K_v^2(x, x')]}{\mathbb{E}[K_v^2]} = \mathbb{E}_{x'}[y'^2] = 1. \]

\[ \mathbb{E}[(y - h^*(x))^2] = \mathbb{E}_x[h^*(x)^2] + \mathbb{E}_x[y^2] - 2\mathbb{E}_x[yh^*(x)] \]
\[ \leq 1 + 1 - 2\rho(K, K_Y). \quad \text{(def. of } h^*) \]
Conclusion

- Centered alignment-based Learning Kernel algorithm:
  - based on new definition of centered alignment.
  - outperforms uniform combination in several tasks.
  - effective in classification and regression.
  - proof of existence of good predictors.
  - concentration bound for centered alignment.
  - algorithm reduced to a simple QP.

- Is there a better criterion for the first stage?
Outline

- Learning kernels
- Learning similarities
Google and Similarities

- A lot of Google is based on similarities:

- Given a query text, spoken utterance, or image Google returns relevant information (web pages, maps, images, etc.)

- Some of what we do at Google:
  - dream up new search applications;
  - design similarity functions to guarantee quality;
  - make them run effectively.

- How to learn a good similarity function for a particular task?
Image Search

Image search relies on some measure of similarity.

\[
\text{Sim}(x_1, x_2) = \left( \begin{array}{c} x_{11} \\ x_{12} \\ \vdots \\ x_{1N} \end{array} \right) \cdot \left( \begin{array}{c} x_{21} \\ x_{22} \\ \vdots \\ x_{2N} \end{array} \right)
\]
Clustering of Images

Compute all the pair-wise distances between related images and form clusters:

Cluster Dendrogram
More Like This - Image Browsing
[Yushi Jing, Henry A. Rowley, Charles R. Rosenberg, Jingbin Wang, Ming Zhao, Michele Covell: Google image swirl, a large-scale content-based image browsing system. ICME 2010: 267]
Learning Similarities

- Different methods and applications:
  - **unsupervised**:
    - full feature space.
    - low-dimensional embedding.
  - **supervised**:
    - full feature space.
    - low-dimensional embedding.
The Web is Huge

- Every year we add billions of documents to the index.
- Cannot compute billions of distances

  we need to make some approximations of the matrix $K$ of all pairwise similarities.
Nyström Approximation

- Idea: approximate the low dimensional space using sampling techniques.

- Definition: $K = \begin{bmatrix} W & K_{21}^T \\ K_{21} & K_{22} \end{bmatrix}$ and $C = \begin{bmatrix} W \\ K_{21} \end{bmatrix}$

\[ \tilde{K} = CW_k^+ C^T \approx K. \quad O(m^3 + nmk) \]

- Ensemble Nyström: accurate large-scale approx.
  - mixture of Nyström approx.: $\tilde{K}_{\text{ens}} = \sum_{r=1}^{p} \mu_r \tilde{K}_r$.
  - significant performance improvement.
  - stability-based theoretical guarantees.

[Ensemble Nyström Method, S. Kumar, M. Mohri, and A. Talwalkar, NIPS 2009]
People Hopper

http://people-hopper.googlelabs.com/
Supervised Learning - Full Space

- Learn from known pairs of similar and dissimilar images

\[ \text{Sim}(x_i, x_j) = x_i^T W x_j \quad W \in \mathbb{R}^{N \times N}, W \succeq 0 \]

- Training data is triplets \((x_i, x_i^+, x_i^-) \quad i \in [1; m]\)

- Goal of learning:

\[ \text{Sim}(x_i, x_i^+) \geq \text{Sim}(x_i, x_i^-) + 1 \]

\[ 0 \geq 1 - \text{Sim}(x_i, x_i^+) + \text{Sim}(x_i, x_i^-) \Rightarrow \]

\[ 0 \geq 1 - x_i W x_i^+ + x_i W x_i^- \]

- Ranking loss:

\[ l_W(x_i, x_i^+, x_i^-) = \max(0, 1 - x_i^T W x_i^+ + x_i^T W x_i^-) \]
Idea: let $W$ be unrestricted. Use ranking hinge loss, train as online learning, passive-aggressive algorithm.

**Ranking loss:**

$$l_W(x_i, x_i^+, x_i^-) = \max(0, 1 - x_i^T W x_i^+ + x_i^T W x_i^-)$$

**Online updates:**

$$W_i = \arg\min_W \frac{1}{2} \|W - W_i^{-1}\|_F^2 + C\xi \quad l_W(x_i, x_i^+, x_i^-) \leq \xi$$

If triplet OK, do nothing, else:

$$\mathcal{L}(W, \tau, \xi, \lambda) = \frac{1}{2} \|W - W_i^{-1}\|_F^2 + C\xi + \tau(1 - \xi - x_i^T W (x_i^+ - x_i^-)) - \lambda\xi$$

$$W_i = W_i^{-1} + \tau V_i \quad V_i = \frac{\partial l_W(x_i, x_i^+, x_i^-)}{\partial W} \quad \tau = \min \left\{ C, \frac{l_W(x_i, x_i^+, x_i^-)}{\|V_i\|^2} \right\}$$
AudioSwap - YouTube

Beat It

Dear corinnacortes,

Your video, Beat It, may have content that is owned or licensed by Sony Music Entertainment.

No action is required on your part; however, if you are interested in learning how this affects your video, please visit the Content ID Matches section of your account for more information.

Sincerely,

- The YouTube Team

PAMIR

Idea: Ranking algorithm for query/picture pairs (q,p)

- Queries are represented as a bag-of-words vector.
  \[ \text{Sim}(q, p) = q^T W p \]
- Pictures are mapped to words by the linear map
  \[ F_W(p) = W p \]
- Trained on triplets \((q_i, p_i^+, p_i^-)\)
- Ranking loss, passive-aggressive learning algorithm, just like OASIS.

Image Annotation

den lille havfrue

About 55,100 results (0.78 seconds)
Supervised - Low-Dimensional

Idea: projecting to low-dimensional joint embedding space:

\[ \text{Sim}(q, p) = qWp = q^T U^T Vp \quad U \in \mathbb{R}^{k \times N} \quad V \in \mathbb{R}^{k \times d} \]

- Ranking loss

\[ l(U, V)(q_i, p_i^+, p_i^-) = \max(0, 1 - q_i^T U^T Vp_i^+ + q_i^T U^T Vp_i^-) \]

- Stochastic gradient descent. If triplet OK, do nothing, else:

\[ U^{i+1} = U^i + \lambda V^i (p_i^+ - p_i^-) q_i^T \]
\[ V^{i+1} = V^i + \lambda U^i (p_i^+ - p_i^-) q_i^T \]

WSABI

**Idea:** focus on the top of the ranked list. Approximate the rank of a label.

- **Weighted Approximate-Ranking loss**

  \[
  l_{(\alpha, \mathbf{u}, \mathbf{v})}(q_i, p_i^+, p_i^-) = \alpha_{\text{rank}_{p_i^-}} \max(0, 1 - q_i^T \mathbf{U}^T \mathbf{V} p_i^+ + q_i^T \mathbf{U}^T \mathbf{V} p_i^-)
  \]

  \[
  \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_N \geq 0
  \]

  \[
  \alpha_j = 1 \quad \text{uniform weighting}
  \]

  \[
  \alpha_j = 1/j \quad \text{more weight on optimizing the top of the list}
  \]

- **Use sampling to estimate the rank of an error.**

WSABLI cow-like images

rank 1-5

rank 50-55

rank 100-105
Summary

- Challenging very large-scale problems:
  - effective ranking algorithms.
  - key low-rank approximations.
- Still more to do:
  - better use of user-derived information.
  - ranking at the top.
  - combined ranking and low-rank approximation.