Class is 170.

Announcements

Matlab Grader homework,
1 and 2 (of less than 9) homeworks Due 22 April **tonight**, Binary graded. 
For HW1, please get word count <100
167, 165, 164 has done the homework. **(If you have not done it talk to me/TA!)**
Homework 3 (released ~tomorrow) due ~5 May

Jupiter “GPU” home work released Wednesday. Due 10 May

Guidelines is on Piazza
May 5 proposal due. TAs and Peter can approve.

Today:
• Stanford CNN 9, Kernel methods (Bishop 6),
• Linear models for classification, Backpropagation

Monday
• Stanford CNN 10, Kernel methods (Bishop 6), SVM,
• Play with Tensorflow playground before class [http://playground.tensorflow.org](http://playground.tensorflow.org)
Projects

• **3-4** person groups preferred

• Deliverables: Poster & Report & main code (plus proposal, midterm slide)

• **Topics** your own or chose form suggested topics. Some **physics inspired**.

• **April 26 groups** due to TA (if you don’t have a group, ask in piazza we can help). TAs will construct group after that.

• **May 5** proposal due. TAs and Peter can approve.
• Proposal: One page: Title, A large paragraph, data, weblinks, references.
• Something **physical**
DataSet

- 80 % preparation, 20 % ML
- Kaggle: https://inclass.kaggle.com/datasets
  https://www.kaggle.com

- Past projects…

- Ocean acoustics data
In 2017 Many choose the source localization

- two CNN projects,

Many thanks for the fun projects! Below are the final projects from the class. Only the report is posted, the corresponding code is just as important.

1. Source localization in an ocean waveguide using supervised machine learning, Group3, Group6, Group8, Group10, Group11, Group15
2. Indoor positioning framework for most Wi-Fi-enabled devices, Group1
3. MyShake Seismic Data Classification, Group2
4. Multi Label Image Classification, Group4
5. Face Recognition using Machine Learning, Group7
6. Deep Learning for Star-Galaxy Classification, Group9
7. Modeling Neural Dynamics using Hidden Markov Models, Group12
8. Star Prediction Based on Yelp Business Data And Application in Physics, Group13
9. Si K edge X-ray spectrum absorption interpretation using Neural Network, Group14
10. Plankton Classification Using VGG16 Network, Group16
11. A Survey of Convolutional Neural Networks: Motivation, Modern Architectures, and Current Applications in the Earth and Ocean Sciences, Group17
12. Use satellite data to track the human footprint in the amazon rainforest, Group18
13. Automatic speaker diarization using machine learning techniques, Group19
14. Predicting Coral Colony Fate with Random Forest, Group20

---
<table>
<thead>
<tr>
<th>Group</th>
<th>Topic</th>
<th>Authors</th>
<th>Poster</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reimplementation of source localization in an ocean waveguide using supervised learning</td>
<td>Jinzhao Feng, Zhuoxi Zeng, Yu Zhang</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>2</td>
<td>Machine learning methods for ship detection in satellite images</td>
<td>Yifan Li, Huadong Zhang, Xiaoshi Li, Quianfeng Guo</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>3</td>
<td>Transparent Conductor Prediction</td>
<td>Yan Sun, Yiyuan Xing, Xufan Xiong, Tianduo Hao</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>4</td>
<td>Ship identification in satellite Images</td>
<td>Weilun Zhang, Zhaoliang Zheng, Mingchen Mao,</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>5</td>
<td>Fruit Recognition</td>
<td>Eskil Jarlskog, Richard Wang, Joel Andersson</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>6</td>
<td>RSNA Bone Age Prediction</td>
<td>Juan Camilo Castillo, Yitian Tong, Jiyang Zhao, Fengcan Zhu</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>7</td>
<td>Facial Expression Classification into Emotions</td>
<td>David Orozco, Christopher Lee, Yevgeniy Arabadzhi, Deval Gupta</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>8</td>
<td>Urban Scene Segmentation for Autonomous Vehicles</td>
<td>Hsiao-Chen Huang, Eddie Tseng, Ping-Chun Chiang, Chih-Yen Lin</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>9</td>
<td>Face Detection Using Deep Learning</td>
<td>Yu Shen, Kuan-Wei Chen, Yizhou Hao, Min Hsuan Wu</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>10</td>
<td>Understanding the Amazon Rainforest using Neural Networks</td>
<td>Naveen Dharshana Ketagoda, Christian Jonathan Koguchi, Niral Lalit Pathak, Samuel Sunarjo</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>11</td>
<td>Mercedes-Benz Bench Test Time Estimation</td>
<td>Lanjihong Ma, Kexiong Wu, Bo Xiao, Zihang Yu</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>12</td>
<td>Vegetation Classification in Hyperspectral Image</td>
<td>Osman Cihan Kilinc, Kazim Ergun, Yuming Qiao, Fengjunyan Li</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>13</td>
<td>Threat Detection Using AlexNet on TSA scans</td>
<td>Amartya Bhattacharyya, Christine H Lind, Rahul Shirpurkar</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>14</td>
<td>Flagellates Classification via Transfer Learning</td>
<td>Eric Ho, Brian Henriquez, Jeffrey Yeung</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>15</td>
<td>Biomedical Image Segmentation</td>
<td>Lucas Tindall, Amir Persekian, Max Jiao</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>16</td>
<td>“Deep Fakes” using Generative Adversarial Networks (GAN)</td>
<td>Tianxiang Shen, Ruixian Liu, Ju Bai, Zheng Li</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>17</td>
<td>Dog Breed Classification via Convolutional Neural Network</td>
<td>Yizhou Chen; Xiaotong Chen; Xuanzhen Xu</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>18</td>
<td>Dog Breed Identification</td>
<td>Wenting Shi, Jiaquan Chen, Fangyu Liu, Muyun Liu</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>19</td>
<td>Impact of Skewed Distributions on an Automated Plankton Classifier</td>
<td>Will Chapman, Ema Fatima, William Jenkins, Steven Tien, Shawheen Tosifian</td>
<td>Poster</td>
<td>Paper</td>
</tr>
<tr>
<td>20</td>
<td>Blood Cell Detection using Single-shot MultiBox Detector</td>
<td>Hyoung Hui</td>
<td>Poster</td>
<td>Paper</td>
</tr>
</tbody>
</table>
Bayes and Softmax (Bishop p. 198)

- **Bayes:**
  \[
  p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{y \in Y} p(x, y)}
  \]

- **Classification of N classes:**
  \[
  p(C_n|x) = \frac{p(x|C_n)p(C_n)}{\sum_{k=1}^{N} p(x|C_k)p(C_k)} = \frac{\exp(a_n)}{\sum_{k=1}^{N} \exp(a_k)}
  \]
  with
  \[
  a_n = \ln (p(x|C_n)p(C_n))
  \]

Parametric Approach: Linear Classifier

\[
\mathbf{f}(\mathbf{x}, \mathbf{W}) = \mathbf{Wx} + \mathbf{b}
\]

Array of $32 \times 32 \times 3$ numbers (3072 numbers total)
Softmax to Logistic Regression (Bishop p. 198)

\[ p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{\sum_{k=1}^{2} p(\mathbf{x}|C_k)p(C_k)} \]

\[ = \frac{\exp(a_1)}{\sum_{k=1}^{2} \exp(a_k)} = \frac{1}{1 + \exp(-a)} \]

with

\[ a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} \]

- \( a_1 = \ln[p(\mathbf{x}|C_1)p(C_1)] \)
- \( a = a_1 - a_2 \)
- \( p(C_1|x) = \frac{1}{1 + \exp(a_2-a_1)} \)
The Kullback-Leibler Divergence

P true distribution, q is approximating distribution

\[ \text{KL}(p||q) = - \int p(x) \ln q(x) \, dx - \left( - \int p(x) \ln p(x) \, dx \right) \]

\[ = - \int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} \, dx \]

\[ \text{KL}(p||q) \simeq \frac{1}{N} \sum_{n=1}^{N} \left\{ - \ln q(x_n|\theta) + \ln p(x_n) \right\} \]

\[ \text{KL}(p||q) \geq 0 \quad \text{KL}(p||q) \neq \text{KL}(q||p) \]
Cross entropy

• KL divergence (\( p \) true \( q \) approximating)

\[
D_{\text{KL}}(p||q) = \sum_{n} p_n \ln(p_n) - \sum_{n} p_n \ln(q_n) \\
= -H(p) + H(p, q)
\]

• Cross entropy

\[
H(p, q) = H(q) + D_{\text{KL}}(p||q) = -\sum_{n} p_n \ln(q_n) \\
= -p_k \ln(q_{\text{true}}) = -\ln(q_{\text{true}})
\]

• Implementations

\texttt{tf.keras.losses.CategoricalCrossentropy()}
\texttt{tf.losses.sparse_softmax_cross_entropy}
\texttt{torch.nn.CrossEntropyLoss()}
Cross-entropy or "**softmax**" function for multi-class classification

The output units use a non-local non-linearity:

\[ y_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \]

\[ \frac{\partial y_i}{\partial z_i} = y_i (1 - y_i) \]

The natural cost function is the negative log prob of the right answer

\[ E = -\sum_j t_j \ln y_j \]

\[ \frac{\partial E}{\partial z_i} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_i} = y_i - t_i \]
Reminder: 1x1 convolutions

- 1x1 CONV with 32 filters
  (each filter has size 1x1x64, and performs a 64-dimensional dot product)

Preserves spatial dimensions, reduces depth!

Projects depth to lower dimension (combination of feature maps)

Summary: CNN Architectures

Case Studies
- AlexNet
- VGG
- GoogLeNet
- ResNet
Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC’15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC’15 and COCO’15!
Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it’s not caused by overfitting!

Hypothesis: the problem is an \textit{optimization} problem, deeper models are harder to optimize
Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

\[ H(x) \approx x + F(x) \]

Use layers to fit residual \( F(x) = H(x) - x \) instead of \( H(x) \) directly

\[ H(x) = F(x) + x \]
Kernels

We might want to consider something more complicated than a linear model:

**Example 1:** \([x^{(1)}, x^{(2)}] \rightarrow \Phi \left([x^{(1)}, x^{(2)}]\right) = [x^{(1)2}, x^{(2)2}, x^{(1)}x^{(2)}]\)

Information unchanged, but now we have a **linear** classifier on the transformed points.

With the kernel trick, we just need kernel

\[ k(a, b) = \Phi(a)^T \Phi(b) \]

\[ k(x, x') = \phi(x)^T \phi(x'). \]  \hspace{1cm} (6.1)
Basis expansion

\[ \begin{align*}
\mathbf{w}^{T} \varphi(\mathbf{x}) &> 0 \\
\mathbf{w}^{T} \varphi(\mathbf{x}) &< 0 \\
\mathbf{w}^{T} \varphi(\mathbf{x}) = 0 &= x_1 - x_2^2 - 2
\end{align*} \]

\[ x_1 = x_2^2 + 2 \]

\[ x_1 - x_2^2 - 2 = 0 \]

\[ \varphi = [\varphi_1, \varphi_2] = [x_1, x_2, x_1^2, x_2^2, x_1 x_2] \]

\[ \mathbf{w} = [-2, 1, 0, 0, -1, 0] \]

\[ M = 6 \]

\[ \text{linearly separable} \]
This is what a Gaussian process \textit{posterior} looks like with 4 data points and a squared exponential covariance function. The bold blue line is the predictive mean, while the light blue shade is the predictive uncertainty (2 standard deviations). The model uncertainty is small near the data, and increases as we move away from the data points.

\begin{equation}
t_n = y_n + \epsilon_n
\end{equation}

\begin{equation}
f(x) \sim GP(m(x), \kappa(x, x'))
\end{equation}

\textbf{Figure 15.2}  Left: some functions sampled from a GP prior with SE kernel. Right: some samples from a GP posterior, after conditioning on 5 noise-free observations. The shaded area represents $\mathbb{E}[f(x)] \pm 2\text{std}(f(x))$. Based on Figure 2.2 of (Rasmussen and Williams 2006). Figure generated by gprDemoNoiseFree.
Dual representation, Sec 6.2

Primal problem: \( \min_w E(w) \)

\[
E = \frac{1}{2} \sum_n (w^T x_n - t_n)^2 + \frac{\lambda}{2} \|w\|^2 = \|Xw - t\|_2^2 + \frac{\lambda}{2} \|w\|^2
\]

Solution: \( \hat{w} = X^+ t = (X^T X + \lambda I_M)^{-1} X^T t \)

\[
= X^T (X X^T + \lambda I_N)^{-1} t = X^T (K + \lambda I_N)^{-1} t = X^T a
\]

The kernel is \( K = X X^T \)

Dual representation is: \( \min_a E(a) \)

\[
E = \frac{1}{2} \sum_n (w^T x_n - t_n)^2 + \frac{\lambda}{2} \|w\|^2 = \|K a - t\|_2^2 + \frac{\lambda}{2} a^T K a
\]

\( a \) is found inverting \( N \times N \) matrix

\( w \) is found inverting \( M \times M \) matrix

Only kernels, no feature vectors
Dual representation, Sec 6.2

Dual representation is:

\[ \min_a E(a) \]

\[ E = \frac{1}{2} \sum_n \{w^T x_n - t_n \}^2 + \frac{\lambda}{2} \|w\|^2 = \|Ka - t\|^2 + \frac{\lambda}{2} a^T Ka \]

Prediction

\[ y = w^T x = a^T X x = \sum_n a_n x_n^T x = \sum_n a_n k(x_n, x) \]

- Often \( a \) is sparse (… Support vector machines) SVM
- We don’t need to know \( x \) or \( \phi(x) \). Just the Kernel

\[ E(a) = \|Ka - t\|^2 + \frac{\lambda}{2} a^T Ka \]

\[ \exp\left(-\xi \frac{\|x_i - x_j\|}{\delta}\right) \]
Gaussian Kernels

- Gaussian Kernel

\[ k(x, x') = \exp \left( -\frac{1}{2} (x - x')^T \Sigma^{-1} (x - x') \right) \]

Diagonal \( \Sigma \): (this gives ARD)

\[ k(x, x') = \exp \left( -\frac{1}{2} \sum_{i}^{N} \frac{(x_i - x'_i)^2}{\sigma_i^2} \right) \]

Isotropic \( \sigma_i^2 \) gives an RBF

\[ k(x, x') = \exp \left( -\frac{\|x - x'\|_2^2}{2\sigma^2} \right) \]

\[ \Sigma = \sigma^2 I \]
Commonly used kernels

Polynomial:  \[ K(x, y) = (x \cdot y + 1)^p \]

Gaussian radial basis function:  \[ K(x, y) = e^{-\|x-y\|^2 / 2\sigma^2} \]

Neural net:  \[ K(x, y) = \tanh (k x \cdot y - \delta) \]

Parameters that the user must choose

For the neural network kernel, there is one “hidden unit” per support vector, so the process of fitting the maximum margin hyperplane decides how many hidden units to use. Also, it may violate Mercer’s condition.
Example 4:

\[
 k(x, z) = (x^T z + c)^2 = \left( \sum_{j=1}^{n} x^{(j)} z^{(j)} + c \right) \left( \sum_{\ell=1}^{n} x^{(\ell)} z^{(\ell)} + c \right) \\
= \sum_{j=1}^{n} \sum_{\ell=1}^{n} x^{(j)} x^{(\ell)} z^{(j)} z^{(\ell)} + 2c \sum_{j=1}^{n} x^{(j)} z^{(j)} + c^2 \\
= \sum_{j,\ell=1}^{n} (x^{(j)} x^{(\ell)}) (z^{(j)} z^{(\ell)}) + \sum_{j=1}^{n} (\sqrt{2cx^{(j)}}) (\sqrt{2cz^{(j)}}) + c^2,
\]

and in \( n = 3 \) dimensions, one possible feature map is:

\[
 \Phi(x) = [x^{(1)^2}, x^{(1)} x^{(2)}, \ldots, x^{(3)^2}, \sqrt{2cx^{(1)}}, \sqrt{2cx^{(2)}}, \sqrt{2cx^{(3)}}, c]
\]

and \( c \) controls the relative weight of the linear and quadratic terms in the inner product.

Even more generally, if you wanted to, you could choose the kernel to be any higher power of the regular inner product.
• FINISHED HERE 30 April 2018
• Showed also http://playground.tensorflow.org/ in the last 10 min.