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**

MNIST

Jupiter "GPU" home work released Wednesday. Due 10 May

Projects: 27 Groups formed. Look at Piazza for help.
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

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 piaza 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
- UCI datasets: http://archive.ics.uci.edu/ml/index.php

- 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

Syllabus

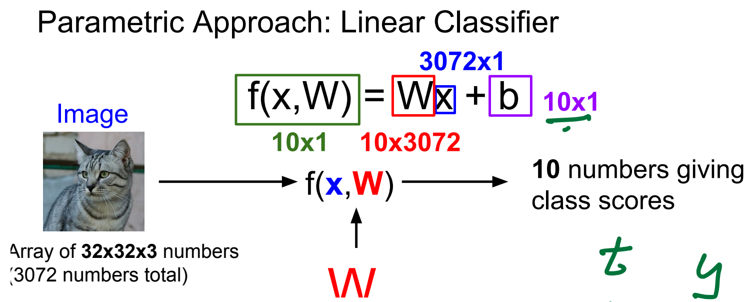
2018: Best reports 6,10,12 15; interesting 19, 47
poor 17; alone is hard 20.

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

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:

$$p(\mathcal{C}_n|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_n)p(\mathcal{C}_n)}{\sum_{k=1}^{N} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}$$

$$= \frac{\exp(a_n)}{\sum_{k=1}^{N} \exp(a_k)}$$

with

$$a_n = \ln\left(p(\mathbf{x}|\mathcal{C}_n)p(\mathcal{C}_n)\right)$$

likelihood prior

t	y
1	0.75
0	.25
0	0
0	0
0	0
target	pred

Softmax to Logistic Regression (Bishop p. 198)

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{\sum_{k=1}^{2} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}$$

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

logistic reg

sigmoid

$= \exp(a_1) + \exp(a_2)$

with

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$\sum_{n}^{2} p(C_n|x) = 1$$

- $a_1 = \ln[p(\boldsymbol{x}|C_1)p(C_1)]$
- $a = a_1 - a_2$
- $p(C_1|x) = \frac{1}{1+\exp(a_2 - a_1)}$

The Kullback-Leibler Divergence

P true distribution, q is approximating distribution

$$\begin{aligned} \mathrm{KL}(p\|q) &= -\int p(\mathbf{x}) \ln q(\mathbf{x})\, \mathrm{d}\mathbf{x} - \left(-\int p(\mathbf{x}) \ln p(\mathbf{x})\, \mathrm{d}\mathbf{x}\right) \\ &= -\int p(\mathbf{x}) \ln \left\{\frac{q(\mathbf{x})}{p(\mathbf{x})}\right\} \mathrm{d}\mathbf{x} \end{aligned}$$

$$\mathrm{KL}(p\|q) \simeq \frac{1}{N} \sum_{n=1}^{N} \{-\ln q(\mathbf{x}_n|\boldsymbol{\theta}) + \ln p(\mathbf{x}_n)\}$$

$$\mathrm{KL}(p\|q) \geqslant 0 \qquad \mathrm{KL}(p\|q) \not\equiv \mathrm{KL}(q\|p)$$

Cross entropy

- KL divergence (p true q approximating)

$$D_{\{KL\}}(p||q) = \sum_n^N p_n \ln(p_n) - \sum_n^N p_n \ln(q_n)$$
$$= -H(p) + \underbrace{H(p,q)}_{\text{cross entropy}}$$

- Cross entropy

$$H(p,q) = H(q) + D_{\{KL\}}(p||q) = -\sum_n^N p_n \ln(q_n)$$

$$= -p_k \ln(q_k) = -\ln(q_k)$$

- *Implementations*

tf.keras.losses.CategoricalCrossentropy()

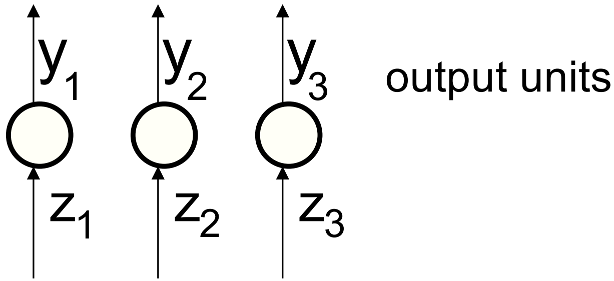
tf.losses.sparse_softmax_cross_entropy

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

target value

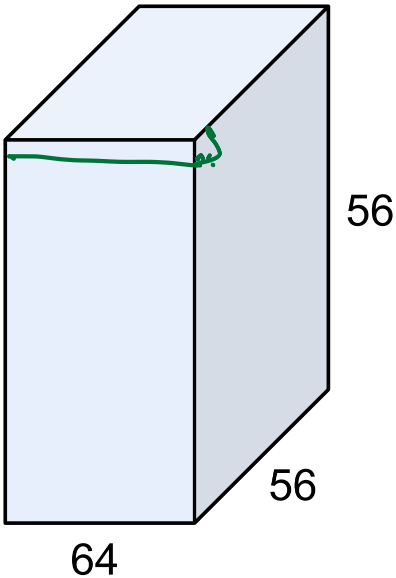
$$E = -\sum_j t_j \ln y_j$$

$$\frac{\partial E}{\partial z_i} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_i} = y_i - t_i$$

Reminder: 1x1 convolutions

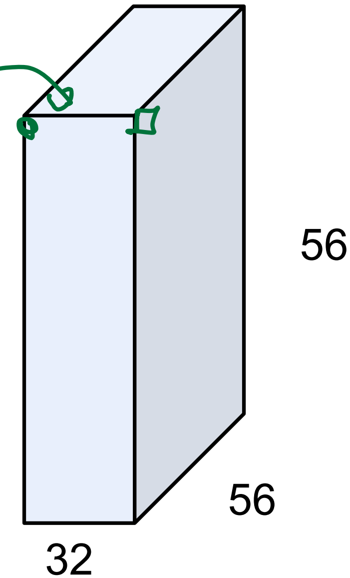
preserves spatial
dimensions, reduces depth!

Projects depth to lower
dimension (combination of
feature maps)



1x1 CONV
with 32 filters

(each filter has size 1x1x64, and performs a 64-dimensional dot product)



Summary: CNN Architectures

Case Studies

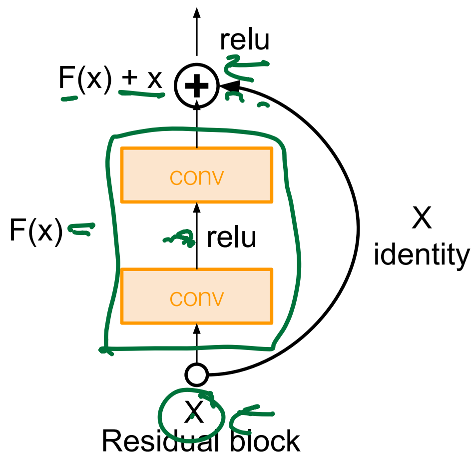
- AlexNet
- VGG
- GoogLeNet
- ResNet

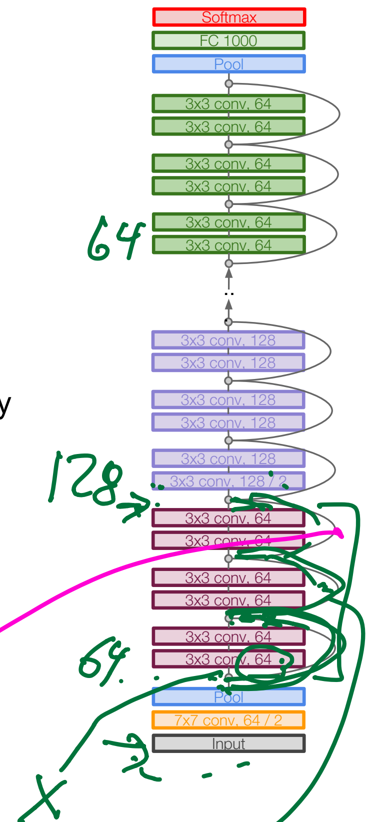
Case Study: ResNet

[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!





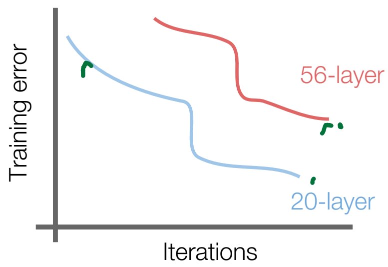
$X + F(x) + F_2(x + F(x))$

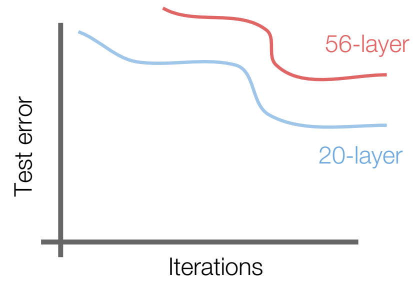
$X + F(x)$

Case Study: ResNet

[He et al., 2015]

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





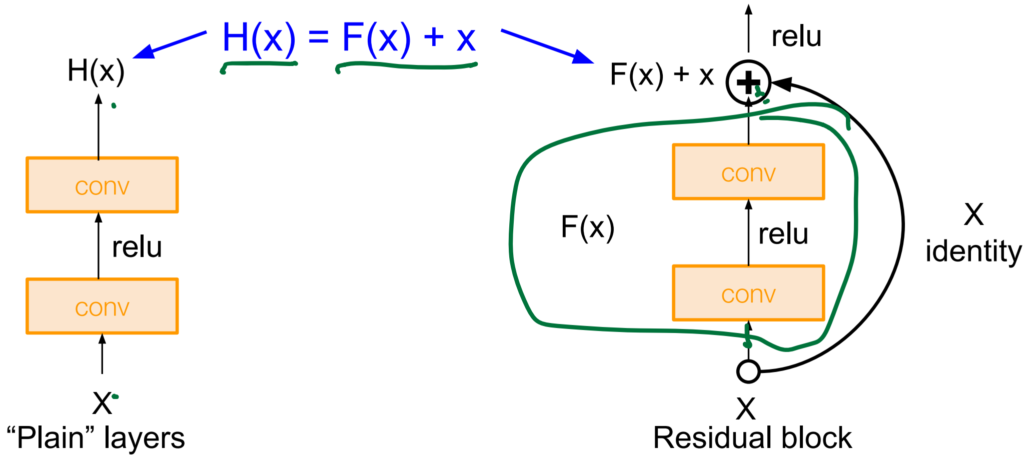
56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it’s not caused by overfitting!

Hypothesis: the problem is an *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



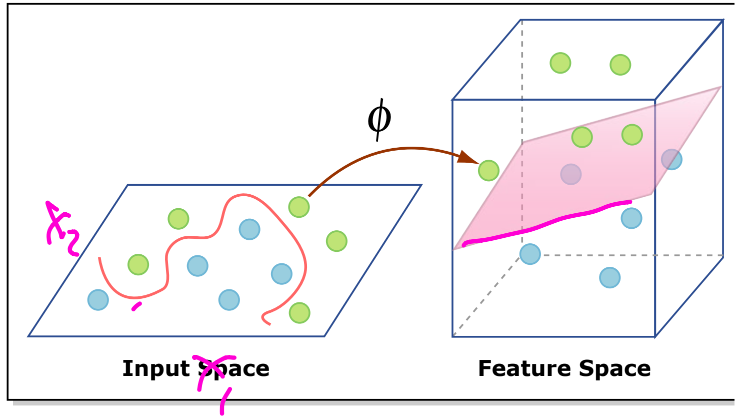
Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

$$H(x) \approx \underline{x} + \underline{F(x)}$$

Kernels

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

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

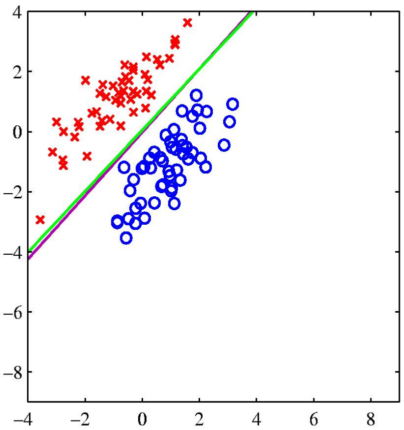


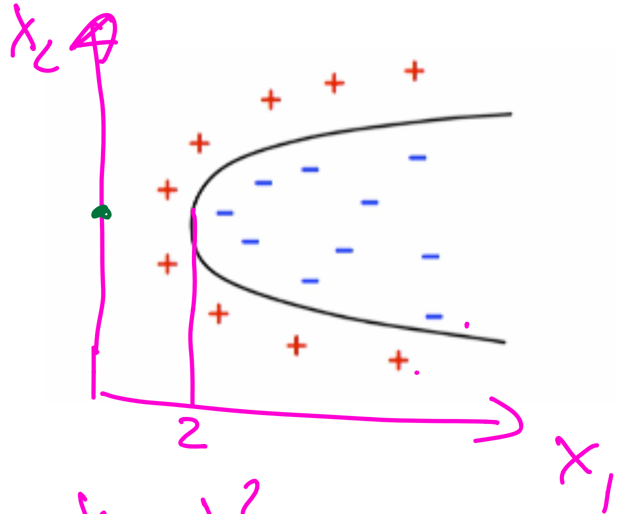
Information unchanged, but now we have a **linear** classifier on the transformed points.

With the kernel trick, we just need kernel $k(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{\Phi}(\boldsymbol{a})^T\ \boldsymbol{\Phi}(\boldsymbol{b})$

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}'). \tag{6.1}$$

Basis expansion





$$x_1 = x_2^2 + 2$$

$$x_1 - x_2^2 - 2 = 0$$

$M = 6$

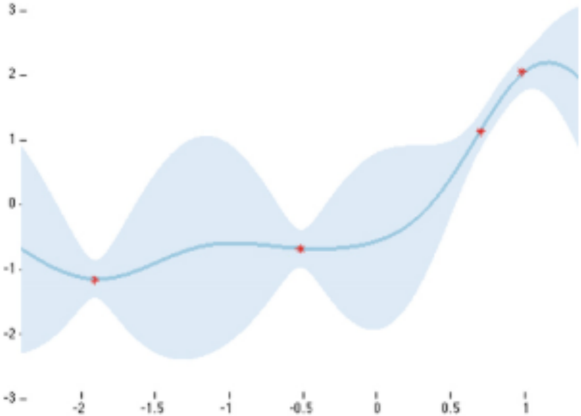
$$w^T \phi(x) \geq 0$$
$$< 0$$

$$\phi = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$$

$$w = [-2, 1, 0, 0, -1, 0]$$

$$w^T \phi(x) = 0 = x_1 - x_2^2 - 2$$

Gaussian Process (Bishop 6.4, Murphy15)



$$t_n = y_n + \epsilon_n$$

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$$

This is what a Gaussian process *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.

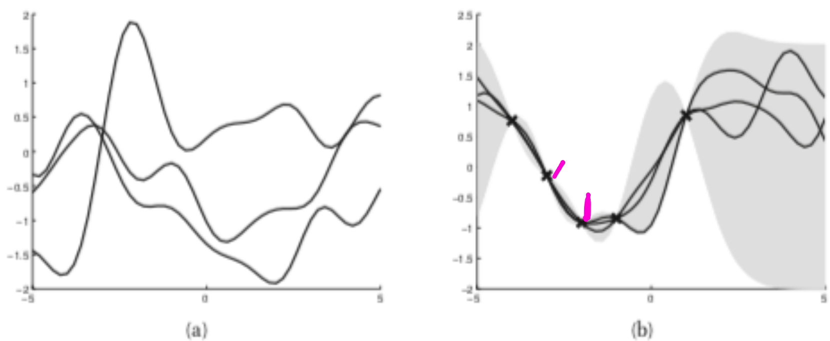


Figure 15.2 Left: some functions sampled from a GP prior with SE kernel. Right: some samples from a GP posterior, after conditioning on 5 noise-free observations. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\text{std}(f(\mathbf{x}))$. Based on Figure 2.2 of (Rasmussen and Williams 2006).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

Dual representation, Sec 6.2

Primal problem: $\min_{\boldsymbol{w}} E(\boldsymbol{w})$

$$E = \frac{1}{2}\sum_n^N \{\boldsymbol{w}^T \boldsymbol{x}_n - t_n\}^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

$t = \underbrace{\begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}}_{N}$

$X = \overbrace{\begin{bmatrix} - & x_1^T & - \\ - & \vdots & - \\ - & x_N^T & - \end{bmatrix}}^{M} \Big\} N \quad \in R^{N \times M}$

$N > M$

Solution $\boldsymbol{w} = \boldsymbol{X}^+\boldsymbol{t} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_M)^{-1}\boldsymbol{X}^T\boldsymbol{t}$

$$= \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_N)^{-1}\boldsymbol{t} = \boldsymbol{X}^T(\boldsymbol{K} + \lambda \boldsymbol{I}_N)^{-1}\boldsymbol{t} = \boldsymbol{X}^T\boldsymbol{a}$$

The kernel is $\mathbf{K} = \boldsymbol{X}\boldsymbol{X}^T$ $\in R^{N \times N}$

$a = (K + \lambda I_N)^{-1} t$ $\in R^N$

$X^T X \in R^{M \times M}$

$X X^T \in R^{N \times N}$

Dual representation is : $\min_{\boldsymbol{a}} E(\boldsymbol{a})$

$$E = \frac{1}{2}\sum_n^N \{\boldsymbol{w}^T \boldsymbol{x}_n - t_n\}^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 = \|\boldsymbol{K}\boldsymbol{a} - \boldsymbol{t}\|_2^2 + \frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a}$$

a is found inverting NxN matrix

w is found inverting MxM matrix

Only kernels, no feature vectors

Dual representation, Sec 6.2

Dual representation is : $\min_{\boldsymbol{a}} E(\boldsymbol{a})$

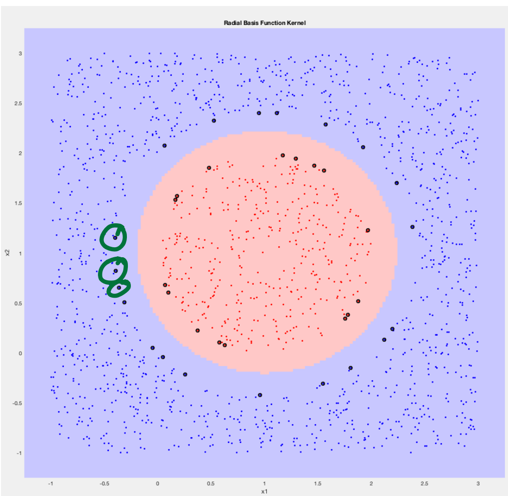
$$E = \frac{1}{2}\sum_n^N\{\boldsymbol{w}^T\boldsymbol{x}_n - t_n\}^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 = \|\boldsymbol{Ka} - \boldsymbol{t}\|_2^2 + \frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{Ka}$$

Prediction

$$y = \boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{x} = \sum_n^N a_n \boxed{\boldsymbol{x}_n^T\boldsymbol{x}} = \sum_n^N a_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

- Often **a is sparse (…** Support vector machines**)** SVM
- We don't need to know **x or** $\boldsymbol{\varphi}(\boldsymbol{x})$. ***Just the Kernel***

$$E(\boldsymbol{a}) = \|\boldsymbol{Ka} - \boldsymbol{t}\|_2^2 + \frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{Ka}$$



$$\exp\left(-\gamma \frac{\|x_1 - x_2\|}{2}\right)$$

Gaussian Kernels

- Gaussian Kernel

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

Diagonal Σ: (this gives ARD)

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_N^2 \end{bmatrix}$$

$$k(x, x') = \exp\left(-\frac{1}{2}\sum_i^N \frac{(x_i - x_i')^2}{\sigma_i^2}\right)$$

Isotropic σ_i^2 gives an RBF

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

$$\Sigma = \sigma^2 I$$

Commonly used kernels

Polynomial: $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y} + 1)^p$

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

Parameters that the user must choose

Neural net: $K(\mathbf{x},\mathbf{y}) = \tanh(k\,\mathbf{x}.\mathbf{y} - \delta)$

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:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T\mathbf{z} + c)^2 = \left(\sum_{j=1}^{n} x^{(j)} z^{(j)} + c\right)\left(\sum_{\ell=1}^{n} x^{(\ell)} z^{(\ell)} + c\right) \\
&= \sum_{j=1}^{n}\sum_{\ell=1}^{n} x^{(j)} x^{(\ell)} z^{(j)} z^{(\ell)} + 2c\sum_{j=1}^{n} x^{(j)} z^{(j)} + c^2 \\
&= \sum_{j,\ell=1}^{n} (x^{(j)} x^{(\ell)})(z^{(j)} z^{(\ell)}) + \sum_{j=1}^{n} (\sqrt{2c}x^{(j)})(\sqrt{2c}z^{(j)}) + c^2,
\end{aligned}
$$

and in $n = 3$ dimensions, one possible feature map is:

$$\mathbf{\Phi}(\mathbf{x}) = [x^{(1)2}, x^{(1)}x^{(2)}, ..., x^{(3)2}, \sqrt{2c}x^{(1)}, \sqrt{2c}x^{(2)}, \sqrt{2c}x^{(3)}, c]$$

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.