CIS 520 Machine Learning Summary

Lyle Ungar

What we covered
What we didn't cover

Review Session: Monday
Final: Wednesday 10:00

Course goals

- **Be familiar with all major ML methods**
  - Regression (linear, logistic), regularization, feature selection
  - K-NN, Decision trees, Random Forests, SVMs
  - PCA, K-means, GMM, Autoencoders
  - Naive Bayes, Bayes Nets, Markov Nets, LDA, HMMs
  - Boosting, perceptrons, LMS
  - Deep learning (CNNs, GANs)
  - Reinforcement Learning (MDP, Q-learning)
- **Know their strengths and weaknesses**
  - know jargon, concepts, theory
  - be able to modify and code algorithms
  - be able to read current literature

We did all of these!

Components of ML

- **Representation**
  - Feature set
  - Model form
- **Loss function**
  - And regularization penalty
- **Optimization method**
  - For parameter estimation
  - For model selection and hyperparameter tuning

Representations

- **Linear models**
  - Hyperplane as a separator
  - Kernel methods
- **Decision Trees**
  - Random forests, gradient tree boosting
- **Neural nets**
  - CNNs, GANs and Conditional GANs
  - Recurrent Nets/LSTMs
- **Structured X, y:** Graphs & Trees (not covered)

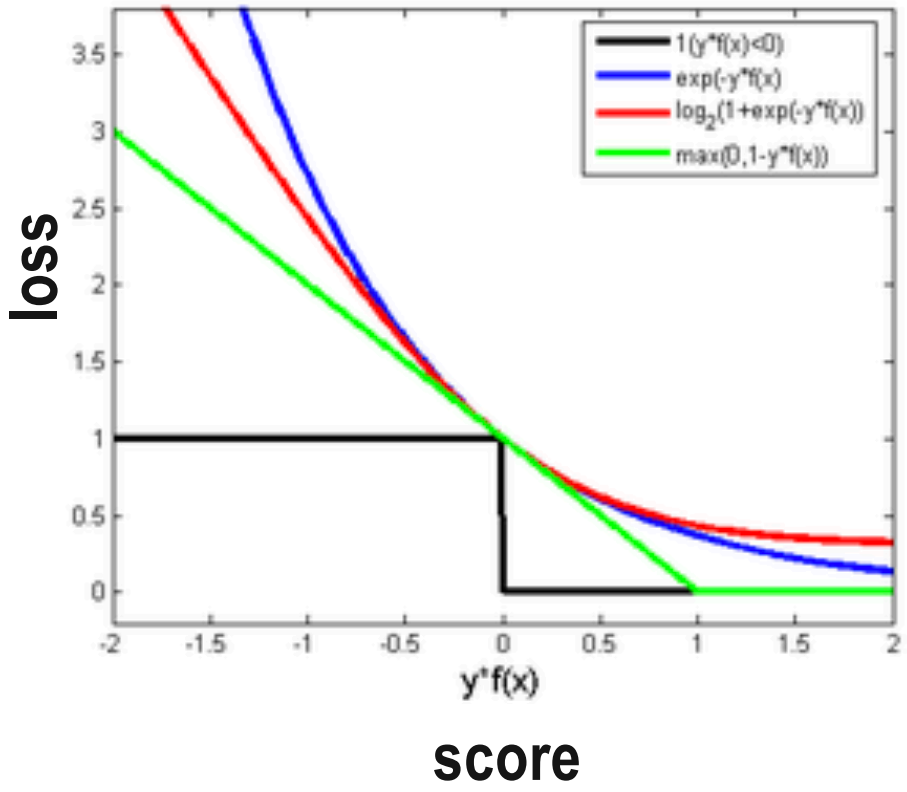
What loss functions have we used?

- L0, L1, L2
- Log-likelihood (MLE, MAP)
- Hinge
- Logistic
- Exponential
- Cross-Entropy; KL-divergence

$$\textbf{Boosting}: \quad \exp(-y_i f_\alpha(\mathbf{x}_i)) \quad \textbf{Logistic}: \quad \log(1+\exp(-y_i f_{\mathbf{w}}(\mathbf{x}_i)))$$

Loss Functions

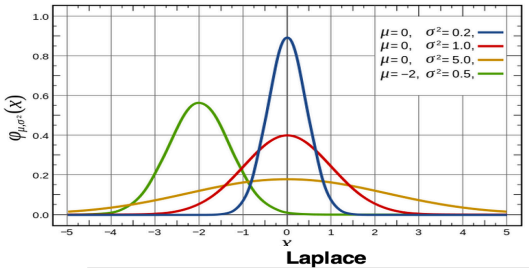
- L_0
- Hinge
- Logistic
- Exponential (adaboost)

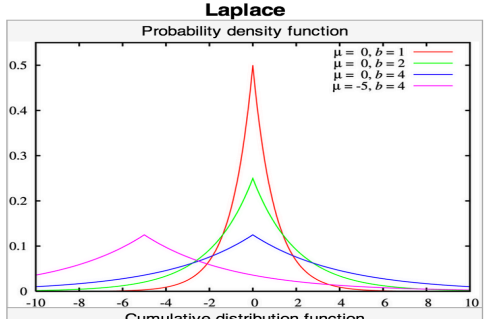


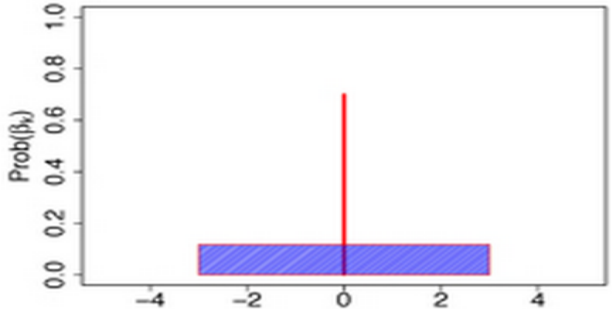
Regularization priors

Argmin$_w$ $||\mathbf{y} - \mathbf{w}\cdot\mathbf{x}||_2^2 + \lambda\, ||\mathbf{w}||_p^p$

- **L_2** $||\mathbf{w}||_2^2$
  - Gaussian prior: $p(w) \sim exp(-|w|_2^2/\sigma^2)$
- **L_1** $||\mathbf{w}||_1$
  - Laplace prior: roughly $p(w) \sim exp(-|w|_1/\sigma^2)$
- **L_0** $||\mathbf{w}||_0$
  - Spike and slab prior







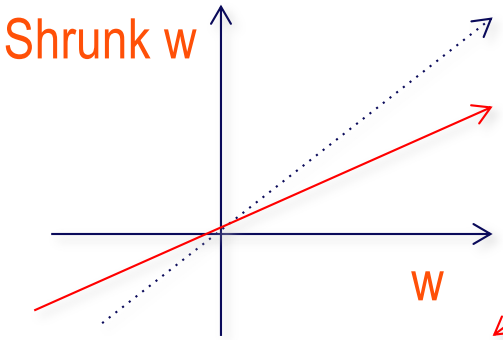
$$\log P(\mathcal{D}_X, \mathcal{D}_Y, \theta) = \log P(\mathcal{D}_X, \mathcal{D}_Y \mid \theta) + \log P(\theta) = -loss(\theta) + regularizer(\theta)$$

L_0, L_1 and L_2 Penalties

- If the x's have been standardized (mean zero, variance 1) then we can visualize the shinkage:

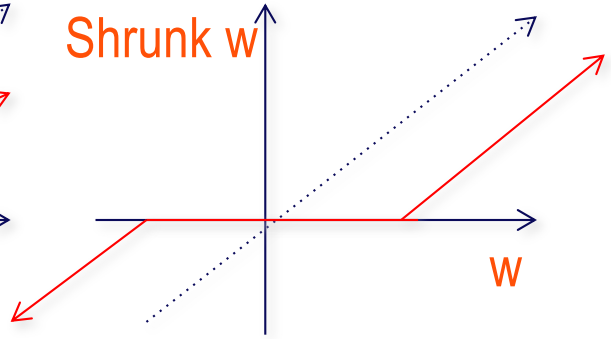
L_2 = Ridge

sum of squares



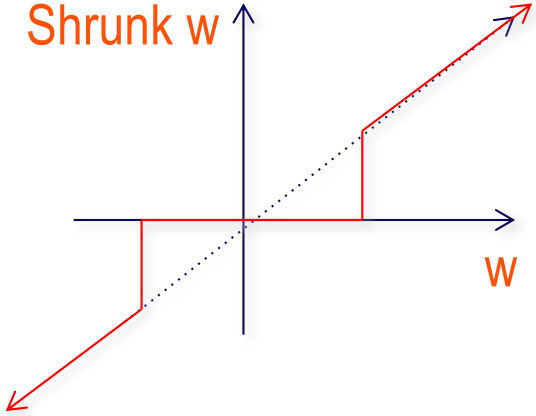
L_1 = Lasso

sum of abs value



L_0 = "stepwise regression"

Number of features



Bias-Variance Trade-off

$$\mathbf{E}_{x,y,D}[(h(x;D) - y)^2] =$$

$$: \underbrace{\mathbf{E}_{x,D}[(h(x;D) - \bar{h}(x))^2]}_{\text{Variance}} + \underbrace{\mathbf{E}_{x}[(\bar{h}(x) - \bar{y}(x))^2]}_{\text{Bias}^2} + \underbrace{\mathbf{E}_{x,y}[(\bar{y}(x) - y)^2]}_{\text{Noise}}$$

Optimization methods

- **Gradient descent:** Stochastic, minibatch
- **Streaming/Online methods:** LMS, Perceptron
- **Closed form** (e.g. $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$)
- **Search:** streamwise, stepwise, stagewise
- **Power method** (for eigenvectors, SVD)
- **Lagrange Multipliers** (constrained optimization)
  - not really covered

Optimization methods

- **EM** (alternating gradient descent in likelihood)
  - **E**: expected value of hidden values
  - **M**: MLE or MAP estimate of parameters
- **Other alternating methods**
  - $\mathbf{X} \sim \mathbf{S}\mathbf{W}^T$ for ICA, NNMF (non-negative matrix factorization)
  - V or Q and policy
  - Model and optimum (response surface)

Hyperparameter Optimization

- **Search**
  - e.g., L_1, L_2, penalties
  - Neural network structure, regularization
- **Auto-SKlearn**
  - Initialize hyperparameters from model predicting accuracy as a function of problem description and hyperparameter values
- **Auto-ML**
  - Use reinforcement learning to learn a ‘design policy’

Distance and similarity

- **Distances induced from norms**
  - $||x_1-x_2||_0$ $||x_1-x_2||_1$ $||x_1-x_2||_2$ …
- **Similarities from kernels**
  - $k(x_1,x_2)$
- **Probability-based divergence**
  - $D_{KL}(p||q) = \sum_k p_k \log(p_k/q_k)$ - KL-divergence
  - $H(p,q) = H(p) + D_{KL}(p||q)$ - cross-entropy

    $= - \sum_k p_k \log(q_k)$
    - *p* is the true distribution, *q* is the approximation

Cross entropy and log-likelihood

- **Cross-entropy**
  - $H(p,q) = - \sum_k p_k \log(q_k)$ summed over labels k
- $- \sum_i \sum_k \delta_{ik} \log(p(y_i=k|x=x_i))$ $\delta_{ik}=1$ *iff* $y_i=k$
  - - Sum of the estimated log probabilities of the true answers
- $\log \prod_i p(y_i|x_i) = \sum_i \log p(y_i|x_i)$ log-likelihood

KL-Divergence

- $D_{KL}(p||q) = \Sigma_k\ p_k\ \log(p_k/q_k)$
- **Mutual information**
  - $MI(X,Y) = D_{KL}(P(X,Y)\ ||\ P(X)P(Y))$
- **Information gain**
  - $IG(Y|X_j) = D_{KL}(P(Y|X_j)\ ||\ P(Y)) = H(Y) - H(Y|X_j)$
  - Which feature X_j will maximize the information gain?
- **Bayesian Experimental Design**
  - For which x will the label y (in expectation) most change p(w)

https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

Types of Learning

- **Supervised** **X, y**
  - Given an observation ***x***, what is the best label *y*?
- **Unsupervised** **X**
  - Given a set of ***x***'s, cluster or summarize them
- **Reinforcement**
  - Given a sequence of states ***x*** and possible actions ***a***, learn which actions maximize reward.

What kind of learning is missing here?

Unsupervised methods

- **PCA, ICA, NNMF**
  - $X \sim S V^T$
- **K-means, GMM**
- **Auto-encoders**
  - Information bottleneck
  - Denoising
  - Variational

Most of these minimize reconstruction error subject to some constraints

Bayesian Belief Nets

- **NB**
  - Binary or real-valued X's;
- **Belief Net**
- **GMM**
  - Different model forms
- **LDA**
- **HMM/MDP**

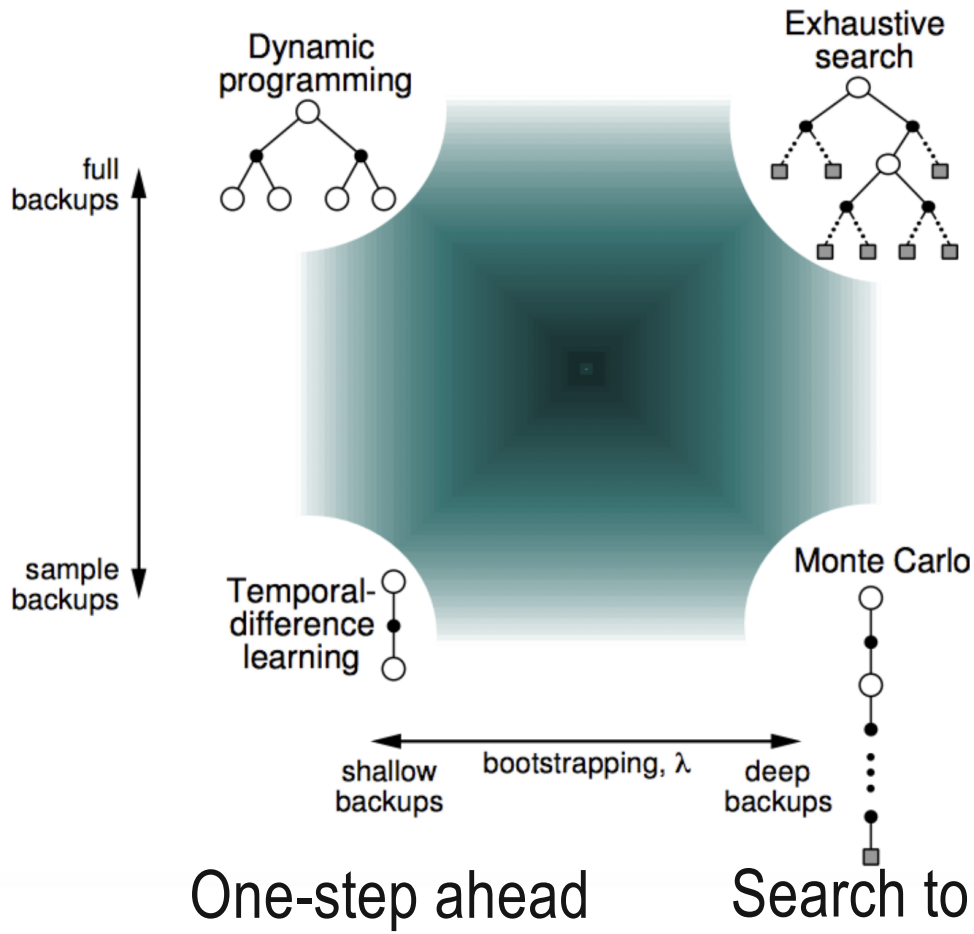
Reinforcement learning

- **Model-based**
  - MDP
- **Model-free**
  - **Shallow**: TD(0) vs. **Deep**: Monte-Carlo
  - **Value:** V(s) vs. **Q-learning** Q(s,a)
- **On policy** (ε-greedy) **vs. off-policy**
  - Trade off *exploration* and *exploitation*

Summary

Response to all possible actions

Response to one possible actions



Model-based

Model-free

One-step ahead

Search to end

From David Silver UCL Course on RL: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

Q-learning

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{(1-\alpha)}_{\text{We just used 1 here}} \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

After we take an ϵ-greedy action

For any MDP, given infinite exploration time and a partly-random policy, *Q*-learning will find an optimal policy: one that maximizes the expected value of the total reward over all successive steps.

wikipedia

Deep Q-Learning (DQL)

$$\text{Argmin}_{\theta} \left[Q(s,a;\theta) - \left(r(s,a) + \gamma \max_{a} Q(s',a;\theta) \right) \right]^2$$

Update this

To be closer to new value estimate

Represent *Q* with a neural net

***s, a* can be one-hot or real valued**

Model Interpretation

- **Global: What does this *model* do?**
  - Variable importance
    - E.g. how much does accuracy suffer if you remove it?
- **Local: Why did you make this decision for this *x*?**
  - **Decision tree**: path taken
  - **LIME** (Local Interpretable Model-Agnostic Explanations)

What to use when? - Supervised

n	p	
100	100,000	PCR, Ridge, SVM, semi-supervised
10,000	10,000	Anything in Sk-learn
100,000	100	Sk-learn, Nnet
1,000,000	10,000	Nnet
1,000,000	100,000	Nnet with pretraining

Ask: What structure do I expect to see?

What to use when? - Unsupervised

n	p	
100	100,000	K-means or PCA
10,000	10,000	K-means or PCA
100,000	10,000	K-means? GMM? Autoencoder?
10,000,000	images	
100,000,000	words	

Feature selection

- **Regression**
  - Do you expect very few, a moderate number of, or most features?
- **Random forests, gradient tree bosting**
  - Feature selection is ‘built in’
- **Neural nets**
  - Generally no feature selection
  - Or screen features before you build the net

The biggest open problem: **Transfer learning**

- **Embeddings**
  - Images, words, products, people
  - Multimodal transfomers …
- **Multitask learning**
  - Shared embeddings for many tasks
- **One shot learning**
    - Generalize from one label on one image …

The biggest open problem: **Transfer learning**

- **Learn "deep structure"**
  - Reusable modules
- **Build in "deep structure"**
  - Physics
  - Ties to symbolic reasoning?
  - Causal models
- **Use external data**
  - Wikipedia, databases, …

The future …

- Every company is an AI company
- Self-driving cars?
- The singularity??

- See many of you for the review session, Monday 10:30
- See all of you for the final, Wednesday
- Stay in touch & let me know how you use ML …

Thank you!!!