CSC 580 Principles of Machine Learning

10 Probabilistic graphical models: Naïve Bayes

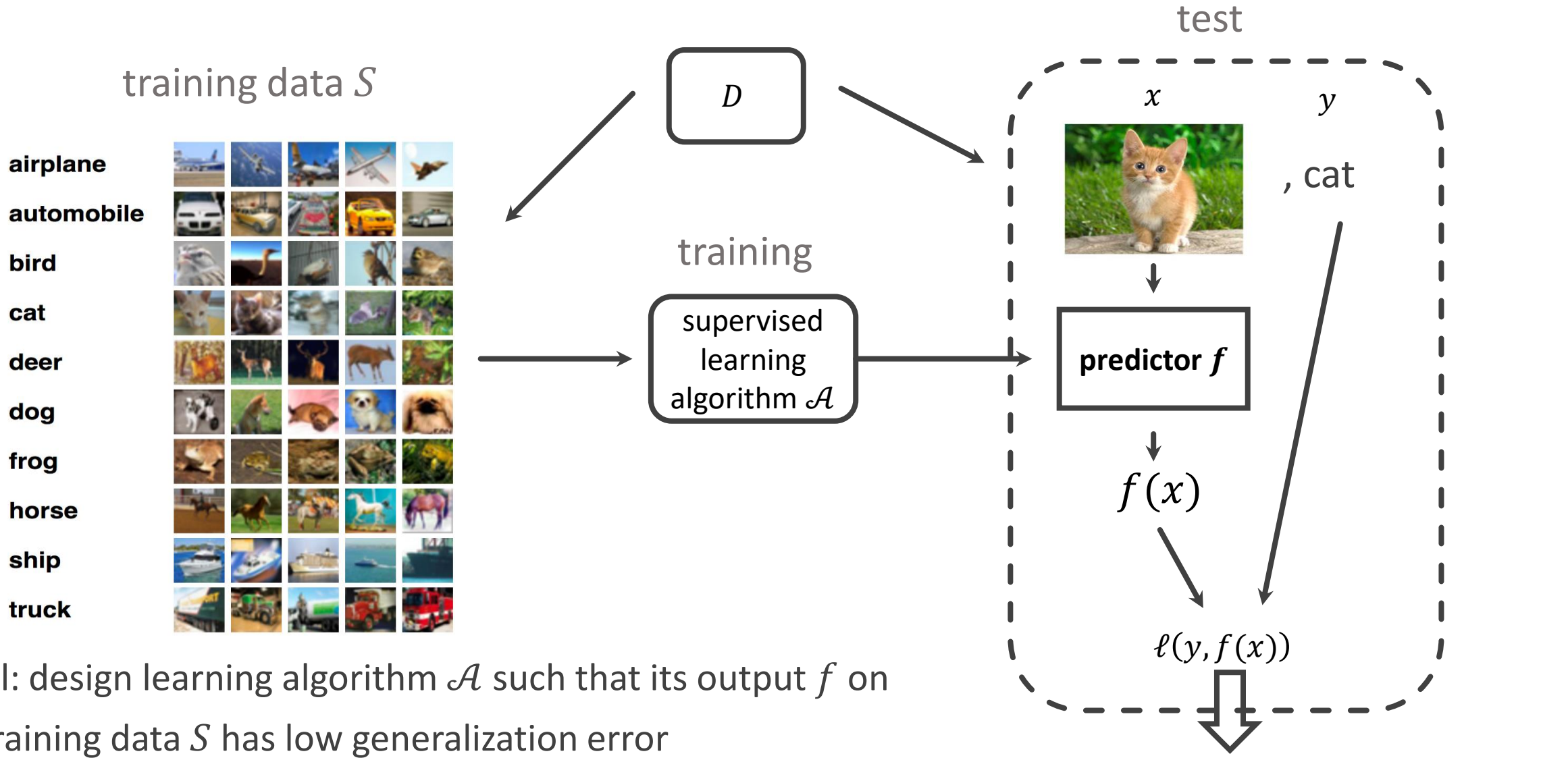
Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun

Recap: supervised learning setup

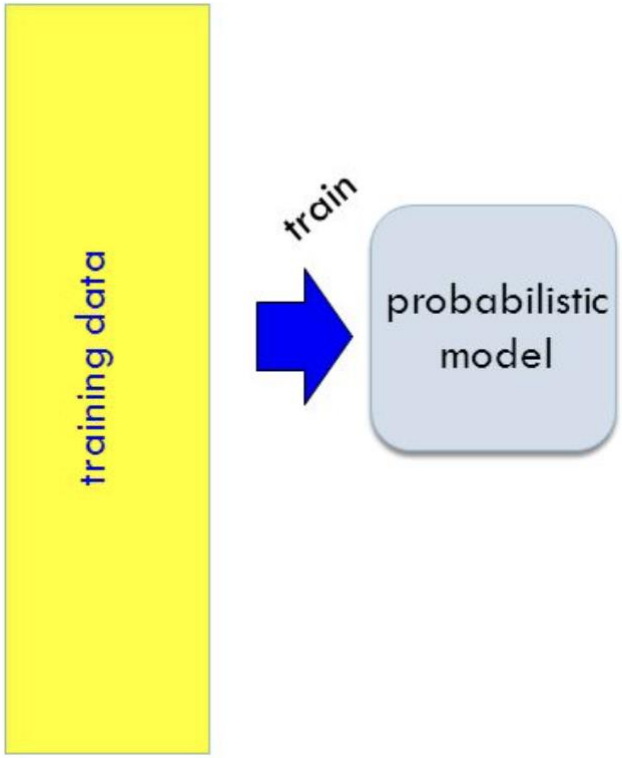


- Goal: design learning algorithm $\mathcal{A}$ such that its output f on iid training data S has low generalization error

Generalization error: $L_D(f) = \mathrm{E}_{(x,y)\sim D}\, \ell(y, f(x))$

Probabilistic modeling

- A systematic approach for machine learning
- Steps:
  1. Model how the data is generated by probabilistic models, but with parameters unspecified (modeling assumption / generative story)
     - Each example $z \sim P(z; \theta)$ for some $\theta \in \Theta$
       - For $z = (x, y)$ => supervised learning
       - For $z = x$ => unsupervised learning
  2. (Training) Learn the model parameter $\hat{\theta}$
     - Important example: maximum likelihood estimation (MLE), i.e., $\text{maximize}_{\theta \in \Theta} \log P(z_1, \ldots, z_n; \theta)$
  3. (Test) Make prediction / decision based on the learned model $P(z; \hat{\theta})$
     - Important example: predict using the Bayes classifier of $P(z; \hat{\theta})$ (for supervised learning)



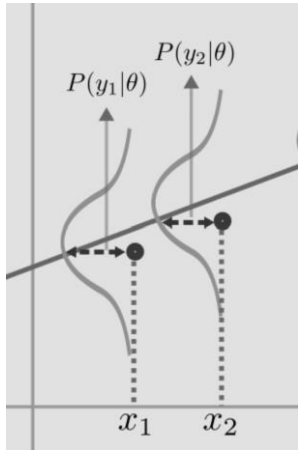
http://slideplayer.com/slide/4527958/

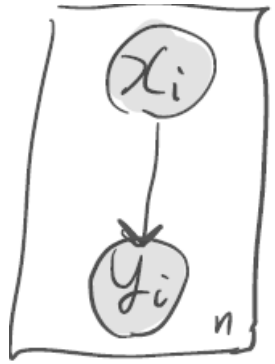
Probabilistic modeling (cont'd)

- Why probabilistic modeling?
  - Right thing to do if the model is correct
  - If not…
    - "All models are wrong, but some are useful" -- George E. P. Box
  - Interpretability
  - A view taken by classical statistics

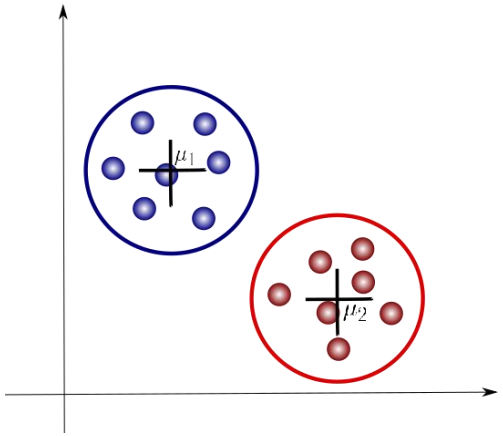
Discriminative vs Generative modeling

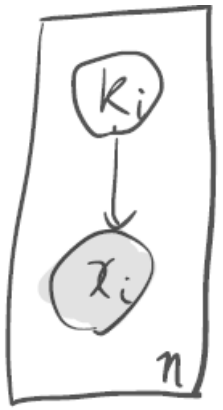
- Discriminative model: models only $P(y \mid x)$
  - Recall linear regression: $y \mid x; \theta \sim N(x^\top \theta, \sigma^2)$
  - Logistic regression: $y \mid x; \theta \sim \text{Bernoulli}(\sigma(x^\top \theta))$





- Generative model for unsupervised learning: models $P(x)$
  - e.g., Gaussian mixture model (GMM)
    - $\theta = (\pi_k, \mu_k, \Sigma_k)_{k=1}^K$
    - $k \sim \text{Categorical}(\pi)$ (*hidden*), i.e. $P(k = l) = \pi_l$
    - $x \mid k \sim N(\mu_k, \Sigma_k)$

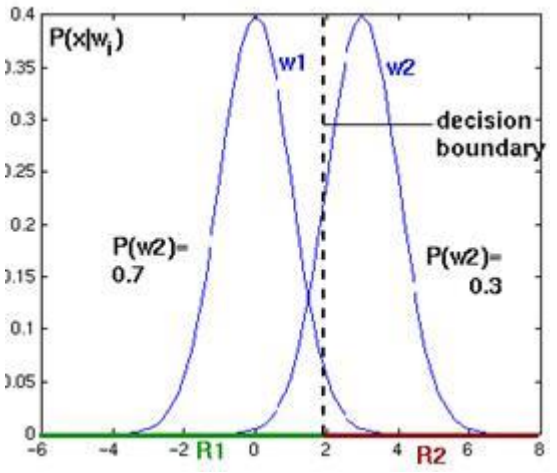


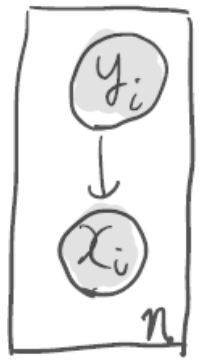


https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95
https://suriyadeepan.github.io/2017-01-22-mle-linear-regression/

Discriminative vs Generative modeling (cont'd)

- Generative model for supervised learning:
  - models 'class-conditional' distribution $P(x \mid y)$
  - E.g. $\theta = (\pi_c, \mu_c, \Sigma_c)_{c=1}^{C}$,
  - $y \sim \text{Categorical}(\pi)$, $x \mid y = c \sim N(x; \mu_c, \Sigma_c)$



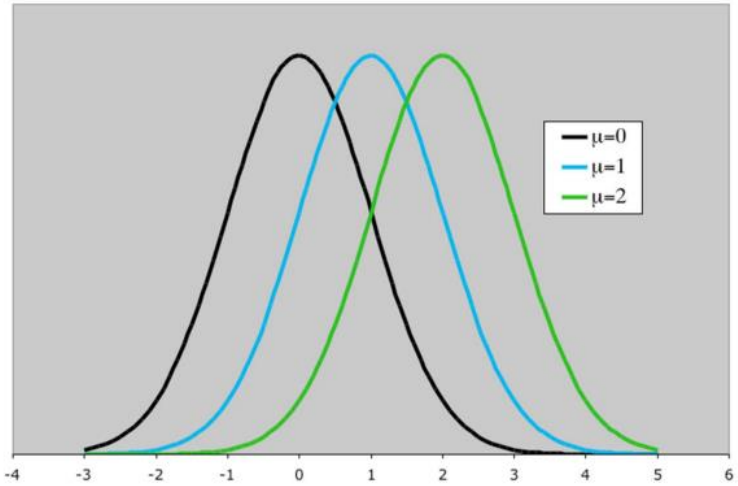


- Given test data from $P(x, y; \theta)$, what is the optimal classification rule?
  - $f_{BO,\theta}(x) = \text{argmax}_c P(y = c \mid x; \theta)$
  - Recall Bayes formula: $P(y = c \mid x) = \frac{P(x|y=c)P(y=c)}{P(x)}$, where $P(x) = \sum_c P(x \mid y = c)P(y = c)$
  - Therefore, $f_{BO,\theta}(x) = \text{argmax}_c \, P(x \mid y = c; \theta)P(y = c; \theta)$

https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

Generative model: basic example I

- **Wait time at the barbershop**: Suppose you go to a barbershop at every last Friday of the month. You want to be able to predict the waiting time. You have collected 12 data points (i.e., how long it took to be served) from the last year: $S = \{x_1, \dots, x_{12}\}$
- 1. Modeling assumption: $x_i \sim$ Gaussian distribution $N(\mu, 1)$
  - $p(x; \mu) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2}\right)$
  - Observation: this distribution has mean μ



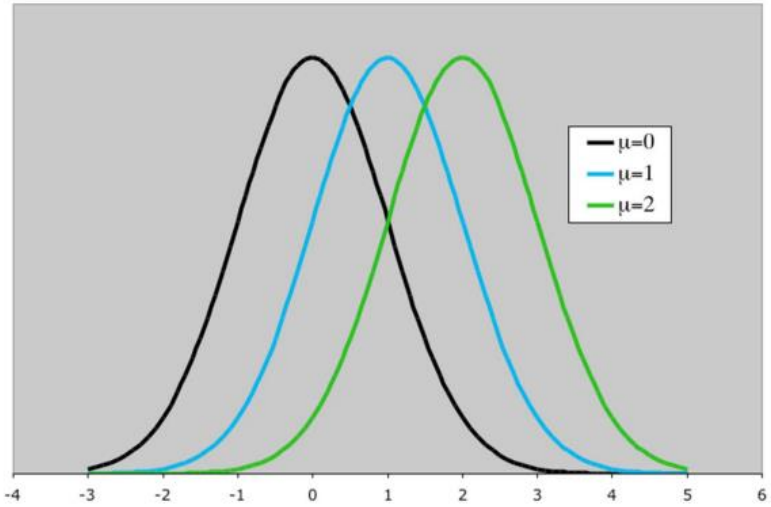
- 2. Find the MLE $\hat{\mu}$ from data S
  - (2.1) write down the neg. log likelihood of the sample

$$L_n(\mu) = -\ln P(x_1, \dots, x_n; \mu) = 12 \ln\sqrt{2\pi} + \frac{1}{2}\sum_{i=1}^{12}(x_i - \mu)^2$$

Generative model: basic example I (cont'd)

- 2. Find the MLE $\hat{\mu}$ from data S
  - (2.2) compute the first derivative, set it to 0, solve for λ (be sure to check convexity)

$$L_n'(\mu) = \sum_{i=1}^{12}(x_i - \mu) = 0 \Rightarrow \mu = \frac{x_1 + \cdots x_{12}}{12}$$



- 3. The learned model $N(\hat{\mu}, 1)$ is yours!
  - Simple prediction: e.g., predict the next wait time by $\mathbb{E}_{X \sim N(\hat{\mu},1)}[X]$
  - which is $\hat{\mu} = \frac{x_1 + \cdots x_{12}}{12}$

Generative modeling: basic example II

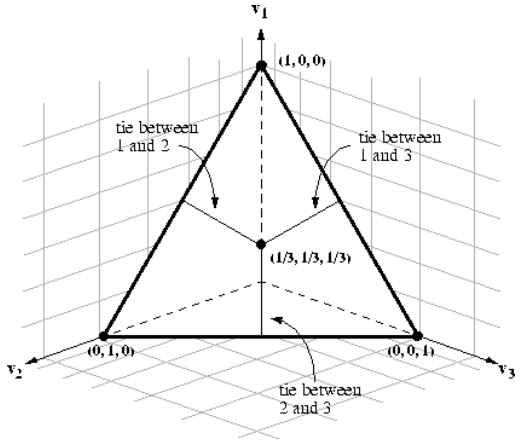
- [Data] $S = \{y_i\}_{i=1}^n$, where $y_i \in \{1, \dots, C\}$

- [Generative story]

  $y \sim \text{Categorical}(\pi)$, where $\pi = (\pi_1, \dots, \pi_C) \in \Delta^{C-1}$ ($\pi_c \geq 0$ and $\pi_1 + \cdots + \pi_C = 1$)

  e.g. y_i = the color of i-th ball drawn randomly from a bin (with replacement)

  $p(y; \pi) = \pi_y \left(= \prod_{c=1}^C \pi_c^{I(y=c)} \right)$



- [Training]

  (2.1) $L_n(\pi) = -\ln P(y_1, \dots, y_n; \pi) = \sum_{i=1}^n -\ln \pi_{y_i} = -\sum_{c=1}^C n_c \ln \pi_c$,

  where $n_c = \#\{i : y_i = c\} = \sum_{i=1}^n I(y_i = c)$

Generative modeling: basic example II (cont'd)

- [Training]

  (2.2) $\text{minimize}_{\pi \in \Delta^{C-1}} L_n(\pi) \coloneqq -\sum_{c=1}^{C} n_c \ln \pi_c$

  Constrained maximization problem; solve by Lagrange multipliers

$$\frac{\partial}{\partial \pi}\left(-\sum_{c=1}^{C} n_c \ln \pi_c - \lambda\left(\sum_{c=1}^{C} \pi_c - 1\right)\right) = -\frac{n_c}{\pi_c} - \lambda = 0 \Rightarrow \pi_c = -\frac{n_c}{\lambda}$$

Combined with the constraint that $\pi_1 + \cdots + \pi_C = 1 \Rightarrow \hat{\pi}_c = \frac{n_c}{n}$, for all c

- [Test] predict label $\text{argmax}_c P(y = c; \hat{\pi}) = \text{argmax}_c \hat{\pi}_c$

Naïve Bayes for supervised learning

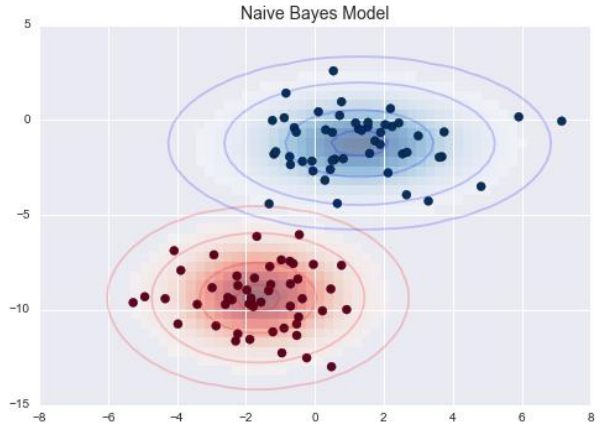
- Motivation: supervised learning for classification
- high-dimensional $x = (x(1), \dots, x(F))$, modeling $P(x \mid y)$ can be tricky
- In general, $P(x \mid y) = P(x(1) \mid y) \cdot P(x(2) \mid x(1), y) \cdot \dots \cdot P(x(F) \mid x(1), \dots, x(F-1), y)$

- A modeling assumption: $x(1), \dots, x(F)$ are conditionally independent given y

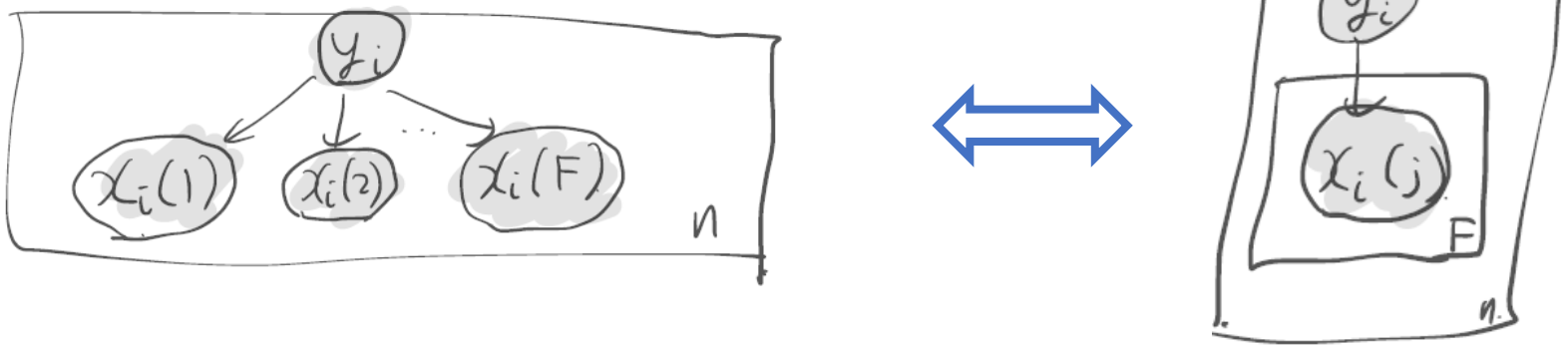
  i.e. for all i

$$x(i) \perp\!\!\!\perp \big(x(1), \dots, x(i-1), x(i+1), \dots, x(F)\big) \mid y$$

  (Conditional independence notation: $A \perp\!\!\!\perp B \mid C$)



- Equivalently $P(x \mid y) = P(x(1) \mid y) \cdot \dots P(x(F) \mid y)$



Naïve Bayes: binary-valued features

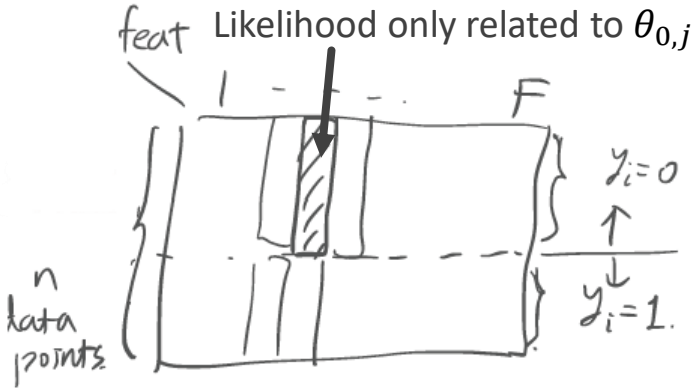
(recall the course recommendation example)

- [Data] $S = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \{0,1\}^F$ $y_i \in \{0,1\}$
- [Generative story]

  $y \sim \text{Bernoulli}(\pi)$; for all $j \in [F]$, $x(j) \mid y = c \sim \text{Bernoulli}(\theta_{c,j})$

  #parameters = $1 + 2F$

  [Training] (denote by $\theta = \{\theta_{c,j}\}$)



$$\max_{\pi,\theta} \sum_{i=1}^n \ln P(x_i, y_i; \pi, \theta) = \sum_{i=1}^n \ln P(y_i; \pi) + \sum_{i=1}^n \ln P(x_i \mid y_i; \theta)$$

$$= \max_{\pi} \sum_{i=1}^n \ln P(y_i; \pi) + \max_{\{\theta_{0,j}\}} \sum_{i:y_i=0} \ln P(x_i \mid y_i; \theta) + \max_{\{\theta_{1,j}\}} \sum_{i:y_i=1} \ln P(x_i \mid y_i; \theta)$$

Key observation: optimal π, optimal $\{\theta_{0,j}\}$, optimal $\{\theta_{1,j}\}$ can be found separately

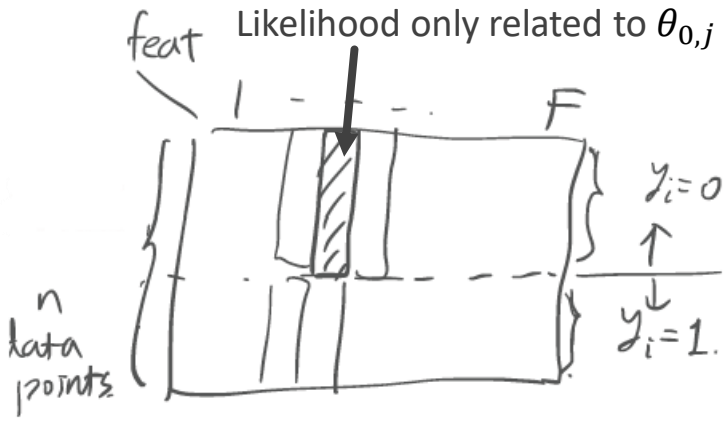
Optimal π: $\max_{\pi} \sum_{i=1}^n \ln P(y_i; \pi) = \max_{\pi} n_0 \ln(1-\pi) + n_1 \ln(\pi)$ => $\hat{\pi} = \frac{n_1}{n}$

Naïve Bayes: binary-valued features (cont'd)

- By the Naïve Bayes modeling assumption,

$$\max_{\{\theta_{0,j}\}} \sum_{i:y_i=0} \ln P(x_i \mid y_i;\theta) = \max_{\{\theta_{0,j}\}} \sum_{j=1}^{F} \sum_{i:y_i=0} \ln P\big(x_i(j) \mid y_i;\theta_{0,j}\big)$$

$$= \sum_{j=1}^{F} \max_{\theta_{0,j}} \sum_{i:y_i=0} \ln P\big(x_i(j) \mid y_i;\theta_{0,j}\big)$$



- Again, can optimize each $\theta_{0,j}$ separately

- Optimal $\theta_{0,j}$:

$$\max_{\theta_{0,j}} \sum_{i:y_i=0,\, x_i(j)=1} \ln \theta_{0,j} + \sum_{i:y_i=0,\, x_i(j)=0} \ln (1-\theta_{0,j})$$

- $\hat{\theta}_{0,j} = \frac{\#\{i:\, y_i=0,\, x_i(j)=1\}}{\#\{i:y_i=0\}}$; similarly, $\hat{\theta}_{1,j} = \frac{\#\{i:\, y_i=1,\, x_i(j)=1\}}{\#\{i:y_i=1\}}$

Naïve Bayes: binary-valued features (cont'd)

[Test] Given $\hat{\pi}$, $\{\hat{\theta}_{c,j}\}$, Bayes optimal classifier

$$\hat{f}_{BO}(x) = \text{argmax}_y \, P(x, y; \hat{\pi}, \{\hat{\theta}_{c,j}\}) = \text{argmax}_y \log P(x, y; \hat{\pi}, \{\hat{\theta}_{c,j}\})$$

- $\log P(x, y = 0; \pi, \{\theta_{c,j}\}) = \ln(1 - \pi) + \sum_{j=1}^{F} \ln P\big(x(j) \mid y; \theta_{0,j}\big)$

  $= \ln(1 - \pi) + \sum_{j=1}^{F} \ln\big(1 - \theta_{0,j}\big) I(x(j) = 0) + \ln\big(\theta_{0,j}\big) I(x(j) = 1)$

  $= \ln(1 - \pi) + \sum_{j=1}^{F} \ln(1 - \theta_{0,j}) + \sum_{j=1}^{F} x(j) \ln \frac{\theta_{0,j}}{1-\theta_{0,j}}$

- Similarly, $\log P(x, y = 1; \pi, \{\theta_{c,j}\}) = \ln(\pi) + \sum_{j=1}^{F} \ln(1 - \theta_{1,j}) + \sum_{j=1}^{F} x(j) \ln \frac{\theta_{1,j}}{1-\theta_{1,j}}$
- Therefore, $\hat{f}_{BO}(x) = 1 \Leftrightarrow \underbrace{\ln\left(\frac{\pi}{1-\pi}\right) + \sum_{j=1}^{F} \ln\left(\frac{1-\theta_{1,j}}{1-\theta_{0,j}}\right)}_{b} + \sum_{j=1}^{F} x(j) \underbrace{\left(\ln \frac{\theta_{1,j}}{1-\theta_{1,j}} - \ln \frac{\theta_{0,j}}{1-\theta_{0,j}}\right)}_{w(j)} \geq 0$
- I.e. Bayes classifier is *linear*

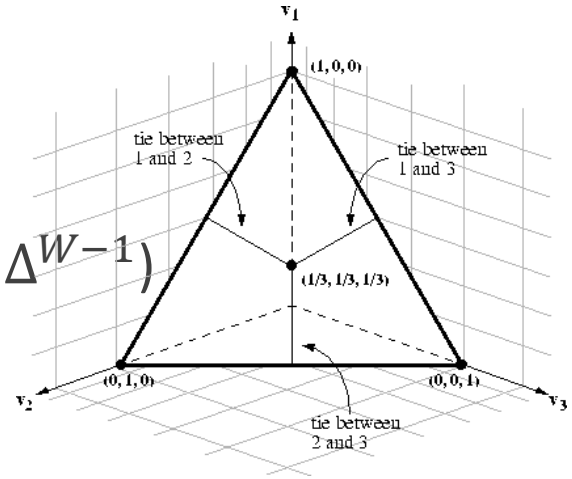
Naïve Bayes: Discrete (Categorical-valued) features

- [Data] $S = \{(x_i, y_i)\}_{i=1}^{n}$, $x_i \in [W]^F$ $y_i \in \{0,1\}$
- [Generative story]

  $y \sim \text{Bernoulli}(\pi)$; for all $j \in [F]$, $x(j) \mid y = c \sim \text{Categorical}(\theta_c)$ $(\theta_c \in \Delta^{W-1})$

  #parameters = $1 + 2W$

  Note: in this example, θ_c *shared across all features*!

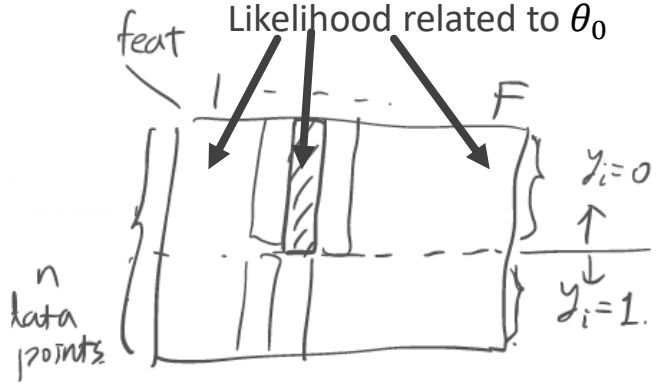


- [Training]

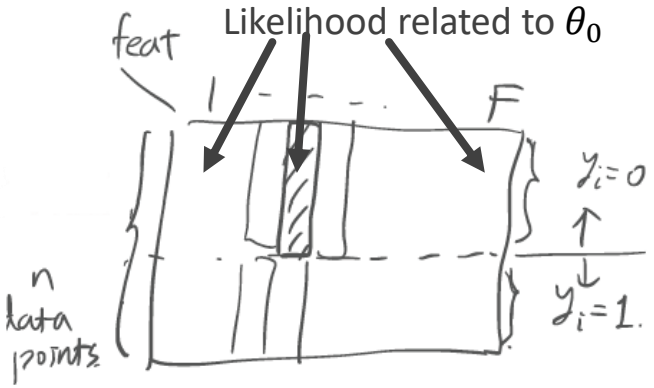
  Similar to previous example, optimal π, optimal θ_0, optimal θ_1 can be found separately,

  by maximizing the respective part of the likelihood function (exercise)

  Optimal π same as previous example



Naïve Bayes: Discrete features (cont'd)



- [Training]

  Optimal θ_c:

$$\max_{\theta_0} \sum_{i:y_i=0} \ln P(x_i \mid y_i; \theta_0) = \max_{\theta_0} \sum_{j=1}^{F} \sum_{i:y_i=0} \ln P(x_i(j) \mid y_i; \theta_0)$$

$$= \max_{\theta_0} \sum_{w=1}^{W} \sum_{j=1}^{F} \sum_{i:y_i=0} I(x_i(j) = w) \ln \theta_{0,w}$$

$$= \max_{\theta_0} \sum_{w=1}^{W} \ln \theta_{0,w} \, \#\{(i,j): y_i = 0, x_i(j) = w\}$$

$$\Rightarrow \hat{\theta}_{c,w} = \frac{\#\{(i,j): y_i=c, x_i(j)=w\}}{\#\{i: y_i=c\} \times F}$$

  Exercise: how to extend this to variable-length x_i's (e.g. for text classification)?

- [Test] Bayes optimal classification rule with $(\hat{\pi}, \hat{\theta}_0, \hat{\theta}_1)$ (exercise)

Probabilistic modeling: quick summary

- Now you see the flow:

  (1) Specify the generative story

  (2) design the maximum likelihood estimator

  => gives you a natural loss function for training

  (3) use the estimated model to make decisions

- Naïve Bayes: can mix different distributions for different features: $x(j) \mid y = c$
  - Bernoulli
  - Categorical
  - Continuous distributions (next)

Next lecture (10/17)

- Naïve Bayes with continuous feature distributions
- Midterm review
- Reading: CIML 9.4-9.7

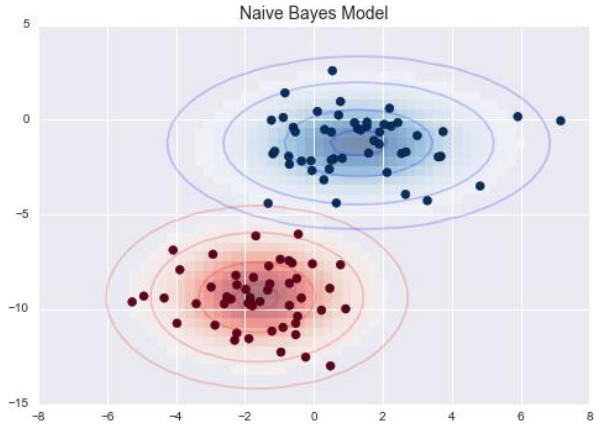
Gaussian naïve Bayes

- [Generative story]

  $y \sim \text{Bernoulli}(\pi)$;

  for all $j \in [F]$, $x(j) \mid y = c \sim N(\mu_{c,j}, \sigma^2)$ with σ known

  #parameters = $1 + 2F$



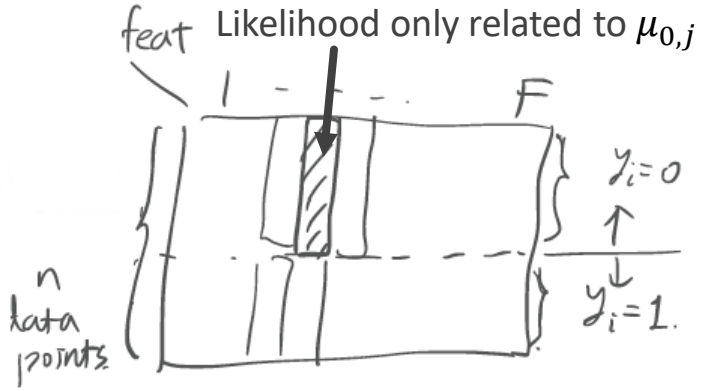
- [Training]

  Again,

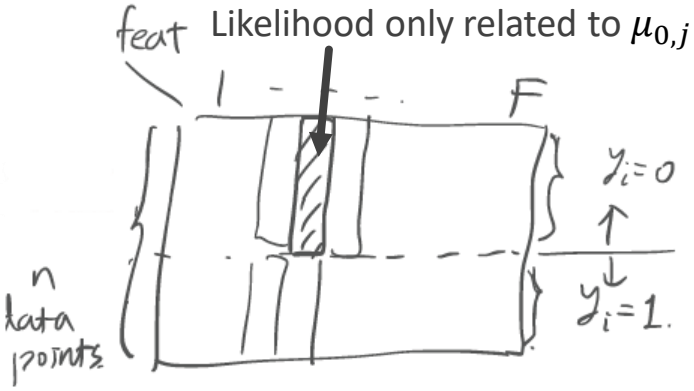
  - the parts of the likelihood function related to π and $\mu_{c,j}$'s are disjoint

  - same formula for optimal π

  For optimal $\mu_{c,j}$: $\max_{\mu_{c,j}} \sum_{i:y_i=c} \ln P\left(x_i(j) \mid y_i; \mu_{c,j}\right)$



Gaussian naïve Bayes (cont'd)



$$\underset{\mu_{c,j}}{\text{maximize}} \sum_{i:y_i=c} \ln P\left(x_i(j) \mid y_i; \mu_{c,j}\right)$$

$x(j) \mid y = c \sim N(\mu_{c,j}, \sigma^2)$ => the above maximization is equivalent to

$$\underset{\mu_{c,j}}{\text{maximize}} \sum_{i:y_i=c} -\frac{1}{2\sigma^2}\left(x_i(j) - \mu_{c,j}\right)^2$$

=> Optimal $\mu_{c,j} = \frac{\sum_{i:y_i=c} x_i(j)}{\#\{i:y_i=c\}}$, for all $c \in \{0,1\}, j \in [F]$

Gaussian naïve Bayes (cont'd)

[Test] Given $\hat{\pi}$, $\{\hat{\theta}_{c,j}\}$, Bayes optimal classifier

$$\hat{f}_{BO}(x) = \text{argmax}_y \log P(x, y; \hat{\pi}, \{\hat{\theta}_{c,j}\})$$

- $\log P(x, y = 0; \pi, \{\mu_{c,j}\}) = \ln(1 - \pi) + \sum_{j=1}^{F} \ln P\left(x(j) \mid y; \mu_{0,j}\right)$

$$= \ln(1 - \pi) - \frac{1}{2\sigma^2} \|x - \mu_0\|_2^2 - F \cdot \ln(\sqrt{2\pi}\sigma)$$

- Similarly,

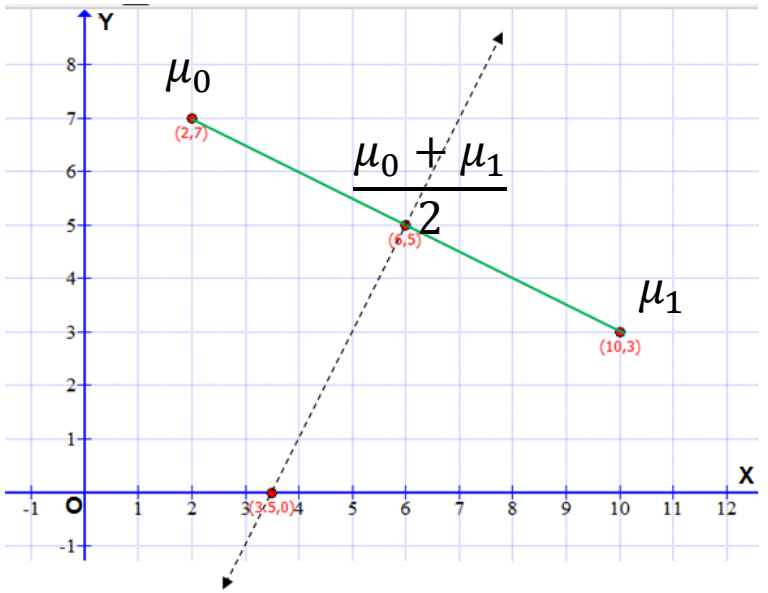
$$\log P(x, y = 1; \pi, \{\mu_{c,j}\}) = \ln(\pi) - \frac{1}{2\sigma^2} \|x - \mu_1\|_2^2 - F \cdot \ln(\sqrt{2\pi}\sigma)$$

Gaussian naïve Bayes (cont'd)

- Therefore,

$$\hat{f}_{BO}(x) = 1 \Leftrightarrow \ln\left(\frac{\pi}{1-\pi}\right) - \frac{1}{2\sigma^2}\|x-\mu_1\|_2^2 + \frac{1}{2\sigma^2}\|x-\mu_0\|_2^2 \geq 0$$

- Special case $\pi = 0.5$: $\hat{f}_{BO}(x) = 1 \Leftrightarrow \|x-\mu_1\|_2 \leq \|x-\mu_0\|_2$
  i.e. the Bayes decision boundary is the perpendicular bisector of the class center



- For general π,

$$\hat{f}_{BO}(x) = 1 \Leftrightarrow 2\sigma^2 \ln\left(\frac{\pi}{1-\pi}\right) + \left\langle \mu_1 - \mu_0, x - \frac{\mu_1+\mu_0}{2} \right\rangle \geq 0$$

- See book for a variant of this model with unknown conditional variance

Summary

- Generative vs Discriminative model
  - “When solving a problem of interest, do not solve a more general problem as an intermediate step” – Vladimir Vapnik (one of the founders of statistical learning)

- Naïve bayes: the most basic probabilistic graphical model (PGM)
  - good for high-dimensional data; the bias helps reducing variance.

Midterm review

General suggestions

- Prioritize reviewing basic concepts & ideas over technical details
  - Remember, you can bring a cheatsheet with necessary technical details
  - Understand the motivations of concepts / algorithm design
  - “Math should be there to aid understanding, not hinder it.” – CIML book

- “Memorization with understanding”

Lec 1: supervised learning; decision trees

- The supervised learning paradigm
  - Training data, test data, data distributions, classifiers
  - Train error, generalization error

- Decision trees
  - Prediction
  - Training algorithm: top-down
  - Label uncertainty reduction given (feature f, dataset S)

$$\text{Score}(S, f) = u(S) - \left(p_L\, u(S_L) + p_R\, u(S_R)\right)$$

Able to represent everything above using (conditional) probability notations under P_S, the uniform distribution over S

Lec 2: Limits of learning

- Given a distribution D, calculate the Bayes optimal classifier f_{BO}
- Calculate $L_D(f)$, the error rate of any classifier f on D, specifically $L_D(f_{BO})$

- Explain the contributing factors of $L_D(f) > 0$ in practical scenarios
- Explain underfitting / overfitting using specific algorithms / datasets as examples

- Using law of large numbers to explain
  - "test set error approximates generalization error"
  - "never touch your test data"
- Hyperparameter tuning: hold-out set method / K-fold cross validation
- Understand "inductive bias" at an intuitive level

Lec 3: Nearest neighbors

- Inductive bias of nearest neighbor methods
- Able to calculate: distances between points, and nearest neighbors of a point, and nearest neighbor classification results
- Hyperparameter in nearest neighbor methods – k
  - Smaller k => underfitting or overfitting?
- Time complexity analysis
- Drawback of nearest neighbors: feature scaling and irrelevant features

Lec 4: Linear classification and Perceptron

- Linear classifiers: homogeneous vs. nohomogeneous
  - Geometric interpretation of linear classifiers;
  - Margin of labeled examples w.r.t. a linear classifier, and the relationship to example's distance to decision boundary;

- The Perceptron algorithm
  - Able to explain why Perceptron works (using the update rule)
  - Perceptron convergence theorem: understand its intuition – converges faster with the dataset margin increasing
  - Perceptron variants: voting Perceptron, averaged Perceptron

- Limitations of linear classification: XOR problem and how to address it

Lec 5: Practical considerations

- Understand how feature normalization and transformation help with learning tasks

- Refined classification performance measures:
  - confusion matrix,
  - TP, TN, FP, FN, TPR, …
  - Precision, recall, F1 score
  - Given real-valued decision scores, understand how ROC and Precision-Recall Curve are drawn
  - Able to calculate AUROC

- Confidence interval and hypothesis testing
  - Using Gaussian approximation for confidence intervals, based on quantiles of t-distribution
  - Paired t-test
  - Review HW2 problem 4

Lec 6: Linear models and convexity

- Linear regression
  - Regression setup – square loss, regression function, Bayes risk
  - Use the chain rule of derivatives to calculate the gradient of general loss functions
  - Regularization for linear regression: ridge, LASSO
  - OLS solution; ridge regression solution
  - Maximum likelihood viewpoint of OLS

- Convexity
  - Defns of convex sets, convex functions
  - Check convexity: using defn / basic properties / second derivative checks
  - Able to check convexity of losses used in ML applications
  - Convex functions f are "nice": any stationary point (w, s.t. $\nabla f(w) = 0$) is a global optimum

Lec 7: Linear models for classification

- Logistic regression: loss minimization viewpoint & maximum likelihood viewpoint
  - Convexity of logistic losses
- SVM: hard margin and soft-margin formulations
  - Equivalent formulation: regularized hinge loss minimization

- Optimization methods:
  - Gradient descent
  - Stochastic (sub)gradient descent
  - Able to derive update formula with specific loss functions & datasets
  - Able to explain their pros and cons

- The dual of SVM; geometric interpretation of support vectors using KKT conditions
  - See HW2 Problem 5

Lec 8: kernel methods

- Motivation: addressing the computational challenge of linear methods with a high-dimensional feature map

- Kernelized representation of linear models: when does it make prediction more efficient than explicit linear coefficient representation?

$$\text{e.g. } \operatorname{sign}(\langle w^*, \phi(x)\rangle + b^*) = \operatorname{sign}(\textstyle\sum_i \alpha_i^* y_i K(x_i, x) + b^*)$$

- Kernelized training algorithms: SVM, Perceptron, ridge regression

- How to prove / disprove that a function $K(x, y)$ is a valid kernel (i.e. exists feature map ϕ s.t. $K(x, y) = \langle \phi(x), \phi(y)\rangle$)
  - Prove: exhibit ϕ (also review: building complex kernels from simpler ones)
  - Disprove: invalidate symmetry; invalidate Mercer's condition

Time complexity of (kernelized) Perceptron & ridge regression

- ϕ: feature map from d dimensions to p dimensions
- n: training set size
- k: the number of operations to evaluate $K(x, x')$

- Test stage:

	Without kernel trick	With Kernel trick
Ridge regression	$O(p)$	$O(nk)$
Perceptron	$O(p)$	$O(nk)$

- Training stage:

	Without kernel trick	With Kernel Trick
Ridge regression	$O(np^2 + p^3)$	$O(n^2k + n^3)$
Perceptron	$O(\#\text{iters} \times p)$	$O(\#\text{iters} \times nk)$