CPSC 540: Machine Learning

Valid Kernels, Fench Duality, Density Estimation

Mark Schmidt

University of British Columbia

Winter 2016

Admin

- Assignment 1:
  - Solutions Posted.
- Assignment 2:
  - Due today.
- Assignment 3:
  - Coming soon: a bit shorter and due Feb 23.
- Extra late days:
  - To give possibility of two week-long extensions, allowing 4 late days.
  - But a maximum of 3 late days on any single assignment.
  - And you can still only use 1 late day on A4.
  - You can use late days on the project too.

Coordinate Optimization vs. Stochastic Gradient

- Consider optimization problem:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

- Coordinate optimization: update one x_j based on all examples:
  - Fast convergence rate, but iterations must be d times cheaper than gradient method.
  - Functions f_i must be smooth.

Coordinate Optimization vs. Stochastic Gradient

- Consider optimization problem:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

- Coordinate optimization: update one x_j based on all examples:
    - Fast convergence rate, but iterations must be d times cheaper than gradient method.
    - Functions f_i must be smooth.
- Stochastic gradient: update all x_i based on one example:
    - Slow convergence rate, and iterations are d times cheaper than gradient method.
    - Functions f_i can be non-smooth.

Coordinate Optimization vs. Stochastic Gradient

- Consider optimization problem:

$$\underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

- Coordinate optimization: update one x_j based on all examples:
  - Fast convergence rate, but iterations must be d times cheaper than gradient method.
  - Functions f_i must be smooth.
- Stochastic gradient: update all x_i based on one example:
  - Slow convergence rate, and iterations are d times cheaper than gradient method.
  - Functions f_i can be non-smooth.
- SAG: update all x_i based on one example (and old versions of others:
  - Fast convergence rate, and iterations are d times cheaper than gradient method.
  - Functions f_i must be smooth.

Last Time: Kernel Trick

- Suppose we have 2 features ($d = 2$).
- Our usual L2-regularized least squares estimate is

$$w = (X^T X + \lambda I_2)^{-1} X^T y.$$

Last Time: Kernel Trick

- Suppose we have 2 features ($d = 2$).
- Our usual L2-regularized least squares estimate is

$$w = (X^T X + \lambda I_2)^{-1} X^T y.$$

- If we instead use a quadratic polynomial basis,

$$\phi(x_i) = \begin{bmatrix} 1 & 2x_{i1} & 2x_{i2} & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 \end{bmatrix},$$

  then our usual MAP estimate is

$$w = (\Phi(X)^T \Phi(X) + \lambda I_6)^{-1} \Phi(X)^T y,$$

  where $\Phi(X)$ has $\phi(x_i)$ as its rows.

Last Time: Kernel Trick

- Suppose we have 2 features ($d = 2$).
- Our usual L2-regularized least squares estimate is

$$w = (X^T X + \lambda I_2)^{-1} X^T y.$$

- If we instead use a quadratic polynomial basis,

$$\phi(x_i) = \begin{bmatrix} 1 & 2x_{i1} & 2x_{i2} & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 \end{bmatrix},$$

  then our usual MAP estimate is

$$w = (\Phi(X)^T \Phi(X) + \lambda I_6)^{-1} \Phi(X)^T y,$$

  where $\Phi(X)$ has $\phi(x_i)$ as its rows.
- This assumes we can compute $\Phi(X)$.

Last Time: Kernel Trick

- We ultimately make predictions using

$$\begin{aligned}
\hat{y} &= \Phi(\hat{X})w \\
&= \Phi(\hat{X})\underbrace{(\Phi(X)^T\Phi(X) + \lambda I_6)^{-1}\Phi(X)^T y}_{w} \\
&= \underbrace{\Phi(\hat{X})\Phi(X)^T}_{\hat{K}}(\underbrace{\Phi(X)\Phi(X)^T}_{K} + \lambda I_n)^{-1} y \\
&= \hat{K}(K + \lambda I_n)^{-1} y.
\end{aligned}$$

Last Time: Kernel Trick

- We ultimately make predictions using

$$\begin{aligned}
\hat{y} &= \Phi(\hat{X})w \\
&= \Phi(\hat{X})\underbrace{(\Phi(X)^T\Phi(X) + \lambda I_6)^{-1}\Phi(X)^T y}_{w} \\
&= \underbrace{\Phi(\hat{X})\Phi(X)^T}_{\hat{K}}(\underbrace{\Phi(X)\Phi(X)^T}_{K} + \lambda I_n)^{-1}y \\
&= \hat{K}(K + \lambda I_n)^{-1}y.
\end{aligned}$$

- Kernel trick:
  - We have kernel function $k(x_i, x_j)$ that gives element (i, j) of K or $\hat{K}$.
    - For quadratic polynomials we have $k(x_i, x_j) = (1 + x_i^T x_j)^2$.
  - Skip forming $\Phi(X)$ and directly form K and $\hat{K}$.
  - Size of K is n by n even if $\Phi(X)$ has exponential or infinite columns.

Guasian-RBF Kernels

- The most common kernel is the Gaussian-RBF (or 'squared exponential') kernel,

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right).$$

- What function $\phi(x)$ would lead to this as the inner-product?

Guasian-RBF Kernels

- The most common kernel is the Gaussian-RBF (or 'squared exponential') kernel,

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right).$$

- What function $\phi(x)$ would lead to this as the inner-product?
  - To simplify, assume $d = 1$ and $\sigma = 1$,

$$\begin{aligned} k(x_i, x_j) &= \exp(-x_i^2 + 2x_i x_j - x_j^2) \\ &= \exp(-x_i^2)\exp(2x_i x_j)\exp(-x_j^2), \end{aligned}$$

Guasian-RBF Kernels

- The most common kernel is the Gaussian-RBF (or 'squared exponential') kernel,

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right).$$

- What function $\phi(x)$ would lead to this as the inner-product?
  - To simplify, assume $d = 1$ and $\sigma = 1$,

$$\begin{aligned} k(x_i, x_j) &= \exp(-x_i^2 + 2x_i x_j - x_j^2) \\ &= \exp(-x_i^2)\exp(2x_i x_j)\exp(-x_j^2), \end{aligned}$$

  so we need $\phi(x_i) = \exp(-x_i^2) z_i$ where $z_i z_j = \exp(2x_i x_j)$.
    - For this to work for *all* x_i and x_j, z_i must be infinite-dimensional.

Guasian-RBF Kernels

- The most common kernel is the Gaussian-RBF (or 'squared exponential') kernel,

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right).$$

- What function $\phi(x)$ would lead to this as the inner-product?
  - To simplify, assume $d = 1$ and $\sigma = 1$,

$$\begin{aligned} k(x_i, x_j) &= \exp(-x_i^2 + 2x_i x_j - x_j^2) \\ &= \exp(-x_i^2)\exp(2x_i x_j)\exp(-x_j^2), \end{aligned}$$

  so we need $\phi(x_i) = \exp(-x_i^2) z_i$ where $z_i z_j = \exp(2x_i x_j)$.
    - For this to work for *all* x_i and x_j, z_i must be infinite-dimensional.
  - If we use that

$$\exp(2x_i x_j) = \sum_{k=0}^{\infty} \frac{2^k x_i^k x_j^k}{k!},$$

  then we obtain

$$\phi(x_i) = \exp(-x_i^2)\left[1 \quad \sqrt{\tfrac{2}{1!}}x_i \quad \sqrt{\tfrac{2^2}{2!}}x_i^2 \quad \sqrt{\tfrac{2^3}{3!}}x_i^3 \quad \cdots\right].$$

Kernel Trick for Structured Data

- Kernel trick is useful for structured data:
  - Consider that doesn't look like this:

$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

Kernel Trick for Structured Data

- Kernel trick is useful for structured data:
  - Consider that doesn't look like this:

$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

but instead looks like this:

$$X = \begin{bmatrix} \text{Do you want to go for a drink sometime?} \\ \text{J'achète du pain tous les jours.} \\ \text{Fais ce que tu veux.} \\ \text{There are inner products between sentences?} \end{bmatrix}, y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

Kernel Trick for Structured Data

- Kernel trick is useful for structured data:
  - Consider that doesn't look like this:

$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

    but instead looks like this:

$$X = \begin{bmatrix} \text{Do you want to go for a drink sometime?} \\ \text{J'achète du pain tous les jours.} \\ \text{Fais ce que tu veux.} \\ \text{There are inner products between sentences?} \end{bmatrix}, y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

  - We could convert sentences to features, or define kernel between sentences.
  - For example, "string" kernels:
    - Weighted frequency of common subsequences (dynamic programming).
  - There are also "graph kernels", "image kernels", and so on...

Valid Kernels

- What kernel functions $k(x_i, x_j)$ can we use?
- Kernel k must be an inner product in some space:
  - There exists ϕ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

Valid Kernels

- What kernel functions $k(x_i, x_j)$ can we use?
- Kernel k must be an inner product in some space:
  - There exists ϕ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

We can decompose a (continuous or finite-domain) function k into

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

iff it is symmetric and for any finite $\{x_1, x_2, \ldots, x_n\}$ we have $K \succeq 0$.

Valid Kernels

- What kernel functions $k(x_i, x_j)$ can we use?
- Kernel k must be an inner product in some space:
  - There exists ϕ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

We can decompose a (continuous or finite-domain) function k into

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

iff it is symmetric and for any finite $\{x_1, x_2, \ldots, x_n\}$ we have $K \succeq 0$.

- Nice in theory, what do we do in practice?
  - Show explicitly that $k(x_i, x_j)$ is an inner product.
  - Or show it can be constructed from other valid kernels.

Valid Kernels

- What kernel functions $k(x_i, x_j)$ can we use?
- Kernel k must be an inner product in some space:
  - There exists ϕ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

We can decompose a (continuous or finite-domain) function k into

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

iff it is symmetric and for any finite $\{x_1, x_2, \ldots, x_n\}$ we have $K \succeq 0$.

- Nice in theory, what do we do in practice?
  - Show explicitly that $k(x_i, x_j)$ is an inner product.
  - Or show it can be constructed from other valid kernels.
- If we use invalid kernel, lose inner-product interpretation but may work fine.

Bonus Slide: Constructing Feature Space

- Why is positive semi-definiteness important?
  - With finite domain we can define K over all points.
  - The condition $K \succeq 0$ means it has a spectral decomposition

$$K = U^T \Lambda U,$$

    where the eignevalues $\lambda_i \geq 0$ and so we have a real $\Lambda^{\frac{1}{2}}$.
  - Thus we hav $K = U^T \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} U = \|\Lambda^{\frac{1}{2}} U\|^2$ and we could use

$$\Phi(X) = \Lambda^{\frac{1}{2}} U, \text{ or } \phi(x_i) = \Lambda^{\frac{1}{2}} U_{:,i}.$$

- The above reasoning isn't quite right for continuous domains.
- The more careful generalization is known as "Mercer's theorem".

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
  - $k_1(\phi(x_i), \phi(x_j))$.

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
  - $k_1(\phi(x_i), \phi(x_j))$.
  - $\alpha k_1(x_i, x_j) + \beta k_2(x_i, x_j)$ for $\alpha \geq 0$ and $\beta \geq 0$.

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
    - $k_1(\phi(x_i), \phi(x_j))$.
    - $\alpha k_1(x_i, x_j) + \beta k_2(x_i, x_j)$ for $\alpha \geq 0$ and $\beta \geq 0$.
    - $k_1(x_i, x_j) k_2(x_i, x_j)$.

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
  - $k_1(\phi(x_i), \phi(x_j))$.
  - $\alpha k_1(x_i, x_j) + \beta k_2(x_i, x_j)$ for $\alpha \geq 0$ and $\beta \geq 0$.
  - $k_1(x_i, x_j)k_2(x_i, x_j)$.
  - $\phi(x_i)k_1(x_i, x_j)\phi(x_j)$.

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
  - $k_1(\phi(x_i), \phi(x_j))$.
  - $\alpha k_1(x_i, x_j) + \beta k_2(x_i, x_j)$ for $\alpha \geq 0$ and $\beta \geq 0$.
  - $k_1(x_i, x_j)k_2(x_i, x_j)$.
  - $\phi(x_i)k_1(x_i, x_j)\phi(x_j)$.
  - $\exp(k_1(x_i, x_j))$.

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
  - $k_1(\phi(x_i), \phi(x_j))$.
  - $\alpha k_1(x_i, x_j) + \beta k_2(x_i, x_j)$ for $\alpha \geq 0$ and $\beta \geq 0$.
  - $k_1(x_i, x_j) k_2(x_i, x_j)$.
  - $\phi(x_i) k_1(x_i, x_j) \phi(x_j)$.
  - $\exp(k_1(x_i, x_j))$.
- Example: Gaussian-RBF kernel:

$$
\begin{aligned}
k(x_i, x_j) &= \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) \\
&= \underbrace{\exp\left(-\frac{\|x_i\|^2}{\sigma^2}\right)}_{\phi(x_i)} \underbrace{\exp\left(\underbrace{\frac{2}{\sigma^2}}_{\alpha \geq 0} \underbrace{x_i^T x_j}_{\text{valid}}\right)}_{\exp(\text{valid})} \underbrace{\exp\left(-\frac{\|x_j\|^2}{\sigma^2}\right)}_{\phi(x_j)}.
\end{aligned}
$$

Kernels Trick for Distance-Based Methods

- Besides ridge regression, when can we apply the kernel trick?

Kernels Trick for Distance-Based Methods

- Besides ridge regression, when can we apply the kernel trick?
  - Distance-based methods from CPSC 340:

$$\|x_i - x_j\|^2 = \langle x_i, x_j \rangle - 2\langle x_i, x_j \rangle + \langle x_j, x_j \rangle.$$

Kernels Trick for Distance-Based Methods

- Besides ridge regression, when can we apply the kernel trick?
  - Distance-based methods from CPSC 340:

$$\|x_i - x_j\|^2 = \langle x_i, x_j \rangle - 2\langle x_i, x_j \rangle + \langle x_j, x_j \rangle.$$

    - k-nearest neighbours.
    - Clustering algorithms (k-means, density-based clustering, hierarchical clustering).
    - Amazon item-to-item product recommendation.
    - Non-parametric regression.
    - Outlier ratio.
    - Multi-dimensional scaling.
    - Graph-based semi-supervised learning.

Kernels Trick for Distance-Based Methods

- Besides ridge regression, when can we apply the kernel trick?
  - Distance-based methods from CPSC 340:

$$\|x_i - x_j\|^2 = \langle x_i, x_j \rangle - 2\langle x_i, x_j \rangle + \langle x_j, x_j \rangle.$$

    - k-nearest neighbours.
    - Clustering algorithms (k-means, density-based clustering, hierarchical clustering).
    - Amazon item-to-item product recommendation.
    - Non-parametric regression.
    - Outlier ratio.
    - Multi-dimensional scaling.
    - Graph-based semi-supervised learning.
  - Eigenvalue methods:
    - Principle component analysis (trick for centering in high-dimensional space).
    - Canonical correlation analysis.
    - Spectral clustering.
  - L2-regularized linear models...

Representer Theorem

- Consider linear model differentiable with losses f_i and L2-regularization,

$$\mathop{\mathrm{argmin}}_{w \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2.$$

Representer Theorem

- Consider linear model differentiable with losses f_i and L2-regularization,

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2.$$

- Setting the gradient equal to zero we get

$$0 = \sum_{i=1}^{n} f_i'(w^T x_i)x_i + \lambda w.$$

Representer Theorem

- Consider linear model differentiable with losses f_i and L2-regularization,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2.$$

- Setting the gradient equal to zero we get

$$0 = \sum_{i=1}^{n} f_i'(w^T x_i)x_i + \lambda w.$$

- So any solution w^* can written as a linear combination of features x_i,

$$\begin{aligned} w^* &= -\frac{1}{\lambda}\sum_{i=1}^{n} f_i'((w^*)^T x_i)x_i = \sum_{i=1}^{n} z_i x_i \\ &= X^T z. \end{aligned}$$

- This is called a representer theorem (true under much more general conditions).

Representer Theorem

- Using representer theorem we can use $w = X^T z$ in original problem,

$$\begin{aligned} &\operatorname*{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2 \\ =&\operatorname*{argmin}_{z \in \mathbb{R}^n} \sum_{i=1}^n f_i(\underbrace{z^T X x_i}_{x_i^T X^T z}) + \frac{\lambda}{2}\|X^T z\|^2 \end{aligned}$$

Representer Theorem

- Using representer theorem we can use $w = X^T z$ in original problem,

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2$$

$$= \operatorname*{argmin}_{z \in \mathbb{R}^n} \sum_{i=1}^{n} f_i(\underbrace{z^T X x_i}_{x_i^T X^T z}) + \frac{\lambda}{2}\|X^T z\|^2$$

- Now defining $f(z) = \sum_{i=1}^n f_i(z_i)$ for a vector z we have

$$= \operatorname*{argmin}_{z \in \mathbb{R}^n} f(XX^T z) + \frac{\lambda}{2} z^T X X^T z$$

$$= \operatorname*{argmin}_{z \in \mathbb{R}^n} f(Kz) + \frac{\lambda}{2} z^T K z.$$

Representer Theorem

- Using representer theorem we can use $w = X^T z$ in original problem,

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2$$

$$= \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} f_i(\underbrace{z^T X x_i}_{x_i^T X^T z}) + \frac{\lambda}{2}\|X^T z\|^2$$

- Now defining $f(z) = \sum_{i=1}^n f_i(z_i)$ for a vector z we have

$$= \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} f(XX^T z) + \frac{\lambda}{2} z^T X X^T z$$

$$= \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} f(Kz) + \frac{\lambda}{2} z^T K z.$$

- Similarly, at test time we can use the n variables z,

$$\hat{X}w = \hat{X}X^T z = \hat{K}z.$$

(pause)

Fenchel Dual

- For convex f and g and the primal problem

$$\underset{w \in \mathbb{R}^d}{\text{argmin}}\, P(w) = f(Xw) + g(w),$$

the Fenchel dual is given by

$$\underset{z \in \mathbb{R}^n}{\text{argmax}}\, D(z) = -f^*(-z) - g^*(X^T z),$$

where f^* is the convex conjugate.

Fenchel Dual

- For convex f and g and the primal problem

$$\underset{w \in \mathbb{R}^d}{\text{argmin}}\, P(w) = f(Xw) + g(w),$$

  the Fenchel dual is given by

$$\underset{z \in \mathbb{R}^n}{\text{argmax}}\, D(z) = -f^*(-z) - g^*(X^T z),$$

  where f^* is the convex conjugate.
- Why are we interested in this?
  - Dual has fewer variables if $n < d$.
  - $D(z^*) = P(w^*)$ (strong duality): we can solve dual instead of primal.

Fenchel Dual

- For convex f and g and the primal problem

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \, P(w) = f(Xw) + g(w),$$

the Fenchel dual is given by

$$\underset{z \in \mathbb{R}^n}{\operatorname{argmax}} \, D(z) = -f^*(-z) - g^*(X^T z),$$

where f^* is the convex conjugate.

- Why are we interested in this?
  - Dual has fewer variables if $n < d$.
  - $D(z^*) = P(w^*)$ (strong duality): we can solve dual instead of primal.
  - $D(z) \leq P(w)$ for all w and z (weak duality): dual gives lower bound on primal.

Fenchel Dual

- For convex f and g and the primal problem

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} P(w) = f(Xw) + g(w),$$

  the Fenchel dual is given by

$$\operatorname*{argmax}_{z\in\mathbb{R}^n} D(z) = -f^*(-z) - g^*(X^Tz),$$

  where f^* is the convex conjugate.
- Why are we interested in this?
  - Dual has fewer variables if $n < d$.
  - $D(z^*) = P(w^*)$ (strong duality): we can solve dual instead of primal.
  - $D(z) \leq P(w)$ for all w and z (weak duality): dual gives lower bound on primal.
  - If P is strongly-convex, dual is smooth: smooth formulation of SVMs.

Fenchel Dual

- For convex f and g and the primal problem

$$\underset{w\in\mathbb{R}^d}{\text{argmin}}\, P(w) = f(Xw) + g(w),$$

the Fenchel dual is given by

$$\underset{z\in\mathbb{R}^n}{\text{argmax}}\, D(z) = -f^*(-z) - g^*(X^T z),$$

where f^* is the convex conjugate.

- Why are we interested in this?
  - Dual has fewer variables if $n < d$.
  - $D(z^*) = P(w^*)$ (strong duality): we can solve dual instead of primal.
  - $D(z) \leq P(w)$ for all w and z (weak duality): dual gives lower bound on primal.
  - If P is strongly-convex, dual is smooth: smooth formulation of SVMs.
  - Dual sometimes allows sparse kernel representation.

Supremum and Infimum

- The supremum of a function f is its smallest upper-bound,

$$\sup f(x) = \min_{y | y \geq f(x)} y.$$

Supremum and Infimum

- The supremum of a function f is its smallest upper-bound,

$$\sup f(x) = \min_{y | y \geq f(x)} y.$$

- Generalization of max that includes limits:

$$\max_{x \in \mathbb{R}} -x^2 = 0, \quad \sup_{x \in \mathbb{R}} -x^2 = 0,$$

  but

$$\max_{x \in \mathbb{R}} -e^x = \text{DNE}, \quad \sup_{x \in \mathbb{R}} -e^x = 0.$$

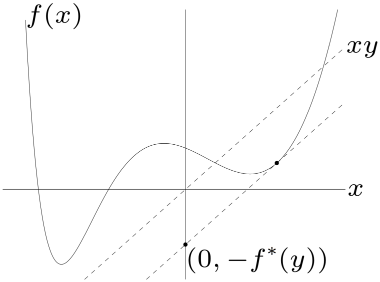
- The analogy for min is called the infimum.

Convex Conjugate

- The convex conjugate f^* of a function f is given by

$$f^*(y) = \sup_{x \in \mathcal{D}} \{y^T x - f(x)\},$$

where $\mathcal{D}$ is values where $\sup$ is finite.



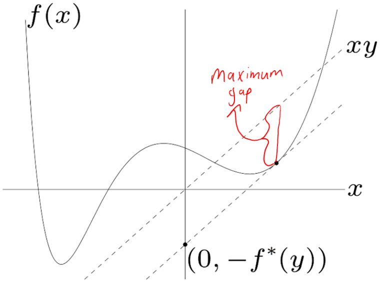
http://www.seas.ucla.edu/~vandenbe/236C/lectures/conj.pdf

Convex Conjugate

- The convex conjugate f^* of a function f is given by

$$f^*(y) = \sup_{x \in \mathcal{D}} \{y^T x - f(x)\},$$

where $\mathcal{D}$ is values where $\sup$ is finite.



http://www.seas.ucla.edu/~vandenbe/236C/lectures/conj.pdf

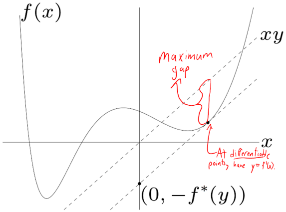
- It's the maximum that the linear function $y^T x$ can get above $f(x)$.

Convex Conjugate

- The convex conjugate f^* of a function f is given by

$$f^*(y) = \sup_{x \in \mathcal{D}} \{ y^T x - f(x) \},$$

  where $\mathcal{D}$ is values where $\sup$ is finite.



http://www.seas.ucla.edu/~vandenbe/236C/lectures/conj.pdf

- If f is differentable, then $\sup$ occurs at x where $y = \nabla f(x)$.
- Note that f^* is convex even if f is not.
- If f is convex (and "closed"), then $f^{**} = f$.

Convex Conjugate Examples

- If $f(x) = \frac{1}{2}\|x\|^2$ we have
  - $f^*(y) = \sup_x \{y^T x - \frac{1}{2}\|x\|^2\}$ or equivalently (by taking derivative and setting to 0):

$$0 = y - x,$$

and pluggin in $x = y$ we get

$$f^*(y) = y^T y - \frac{1}{2}\|y\|^2 = \frac{1}{2}\|y\|^2.$$

Convex Conjugate Examples

- If $f(x) = \frac{1}{2}\|x\|^2$ we have
  - $f^*(y) = \sup_x \{y^T x - \frac{1}{2}\|x\|^2\}$ or equivalently (by taking derivative and setting to 0):

$$0 = y - x,$$

and pluggin in $x = y$ we get

$$f^*(y) = y^T y - \frac{1}{2}\|y\|^2 = \frac{1}{2}\|y\|^2.$$

- If $f(x) = a^T x$ we have

$$f^*(y) = \sup_x \{y^T x - a^T x\} = \sup_x \{(y - a)^T x\} = \begin{cases} 0 & y = a \\ \infty & \text{otherwise.} \end{cases}$$

- For other examples, see Boyd & Vandenberghe.

Fenchel Dual of SVMs

- Consider support vector machines,

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2.$$

The Fenchel dual is given by

$$\underset{0\leq z\leq 1}{\operatorname{argmax}} \sum_{i=1}^{n} z_i - \frac{1}{2\lambda} \underbrace{\|\tilde{X}^T z\|^2}_{z^T \tilde{X}\tilde{X}^T z},$$

where $\tilde{X}$ =diag$(y)X$, $w^* = \frac{1}{\lambda}\tilde{X}^T z^*$ and constraints come from $f^* < \infty$.

Fenchel Dual of SVMs

- Consider support vector machines,

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2.$$

  The Fenchel dual is given by

$$\underset{0 \leq z \leq 1}{\operatorname{argmax}} \sum_{i=1}^{n} z_i - \frac{1}{2\lambda} \underbrace{\|\tilde{X}^T z\|^2}_{z^T \tilde{X}\tilde{X}^T z},$$

  where $\tilde{X}$ =diag$(y)X$, $w^* = \frac{1}{\lambda}\tilde{X}^T z^*$ and constraints come from $f^* < \infty$.
- A couple magical things have happened:
  - We can apply kernel trick.

Fenchel Dual of SVMs

- Consider support vector machines,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2.$$

  The Fenchel dual is given by

$$\underset{0 \leq z \leq 1}{\text{argmax}} \sum_{i=1}^{n} z_i - \frac{1}{2\lambda} \underbrace{\|\tilde{X}^T z\|^2}_{z^T \tilde{X}\tilde{X}^T z},$$

  where $\tilde{X}$ =diag$(y)X$, $w^* = \frac{1}{\lambda}\tilde{X}^T z^*$ and constraints come from $f^* < \infty$.
- A couple magical things have happened:
  - We can apply kernel trick.
  - Dual is differentiable (though not strongly-convex).

Fenchel Dual of SVMs

- Consider support vector machines,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2.$$

  The Fenchel dual is given by

$$\underset{0 \leq z \leq 1}{\text{argmax}} \sum_{i=1}^{n} z_i - \frac{1}{2\lambda} \underbrace{\|\tilde{X}^T z\|^2}_{z^T \tilde{X}\tilde{X}^T z},$$

  where $\tilde{X}$ =diag$(y)X$, $w^* = \frac{1}{\lambda}\tilde{X}^T z^*$ and constraints come from $f^* < \infty$.
- A couple magical things have happened:
  - We can apply kernel trick.
  - Dual is differentiable (though not strongly-convex).
  - Dual variables z are sparse (non-zeroes are called "support vectors"):
    - Can give faster training and testing.

Fenchel Dual of SVMs

- Consider support vector machines,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2.$$

  The Fenchel dual is given by

$$\underset{0 \leq z \leq 1}{\text{argmax}} \sum_{i=1}^{n} z_i - \frac{1}{2\lambda} \underbrace{\|\tilde{X}^T z\|^2}_{z^T \tilde{X}\tilde{X}^T z},$$

  where $\tilde{X} =$diag$(y)X$, $w^* = \frac{1}{\lambda}\tilde{X}^T z^*$ and constraints come from $f^* < \infty$.
- A couple magical things have happened:
  - We can apply kernel trick.
  - Dual is differentiable (though not strongly-convex).
  - Dual variables z are sparse (non-zeroes are called "support vectors"):
    - Can give faster training and testing.
  - Case where coordinate optimization is efficient.

Stochastic Dual Coordinate Ascent

- If we have an L2-regularized linear model with convex f_i,

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2,$$

then the Fenchel dual is given by

$$\operatorname*{argmax}_{z\in\mathbb{R}^n} -\underbrace{\sum_{i=1}^{n} f_i^*(z_i)}_{\text{separable}} - \frac{1}{2\lambda}\underbrace{\|X^T z\|^2}_{z^T {\color{blue}XX^T} z}.$$

Stochastic Dual Coordinate Ascent

- If we have an L2-regularized linear model with convex f_i,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2,$$

  then the Fenchel dual is given by

$$\underset{z \in \mathbb{R}^n}{\text{argmax}} \underbrace{-\sum_{i=1}^{n} f_i^*(z_i)}_{\text{separable}} - \frac{1}{2\lambda} \underbrace{\|X^T z\|^2}_{z^T X X^T z}.$$

- We can apply stochastic dual coordinate ascent (SDCA):
  - Only looks at one training example on each iteration.
  - Obtains $O(\log(1/\epsilon))$ rate if ∇f_i are L-Lipschitz.
    - Performance similar to SAG for many problems, worse if $\mu >> \lambda$.

Stochastic Dual Coordinate Ascent

- If we have an L2-regularized linear model with convex f_i,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2,$$

then the Fenchel dual is given by

$$\underset{z \in \mathbb{R}^n}{\text{argmax}} \underbrace{-\sum_{i=1}^{n} f_i^*(z_i)}_{\text{separable}} - \frac{1}{2\lambda} \underbrace{\|X^T z\|^2}_{z^T X X^T z}.$$

- We can apply stochastic dual coordinate ascent (SDCA):
  - Only looks at one training example on each iteration.
  - Obtains $O(\log(1/\epsilon))$ rate if ∇f_i are L-Lipschitz.
    - Performance similar to SAG for many problems, worse if $\mu >> \lambda$.
  - Obtains $O(1/\epsilon)$ rate for non-smooth f:
    - Same rate as stochastic subgradient, but we can now use exact/adaptive step-size.
    - You could add an L2-regularizer to dual, corresponds to smoothing primal.

(pause)

Notation Change

- We're going to start much more about subsets of features.
- To make this easier, we're going to change notation:

Notation Change

- We're going to start much more about subsets of features.
- To make this easier, we're going to change notation:
  - We'll refer to a generic feature vector as x, and a specific one as x^i:
    - Previously we used x_i for both.

Notation Change

- We're going to start much more about subsets of features.
- To make this easier, we're going to change notation:
  - We'll refer to a generic feature vector as x, and a specific one as x^i:
    - Previously we used x_i for both.
  - We'll refer to a generic feature as x_j, and a feature in a specific instance as x_j^i.
    - Previously we used x_{ij} for both.

Notation Change

- We're going to start much more about subsets of features.
- To make this easier, we're going to change notation:
  - We'll refer to a generic feature vector as x, and a specific one as x^i:
    - Previously we used x_i for both.
  - We'll refer to a generic feature as x_j, and a feature in a specific instance as x_j^i.
    - Previously we used x_{ij} for both.
  - To refer to a set of variables S, we'll use x_S or x_S^i.

Notation Change

- We're going to start much more about subsets of features.
- To make this easier, we're going to change notation:
  - We'll refer to a generic feature vector as x, and a specific one as x^i:
    - Previously we used x_i for both.
  - We'll refer to a generic feature as x_j, and a feature in a specific instance as x_j^i.
    - Previously we used x_{ij} for both.
  - To refer to a set of variables S, we'll use x_S or x_S^i.
  - To refer to all variables except x_j, we'll use x_{-j}.

Unsupervised Learning

- Supervised learning:
  - We have instances of features x^i and class labels y^i.
  - Want a program that gives y^i from corresponding x^i.
- Unsupervised learning:
  - We only have x^i values, but no explicit target labels.
  - You want to do "something" with them.

Unsupervised Learning

- Supervised learning:
  - We have instances of features x^i and class labels y^i.
  - Want a program that gives y^i from corresponding x^i.
- Unsupervised learning:
  - We only have x^i values, but no explicit target labels.
  - You want to do "something" with them.
- Some unsupervised learning tasks from CPSC 340:
  - Clustering: what types of x^i are there?
  - Association rules: which x_j and x_k occur together?
  - Outlier detection: is this a 'normal' x^i?
  - Latent-factors: what 'parts' are x^i made from?
  - Data visualization: what do the high-dimensional x^i look like?
  - Ranking: which are the most important x^i?

Density Estimation

- We're going to focus on the task of density estimation:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

  - What is $p(x^i)$ for a generic feature vector x^i?

Density Estimation

- We're going to focus on the task of density estimation:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

  - What is $p(x^i)$ for a generic feature vector x^i?
- This is abstract, but does has a wide variety of applications:
  - Outlier detection: low $p(x^i)$ could mean "outlier".
  - Missing values: we can estimate likely values of missing features.
  - Finding relationships: how is x_j related to x_k?
  - Compression: we can assign shorter codes to frequent x^i values.

Density Estimation

- We're going to focus on the task of density estimation:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

  - What is $p(x^i)$ for a generic feature vector x^i?
- This is abstract, but does has a wide variety of applications:
  - Outlier detection: low $p(x^i)$ could mean "outlier".
  - Missing values: we can estimate likely values of missing features.
  - Finding relationships: how is x_j related to x_k?
  - Compression: we can assign shorter codes to frequent x^i values.
  - Generative classifiers: build a model of $p(x^i)$ for each class label.

Density Estimation

- We're going to focus on the task of density estimation:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

  - What is $p(x^i)$ for a generic feature vector x^i?
- This is abstract, but does has a wide variety of applications:
  - Outlier detection: low $p(x^i)$ could mean "outlier".
  - Missing values: we can estimate likely values of missing features.
  - Finding relationships: how is x_j related to x_k?
  - Compression: we can assign shorter codes to frequent x^i values.
  - Generative classifiers: build a model of $p(x^i)$ for each class label.
  - Structured prediction: computing $p(y^i|x^i)$ where y^i is an object.
  - Bayesian learning: computing posterior $p(w|y, X)$.
  - Most unsupervised deep learning models.

Bernoulli Distribution on Binary Variables

- Let's start with the simplest case: $x \in \{0, 1\}$ (e.g., coin flips),

$$X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Bernoulli Distribution on Binary Variables

- Let's start with the simplest case: $x \in \{0, 1\}$ (e.g., coin flips),

$$X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

- In this case the only choice is the Bernoulli distribution:

$$p(x = 1|\theta) = \theta, \quad p(x = 0|\theta) = 1 - \theta.$$

Bernoulli Distribution on Binary Variables

- Let's start with the simplest case: $x \in \{0, 1\}$ (e.g., coin flips),

$$X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

- In this case the only choice is the Bernoulli distribution:

$$p(x = 1|\theta) = \theta, \quad p(x = 0|\theta) = 1 - \theta.$$

- We can write both cases

$$p(x|\theta) = \theta^{\mathcal{I}[x=1]}(1-\theta)^{\mathcal{I}[x=0]}, \text{ where } \mathcal{I}[y] = \begin{cases} 1 & \text{if } y \text{ is true} \\ 0 & \text{if } y \text{ is false} \end{cases}.$$

- Given an IID X, what is $p(x|\theta)$?

Maximum Likelihood with Bernoulli Distribution

- Maximum likelihood: choose θ maximizing likelihood of data we saw:

$$\begin{aligned}\underset{0\leq\theta\leq 1}{\text{argmax}}\, p(X|\theta) &= \underset{0\leq\theta\leq 1}{\text{argmax}} \prod_{i=1}^{n} p(x^i|\theta) \\ &= \underset{0\leq\theta\leq 1}{\text{argmax}} \prod_{i=1}^{n} \theta^{\mathcal{I}[x^i=1]}(1-\theta)^{\mathcal{I}[x^i=0]} \\ &= \underset{0\leq\theta\leq 1}{\text{argmax}}\, \theta^{N_1}(1-\theta)^{N_0},\end{aligned}$$

where N_1 is count of number of 1 values and N_0 is the number of 0 values.

Maximum Likelihood with Bernoulli Distribution

- Maximum likelihood: choose θ maximizing likelihood of data we saw:

$$\begin{aligned}\underset{0\leq\theta\leq 1}{\text{argmax}}\, p(X|\theta) &= \underset{0\leq\theta\leq 1}{\text{argmax}} \prod_{i=1}^{n} p(x^i|\theta)\\ &= \underset{0\leq\theta\leq 1}{\text{argmax}} \prod_{i=1}^{n} \theta^{\mathcal{I}[x^i=1]}(1-\theta)^{\mathcal{I}[x^i=0]}\\ &= \underset{0\leq\theta\leq 1}{\text{argmax}}\, \theta^{N_1}(1-\theta)^{N_0},\end{aligned}$$

  where N_1 is count of number of 1 values and N_0 is the number of 0 values.
- If you equate the derivative of the log-likelihood with zero, you get $\theta = \frac{N_1}{N_1+N_0}$.
- So if you toss a coin 50 times and it lands heads 24 times, your MLE is $24/50$.

Multinomial Distribution on Categorical Variables

- Consider the multi-category case: $x \in \{1, 2, 3, \ldots, k\}$ (e.g., rolling di),

$$X = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 3 \\ 1 \\ 2 \end{bmatrix}.$$

Multinomial Distribution on Categorical Variables

- Consider the multi-category case: $x \in \{1, 2, 3, \ldots, k\}$ (e.g., rolling di),

$$X = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 3 \\ 1 \\ 2 \end{bmatrix}.$$

- The categorical distribution is

$$p(x = c|\theta_1, \theta_2, \ldots, \theta_k) = \theta_c,$$

  where $\sum_{c=1}^{k} \theta_c = 1$.
- We can write this for a generic x as

$$p(x|\theta_1, \theta_2, \ldots, \theta_k) = \prod_{c=1}^{k} \theta_c^{\mathcal{I}[x=c]}.$$

Multinomial Distribution on Categorical Variables

- Using Lagrange multipliers to add constraint to log-likelihood, the MLE is

$$\theta_c = \frac{N_c}{\sum_{c'} N_{c'}}.$$

Multinomial Distribution on Categorical Variables

- Using Lagrange multipliers to add constraint to log-likelihood, the MLE is

$$\theta_c = \frac{N_c}{\sum_{c'} N_{c'}}.$$

- What if we never see category 4 in the data, should we assume $\theta_4 = 0$?

Multinomial Distribution on Categorical Variables

- Using Lagrange multipliers to add constraint to log-likelihood, the MLE is

$$\theta_c = \frac{N_c}{\sum_{c'} N_{c'}}.$$

- What if we never see category 4 in the data, should we assume $\theta_4 = 0$?
  - As before, what we care about is accurately estimating test set likelihood:
  - If we assume $\theta_4 = 0$ we have a 4 in test set, we do very bad.

Multinomial Distribution on Categorical Variables

- Using Lagrange multipliers to add constraint to log-likelihood, the MLE is

$$\theta_c = \frac{N_c}{\sum_{c'} N_{c'}}.$$

- What if we never see category 4 in the data, should we assume $\theta_4 = 0$?
  - As before, what we care about is accurately estimating test set likelihood:
  - If we assume $\theta_4 = 0$ we have a 4 in test set, we do very bad.
- To leave room for this possibility we often use "Laplace smoothing",

$$\theta_c = \frac{N_c + 1}{\sum_{c'}(N'_c + 1)}.$$

- This is like adding a 'fake' example to the training set for each class.

MAP Estimation with Bernoulli Distributions

- In the binary case, a generalization of Laplace smoothing is

$$\theta = \frac{N_1 + \alpha - 1}{(N_1 + \alpha - 1) + (N_0 + \beta - 1)},$$

MAP Estimation with Bernoulli Distributions

- In the binary case, a generalization of Laplace smoothing is

$$\theta = \frac{N_1 + \alpha - 1}{(N_1 + \alpha - 1) + (N_0 + \beta - 1)},$$

  which is a MAP estimate under a beta prior,

$$p(\theta|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha - 1} (1 - \theta)^{\beta - 1},$$

  where the beta function B makes the probability integrate to one.
- We get the MLE when $\alpha = \beta = 1$, and the Laplace smoothing with $\alpha = \beta = 2$.

MAP Estimation with Categorical Distributions

- In the categorical case, a generalization of Laplace smoothing is

$$\theta_c = \frac{N_c + \alpha_c - 1}{\sum_{c'=1}^{k}(N_{c'} + \alpha_c - 1)},$$

which is a MAP estimate under a Dirichlet prior,

$$p(\theta_1, \theta_2, \dots, \theta_k | \alpha_1, \alpha_2, \dots, \alpha_k) = \frac{1}{B(\alpha)} \prod_{c=1}^{k} \theta_c^{\alpha_c - 1},$$

where B is the multinomial beta.

- Because of MAP/regularization connection, Laplace smoothing is regularization.

General Discrete Distribution

- Now consider the case where $x \in \{0, 1\}^d$ (e..g, words in e-mails):

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

General Discrete Distribution

- Now consider the case where $x \in \{0,1\}^d$ (e..g, words in e-mails):

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

- Now there are 2^d possible values of x.
  - Can't afford to even store a θ for each possible x.

General Discrete Distribution

- Now consider the case where $x \in \{0,1\}^d$ (e..g, words in e-mails):

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

- Now there are 2^d possible values of x.
  - Can't afford to even store a θ for each possible x.
  - With n training examples we see at most n unique x^i values.
  - But unless we have a small number of repeated x values, we'll hopelessly overfit.
- With finite dataset, we'll need to make assumptions...

Product of Independent Distributions

- A common assumption is that the variables are independent:

$$p(x_1, x_2, \ldots, x_d|\Theta) = \prod_{j=1}^{d} p(x_j|\theta_j).$$

Product of Independent Distributions

- A common assumption is that the variables are independent:

$$p(x_1, x_2, \dots, x_d | \Theta) = \prod_{j=1}^{d} p(x_j | \theta_j).$$

- Now we just need to model each column of X as its own dataset:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}. \quad X_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \dots$$

.

- A big assumption, but now you can fit Bernoulli for each variable.
  - The assumption underlying naive Bayes in CPSC 340.

Density Estimation and Fundamental Trade-off

- Product of independent distributions:
  - Easily estimate each θ_c but can't model many distributions.

Density Estimation and Fundamental Trade-off

- Product of independent distributions:
  - Easily estimate each θ_c but can't model many distributions.
- General discrete distribution:
  - General discrete: hard to estimate 2^d parameters but can model any distribution.

Density Estimation and Fundamental Trade-off

- Product of independent distributions:
  - Easily estimate each θ_c but can't model many distributions.
- General discrete distribution:
  - General discrete: hard to estimate 2^d parameters but can model any distribution.
- An unsupervised version of the fundamental trade-off:
  - Simple models often don't fit the data well but don't overfit much.
  - Complex models fit the data well but often overfit.

Summary

- Valid kernels are typically constructed from other valid kernels.

Summary

- Valid kernels are typically constructed from other valid kernels.
- Representer theorem allows kernel trick for L2-regularized linear models.

Summary

- Valid kernels are typically constructed from other valid kernels.
- Representer theorem allows kernel trick for L2-regularized linear models.
- Fenchel dual re-writes sum of convex functions with convex conjugates:
  - Dual may have nice structure: differentiable, sparse, coordinate optimization.

Summary

- Valid kernels are typically constructed from other valid kernels.
- Representer theorem allows kernel trick for L2-regularized linear models.
- Fenchel dual re-writes sum of convex functions with convex conjugates:
    - Dual may have nice structure: differentiable, sparse, coordinate optimization.
- Density estimation: unsupervised modelling of probability of feature vectors.

Summary

- Valid kernels are typically constructed from other valid kernels.
- Representer theorem allows kernel trick for L2-regularized linear models.
- Fenchel dual re-writes sum of convex functions with convex conjugates:
  - Dual may have nice structure: differentiable, sparse, coordinate optimization.
- Density estimation: unsupervised modelling of probability of feature vectors.
- Product of independent distributions is simple/crude density estimation method.

- Next time:
  - Continuous density estimation and what lies between independent/general models.