03/09/2016

Kernels

continued

- Kernel Perceptrons
- Note: Everywhere below, we can replace X_i with $\phi(X_i)$

```
Perceptron Algorithm:
    while (some y_i X_i · w < 0):
        w ← w + ε y_i x_i
    while (still have test points z):
        f(z) ← w^T x
```

- Kernelize with $w = X^T a$; $X_i \cdot w = (XX^T a)_i = (Ka)_i$.

```
Dual Perceptron Algorithm:
    a = [y_1, 0, ..., 0]^T
    K_ij = K(X_i, X_j)  ∀i, j
    while (some y_i (Ka)_i · w < 0):
        a ← a + ε y_i
    while (still have test points z):
        f(z) ← Σ_{j=1}^{n} a_j K(X_j, z)
```

- Computing $K_{ij} \Leftarrow \mathcal{O}(n^2 d)$ time (kernel trick).
- Optimization for first while loop: maintain Ka, $\mathcal{O}(n)$ time.
- Second loop runs in $\mathcal{O}(nd)$ time or we can compute $w = X^T a$ once in $\mathcal{O}(nd')$ time and evaluate test points in $\mathcal{O}(d')$ time per point, where d' is the length of $\phi(\cdot)$.

Kernel Logistic Regression

- Stochastic gradient ascent step:

$$a_i \leftarrow a_i + \epsilon(y_i - s((Ka)_i))$$

- Gradient ascent step:

$$a \leftarrow a + \epsilon(y_i - s((Ka)) \Leftarrow \text{ applying s component-wise to vector } Ka$$

- for each test point z: $h(z) \leftarrow s(\sum_{j=1}^{n} a_j k(X_j, z))$.

The Gaussian Kernel

- Gaussian kernel, aka radial basis function kernel
- There exists a $\phi(x)$ such that:

$$k(x, z) = e^{(-\frac{|x-z|^2}{2\sigma^2})}$$

- Key observation: $h(z) = s(\sum_{j=1}^{n} a_j k(X_j, z))$ is a linear combination of Gaussian centered at samples.
- Very popular in practice:
  - Gives very smooth h
  - Behaves somewhat like k-nearest neighbor, but smoother.
  - Oscillates less than polynomial (depending on σ).
  - $k(x, z)$ can be interpreted as a "similarity measure" y value more influenced by points around it.
  - Gaussian is maximum when $x = z$, goes to 0 as distance increases.
  - Samples "vote" for value at z, but closer samples get bigger vote.
- σ trades off bias vs. variance:
  - larger $\sigma \to$ wider Gaussians, smoother $h \to$ more bias, less variance.
  - Choose by (cross)-validation.

Subset Selection

- All features increase variance, but not all features reduce bias.
- Idea: Identify poorly predictive features, ignore them (weight zero). Less over-fitting, lower test errors.
- 2nd motivation: Inference. Simpler models convey interpretable wisdom. Useful in all classification and regression methods. Sometimes it's hard: Different feature can partly encode some info. Combinatorially hard to choose best feature subset.
- Algorithm: Best subset selection. Try all $2^d - 1$ nonempty subsets of features. Choose the best model by cross-validation. This is very slow!
- Heuristic: Forward stepwise selection.
  - Start with null model (0 features); repeatedly add best feature. Repeat until test errors start increasing (due to over-fitting).
  - Requires training $\mathcal{O}(d^2)$ models instead of $\mathcal{O}(2^d)$.
  - Not perfect.
- Heuristic: Backward stepwise selection.
  - Start with all d features; repeatedly remove feature whose removal gives best reduction in test error.
  - Also trains $\mathcal{O}(d^2)$ models.
- If you think there's only a few good features do forward, if you think most features will be good do backwards.

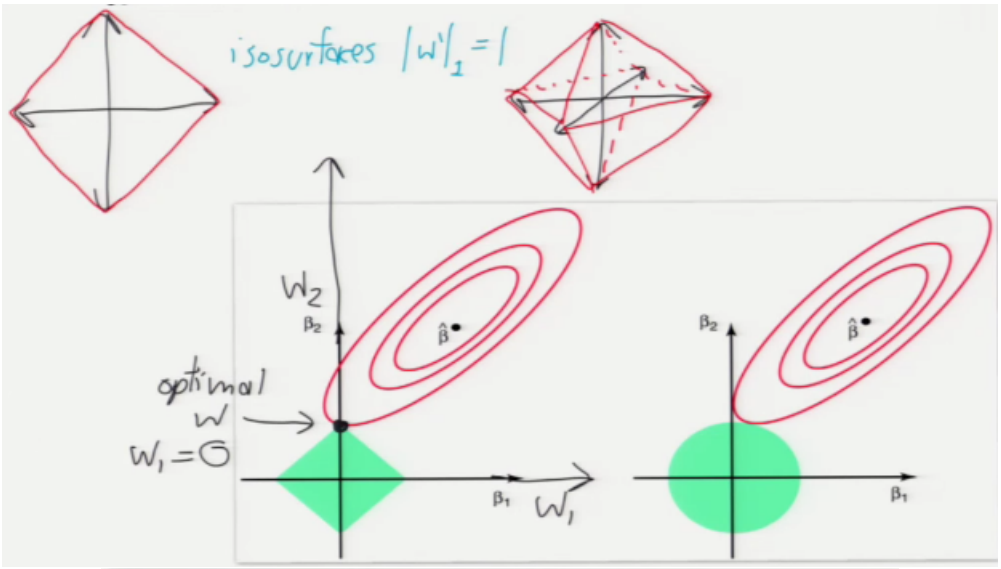
Lasso (Tibsharani, 1996)

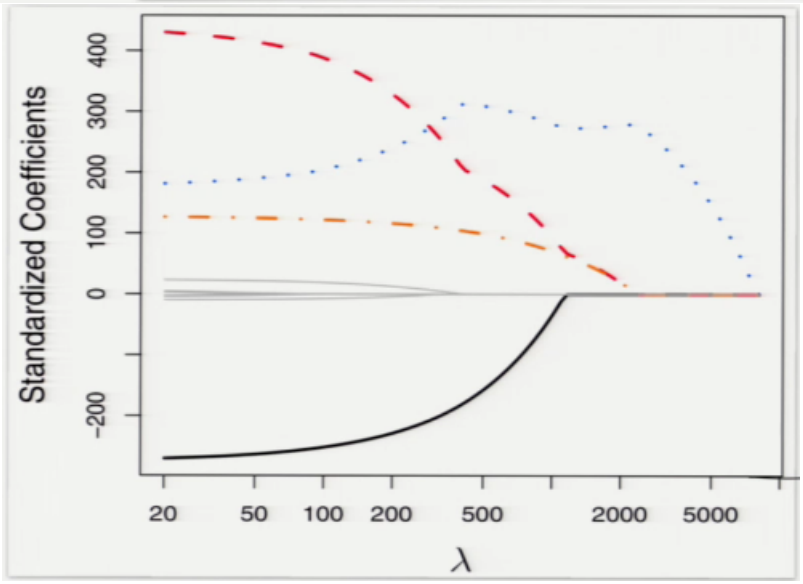
- Regression with regularization: (1) + (A) + ℓ_1 penalized mean loss (e) "Least absolute shrinkage and selection operator"
- Optimization problem:

$$\boxed{\text{Find } w \text{ that minimizes } |Xw - y|^2 + \lambda |w'|_1}$$

  where $|w'|_1 = \sum_{i=1}^{n} |w_i|$.
- Recall ridge regression: isosurfaces of $|w'|_2^2$ are hyperspheres.
- The isosurfaces of $|w'|_1$ are cross-polytopes.

- The unit cross-polytope is the convex hull of all the positive and negative unit coordinate vectors.





- Algorithms to optimize: subgradient descent, least-angle regression (LARS), forward stagewise.