# Machine Learning in Computer Vision
# MVA 2011 - Lecture Notes

Iasonas Kokkinos
Ecole Centrale Paris
iasonas.kokkinos@ecp.fr

October 18, 2011

# Chapter 1

# Introduction

This is an informal set of notes for the first part of the class. They are by no means complete, and do not substitute attending class. My main intent is to give you a skimmed version of the class slides, which you will hopefully find easier to consult.

These notes were compiled based on several standard textbooks including:

- T. Hastie, R. Tibshirani and J. Friedman, 'Elements of Statistical Learning', Springer 2009.
- V. Vapnik, 'The Nature of Statistical Learning', Springer 2000.
- C. Bishop, 'Pattern Recognition and Machine Learning', Springer 2006.

Apart from these, helpful resources can be found online; in particular Andrew Ng has a highly-recommended set of handouts here:
`http://cs229.stanford.edu/materials.html`
while the more theoretically inclined can consult Robert Schapire's course:
`http://www.cs.princeton.edu/courses/archive/spring08/cos511/schedule.html`
and Peter Bartlett's course:
`http://www.cs.berkeley.edu/~bartlett/courses/281b-sp08/`
for more extensive introductions to learning theory, boosting and SVMs.

As this is the first edition of these lecture notes, there will inevitably be inconsistencies in notation with the slides, typos, etc; your feedback will be most valuable in improving them for the next year.

# Chapter 2

# Linear Regression

Assumed form of discriminant function:

$$f(x, \mathbf{w}) = \sum_{j=1}^{M} \mathbf{w}_j x_j + \mathbf{w}_0 = \sum_{j=0}^{M} \mathbf{w}_j x_j, \quad x_0 = 1 \tag{2.1}$$

In the more general case:

$$f(x, \mathbf{w}) = \sum_{j=0}^{M} \mathbf{w}_j B_j(x) \tag{2.2}$$

where $B_j(x)$ can be a nonlinear function of $x$.

We want to minimize the empirical loss (=training loss) of our classifier quantified in terms of the following criterion:

$$\begin{aligned} L(\mathbf{w}) &= \sum_i l(y^i, f(x^i, \mathbf{w})) \\ &= \sum_{i=1}^{N} (y^i - f(x^i; \mathbf{w}))^2 \\ &= \sum_{i=1}^{N} \left( y^i - \sum_{j=1}^{M} \mathbf{w}_j x_j^i \right)^2 \end{aligned}$$

We introduce vector/matrix notation:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad N \times 1 \qquad \mathbf{X} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_M^1 \\ \vdots & & & \\ x_1^N & & & x_N^N \end{bmatrix} \qquad N \times M,$$

and the cost becomes:

$$\begin{aligned} L(\mathbf{w}) &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{e}^T \mathbf{e}, \end{aligned}$$

where $\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w}$ is called the residual vector. This is why the criterion is also known as residual-sum-of-squares cost.

Optimizing the training criterion gives:

$$\begin{aligned} \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} &= 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \\ \mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\hat{\mathbf{w}} \\ \hat{\mathbf{w}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}\mathbf{y} \qquad (2.3) \\ \hat{\mathbf{y}} &= \mathbf{X}\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}X^T y \qquad (2.4) \end{aligned}$$

## 2.1 Ridge regression

What happens when the regression coefficients are correlated? As an extreme case, consider $x_1^i = x_2^i, \forall i$. Then the matrix $(\mathbf{X}^T\mathbf{X})^{-1}$ becomes ill-conditioned and the corresponding coefficients can take on arbitrary values: a large positive value of one coefficient can be canceled by a large negative one.

Therefore we regularize our problem by introducing a penalty $C(\mathbf{w})$ on the expansion coefficients:

$$L(\mathbf{w}, \lambda) = \mathbf{e}^T\mathbf{e} + \lambda C(\mathbf{w}) \qquad (2.5)$$

where $\lambda$ determines the tradeoff between the data term $\mathbf{e}^T\mathbf{e}$ and the regularizer.

For the case of ridge regression, we penalize the $l_2$ norm of the weight vector, i.e. $\|w\|_2^2 = \sum_{k=1}^K w_k^2 = \mathbf{w}^T\mathbf{w}$. The cost function becomes:

$$\begin{aligned} L_{ridge}(\mathbf{w}) &= \mathbf{e}^T\mathbf{e} + \lambda\mathbf{w}^T\mathbf{w} \qquad (2.6) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{w} \end{aligned}$$

The condition for an optimum of $L(\mathbf{w})$ gives:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \qquad (2.7)$$

So the estimate labels will be:

$$\hat{y} = \mathbf{X}\mathbf{w}^* \qquad (2.8)$$

### 2.1.1 Lasso

Instead of the $l_2$ we now penalize the $l_1$ norm of the vector $\mathbf{w}$:

$$E_{Lasso}(\mathbf{w}, \lambda) = \sum_{i=1}^N (y^i - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\sum_{k=1}^M |\mathbf{w}_k| \qquad (2.9)$$

The derivative of $E_{Lasso}$ w.r.t. $\mathbf{w}_i$ at any point will be given by:

$$\frac{\partial(E_{Lasso}(\mathbf{w}))}{\partial \mathbf{w}_i} = \lambda\text{sign}(\mathbf{w}_i) + \sum_{i=1}^N -2(y^i - \mathbf{X}\mathbf{w})^T\mathbf{w}_i$$

If we use for instance Gradient descent to minimize $E_{Lasspo}$, the rate at which $\mathbf{w}_i$ will be updated will not be affected by the magnitude of $\mathbf{w}_i$, but only on its sign. Note that is in contrast to the logistic regression case:

$$\frac{\partial(E_{Ridge}(\mathbf{w}))}{\partial \mathbf{w}_i} = \lambda(\mathbf{w}_i) + \sum_{i=1}^{M} -2(y^i - \mathbf{X}\mathbf{w})^T \mathbf{w}_i,$$

where small-in-magnitude values of $\mathbf{w}_i$ 'can survive' as they become more slowly suppressed as they decrease. This difference, due to the different norms used, leads to sparser solutions in Lasso.

## 2.2 Selection of $\lambda$

If we optimize w.r.t $\lambda$ directly, $\lambda = 0$ is the obvious, but trivial solution. This solution will suffer from ill-posedness: as mentioned earlier, the matrix $\mathbf{X}^T\mathbf{X}$ may be poorly conditioned for correlated $X$. So, the parameters of the model may exhibit high variance, depending on the specific training set used.

What $\lambda$ does is regularize our problem, i.e. impose some constraints that render it well-posed. We first see why this is so.

### 2.2.1 Bias variance decomposition of error

Assume actual generation process

$$y = g(x) + \epsilon \tag{2.10}$$

Our model: $y = \hat{g}(x) = f_{\hat{w}}(x)$, and $\hat{w} = h(\lambda, S)$
Expected generalization error at $x_0$:

$$\begin{aligned} & x_0 && && (2.11) \\ \mathrm{Err}(x_0) &= E[(y - \hat{g}(x_0))^2] \\ &= E[(y - g(x_0) + g(x_0) - \hat{g}(x_0))^2] \\ &= E[((y - g(x_0)) + (g(x_0) - E[\hat{g}(x_0)]) + (E[\hat{g}(x_0)] - \hat{g}(x_0)))^2] \\ &= \sigma^2 + \underbrace{(g(x_0) - E[\hat{g}(x_0])))^2}_{Bias} + \underbrace{E[(E[\hat{g}(x_0)] - \hat{g}(x_0))^2]}_{Variance} && && (2.12) \end{aligned}$$

Bias is a decreasing function of model complexity. Variance is increasing.

### 2.2.2 Cross Validation

Setting $\lambda$ allows us to control the tradeoff between bias and variance. Consider two extreme cases: $\lambda = 0$: no regularization - ill posed problem, and high variance. $\lambda = \inf$: constant model - high bias. The best is somewhere in between. Use cross-validation to estimate $\lambda$.

## 2.3 SVD-based interpretations (optional reading)

Consider the following Singular Value Decomposition for the $N \times M$ matrix $\mathbf{X}$ [1]:

$$\begin{aligned} \underbrace{\mathbf{X}}_{N\times M} &= \underbrace{\mathbf{U}}_{N\times M}\,\underbrace{\mathbf{D}}_{M\times M}\,\underbrace{\mathbf{V^T}}_{M\times M} \\ \mathbf{D} &= \mathrm{diag}(d_1, d_2, \ldots, d_{\min(M,N)}, \underbrace{0, \ldots, 0}_{\max(M-N,0)}), \quad d_i \geq d_{i+1}, d_i \geq 0 \end{aligned}$$

while the columns of $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and unit-norm. We remind that $N$ is the number of training points and $M$ the dimensionality of the features.

A first thing to note is that if we have zero-mean ('centered') data, $\mathbf{X^T X}$ will be the covariance matrix of the data. Moreover we have:

$$\mathbf{X^T X} = \mathbf{V D^T U^T U D V^T} = \mathbf{V D^T D V^T} = \mathbf{V D^2 V^T} \tag{2.14}$$

so $V$ will contain the eigenvalues of the covariance matrix of the data.

Further, we can use the SVD decomposition interpret certain quantities obtained in the previous sections. First, for (2.4), which gives the estimated output $\mathbf{y}$ in linear regression, we have:

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ \hat{\mathbf{y}} &= \mathbf{U D V^T}\left(\mathbf{V D^T U^T U D V^T}\right)^{-1}\mathbf{V^T D^T U y} \\ &= \mathbf{U}\left(\mathbf{U}^T\mathbf{y}\right) \end{aligned}$$

We can interpret this expression as follows:
$\mathbf{U}$: forms an orthonormal basis for a subspace of $R^M$.
$\mathbf{C_H} = \mathbf{U^T h}$: gives us the expansion coefficients for the projection of a vector $\mathbf{h}$ on this basis.
$\hat{\mathbf{h}} = \mathbf{U C_h}$: is the reconstruction of $\mathbf{h}$ on basis $\mathbf{U}$.

We therefore have that $\mathbf{U^T y}$ gives the projection coefficients for $\mathbf{y}$ on $\mathbf{U}$. So $\hat{\mathbf{y}} = \mathbf{U U^T y}$, gives us the projection of $\mathbf{y}$ on basis $U$

For the case of ridge regression, using the expression in (2.8) we get:

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{U}\mathrm{D}\left(\mathbf{D}^2 + \lambda\mathbf{I}\right)^{-1}\mathbf{D U^T y} \\ &= \sum_{i=1}^{M} \mathbf{u}_m \frac{d_j^2}{d_j^2 + \lambda}\mathbf{u_m^T y} \end{aligned}$$

[1] This version of SVD is slightly non-typical; it is the one used in Hastie, Tibshirani and Friedman. The default is for $\mathbf{D}$ to be $N \times M$: Typical SVD, for $N > M$:

$$\begin{aligned} \underbrace{\mathbf{X}}_{N\times M} &= \underbrace{\mathbf{U}}_{N\times N}\,\underbrace{\mathbf{D}}_{N\times M}\,\underbrace{\mathbf{V^T}}_{M\times M} \\ \mathbf{D} &= \left[\mathrm{diag}(d_1, d_2, \ldots, d_N)|(0)_{N\times N-M}\right], \quad d_i \geq d_{i+1}, d_i \geq 0 \end{aligned} \tag{2.13}$$

Hence the name 'shrinkage': the expansion coefficients for the reconstruction of $\mathbf{y}$ on the subspace spanned by $\mathbf{U}$ are shrunk by $\frac{d_j^2}{d_j^2+\lambda} < 1$ w.r.t. the values used in linear regression.

# Chapter 3

# Logistic Regression

## 3.1 Probabilistic transition from linear to logistic regression

Consider that the labels are generated according to the linear model + gaussian noise assumption:

$$P(y_i|\mathbf{x}_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{x}\mathbf{w}^T\right)^2}{2\sigma^2}\right). \tag{3.1}$$

We consider as a 'merit function' the likelihood of the data:

$$C(\mathbf{w}) = P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_i P(y^i|\mathbf{x^i}, \mathbf{w})$$

or more conveniently, the log-likelihood:

$$\begin{aligned} \log(C(\mathbf{w})) = \log P(\mathbf{y}|\mathbf{X}) &= \sum_i \log P(y^i|\mathbf{x^i}) \\ &= c - \frac{1}{2\sigma^2}\sum_i (y^i - \mathbf{w}^T\mathbf{x}^i)^2 \\ &= c - \frac{1}{2\sigma^2}\left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)^T\left(\mathbf{y} - \mathbf{X}\mathbf{w}\right) = \\ &= c - \frac{1}{2\sigma^2}\mathbf{e}^T\mathbf{e} \end{aligned} \tag{3.2}$$

So minimizing the least squares criterion amounts to maximizing the log Likelihood of the labels under the model in (3.1). But this model make sense only for continuous observations. In classification the observations are discrete (labels). We therefore consider using a Bernoulli distribution instead:

$$\begin{aligned} P(y = 1|\mathbf{x}, \mathbf{w}) &= f(\mathbf{x}, \mathbf{w}) \\ P(y = 0|\mathbf{x}, \mathbf{w}) &= 1 - f(\mathbf{x}, \mathbf{w}). \end{aligned}$$

We can compactly express these two equations as:

$$P(y|\mathbf{x}) = (f(\mathbf{x}, \mathbf{w}))^y (1 - f(\mathbf{x}, \mathbf{w})^{1-y} \tag{3.3}$$

We can now write the (log-)likelihood of the data as:

$$\begin{aligned} C(\mathbf{w}) = P(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_i P(y^i|\mathbf{x}^i, \mathbf{w}) \\ \log C(\mathbf{w}) = \log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \sum_i \log P(y^i|\mathbf{x}^i, \mathbf{w}) \\ &= \sum_i y^i \log f(\mathbf{x}^i, \mathbf{w}) + (1 - y^i) \log(1 - f(\mathbf{x}^i, \mathbf{w})) \end{aligned} \tag{3.4}$$

Now consider using the following expression for $f(\mathbf{x}, \mathbf{w})$:

$$f(\mathbf{x}, \mathbf{w}) = g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} \tag{3.5}$$

$g(a) = \frac{1}{1+\exp(-a)}$ is called the sigmoidal function and has several interesting properties, the most useful ones being:

$$\frac{dg}{da} = g(a)(1 - g(a)), \quad 1 - g(a) = g(-a)$$

Plugging (3.5) into (3.4), this is the criterion we should be maximizing:

$$\log C(\mathbf{w}) = \log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{i=1}^{N} y^i \log g(\mathbf{w}^T \mathbf{x}^i) + (1 - y^i) \log(1 - g(\mathbf{w}^T \mathbf{x}^i)) \tag{3.6}$$

A simple approach to maximize it is to use Gradient ascent:

$$w_k^{t+1} = w_k^t + \alpha \left. \frac{\partial C(\mathbf{w})}{\partial w_k} \right|_{\mathbf{w}^t} \tag{3.7}$$

Using the expression in (3.6), we have:

$$\begin{aligned} \frac{\partial C(\mathbf{w})}{\partial w_k} &= \sum_{i=1}^{N} \left[ y^i \frac{1}{g(\mathbf{w}^T \mathbf{x}^i)} \frac{\partial g(\mathbf{w}^T \mathbf{x}^i)}{\partial w_k} + (1 - y^i) \frac{1}{1 - g(\mathbf{w}^T \mathbf{x}^i)} (-\frac{\partial g(\mathbf{w}^T \mathbf{x}^i)}{\partial w_k}) \right] \\ &= \sum_{i=1}^{N} \left[ y^i \frac{1}{g(\mathbf{w}^T \mathbf{x}^i)} - (1 - y^i) \frac{1}{1 - g(\mathbf{w}^T \mathbf{x}^i)} \right] g(\mathbf{w}^T \mathbf{x}^i)(1 - g(\mathbf{w}^T \mathbf{x}^i)) \frac{\partial \mathbf{w}^T \mathbf{x}^i}{\partial w_k} \\ &= \sum_{i=1}^{N} \left[ y^i (1 - g(\mathbf{w}^T \mathbf{x}^i)) - (1 - y^i) g(\mathbf{w}^T \mathbf{x}^i) \right] x_k^i \\ &= \sum_{i=1}^{N} \left[ y^i - g(\mathbf{w}^T \mathbf{x}^i) \right] \mathbf{x}_k^i \end{aligned}$$

We can write this more compactly by using vector notation; using the $N \times 1$ vector $\mathbf{g}$ with elements:

$$\mathbf{g}_i = g(\mathbf{w}^T \mathbf{x}^i) \tag{3.8}$$

and the $N \times 1$-dimensional $k$-th column $\mathbf{X}_k$ of the $N \times M$ matrix $\mathbf{X}$, we have:

$$\frac{\partial C(\mathbf{w})}{\partial w_k} = [\mathbf{y} - \mathbf{g}]^T \mathbf{X}_k = \mathbf{X}_k^T [\mathbf{y} - \mathbf{g}]$$

Stacking the partial derivatives together we form the Jacobian:

$$J(\mathbf{w}) = \frac{dC(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \mathbf{X}_1^T [\mathbf{y} - \mathbf{g}] \\ \dots \\ \mathbf{X}_M^T [\mathbf{y} - \mathbf{g}] \end{bmatrix} = \mathbf{X}^T [\mathbf{y} - \mathbf{g}]$$

The gradient-ascen update for $\mathbf{w}$ will be:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \alpha J_{\mathbf{w}}^T$$

A more elaborate, but more efficient technique is the Newton-Raphson method. Consider first the Newton method for finding the roots of a 1-D function $f(\theta)$:

$$\theta' = \theta - \frac{f(\theta)}{f'(\theta)}$$

If we want to maximize the function, we need to find roots of its derivative $f'$; so we apply the Newton method to find the roots of $f'(\theta)$:

$$\theta' = \theta - \frac{f'(\theta)}{f''(\theta)}$$

In our case we want to maximize a multidimensional function, which leads us to the Newton Raphson method:

$$\mathbf{w}' = \mathbf{w} - (H(\mathbf{w}))^{-1} J(\mathbf{w}). \tag{3.9}$$

We already saw how to compute the Jacobian. For the $k, j$-th element of the Hessian we have

$$\begin{aligned} H(\mathbf{w})(k,j) &= \frac{\partial^2 \log C(\mathbf{w})}{\partial w_k \partial w_j} \\ &= \frac{\partial \sum_{i=1}^N \left[y^i - g(\mathbf{x}^i \mathbf{w}^T)\right] \mathbf{x}_k^i}{\partial w_j} \\ &= -\sum_{i=1}^N \frac{\partial g(\mathbf{x}^i \mathbf{w}^T)}{\partial w_j} \mathbf{x}_k^i \\ &= -\sum_{i=1}^N g(\mathbf{x}^i \mathbf{w}^T)(1 - g(\mathbf{x}^i \mathbf{w}^T)) \mathbf{x}_j^i \mathbf{x}_k^i \\ &= -\mathbf{X}^T \mathbf{R} \mathbf{X}, \quad \text{where} \quad \mathbf{R}_{i,i} = g(x^i)(1 - g(x^i)) \end{aligned} \tag{3.10}$$

This Newton-Raphson update is called 'Iteratively Reweighted Least Squares' in Statistics; to see why this is so, we rewrite the update as:

$$\begin{aligned}
\mathbf{w}' &= \mathbf{w} + \left(\mathbf{X}^T R\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\mathbf{y}-\mathbf{g}\right) \\
&= \left(\mathbf{X}^T R\mathbf{X}\right)^{-1}\left[\left(\mathbf{X}^T R\mathbf{X}\right)\mathbf{w_0} + \mathbf{X}^T(\mathbf{y}-\mathbf{g})\right] \\
&= \left(\mathbf{X}^T R\mathbf{X}\right)^{-1}\mathbf{X}^T R\left[\mathbf{X}\mathbf{w} + \mathbf{R}^{-1}(\mathbf{y}-\mathbf{g})\right] \\
&= \left(\mathbf{X}^T R\mathbf{X}\right)^{-1}\mathbf{X}^T R\mathbf{z}
\end{aligned}$$

where we have introduced the 'adjusted response':

$$\begin{aligned}
\mathbf{z} &= \mathbf{X}\mathbf{w} + \mathbf{R}^{-1}(\mathbf{y}-\mathbf{g}) \\
\mathbf{z}_i &= \mathbf{x}^i \underbrace{\mathbf{w}}_{\text{Previous value}} + \frac{1}{g_i(1-g^i)}(y^i - g^i),
\end{aligned} \tag{3.11}$$

Using this new vector $\mathbf{z}$, we can interpret the vector $\mathbf{w}$ as being the solution to a weighted least squares problem:

$$\mathbf{w}' = \text{argmin}_{\mathbf{w}} \sum_{i=1}^{N} R_i(z_i - \mathbf{x}_i\mathbf{w}^T)^2 \tag{3.12}$$

Note that apart from $\mathbf{z}$, also the matrix $\mathbf{R}$ is updated at each iteration; this explains the 'iteratively re-' prefix in the 'IRLS' name.

## 3.2 Log-loss

We simplify the expression for our training criterion to see what logistic regression does. We therefore introduce:

$$y_{\pm} = 2y_b - 1 \tag{3.13}$$

$$y_b \in \{0, 1\} \tag{3.14}$$

$$y_{\pm} \in \{-1, 1\} \tag{3.15}$$

$$y_b = \frac{y_{\pm}+1}{2}, \tag{3.16}$$

where $y_b$ is the label we have been using so far and $y_{\pm}$ is a new label that makes certain expressions easier.

In specific we have:

$$P(y=1|f(x)) = \frac{1}{1+\exp(-f(x))} \tag{3.17}$$

$$P(y=-1|f(x)) = 1 - P(y=1|f(x)) \tag{3.18}$$

$$= \frac{1}{\exp(f(x))+1} \tag{3.19}$$

$$P(y^i|f(x^i)) = \frac{1}{1+\exp(-y^i f(x^i))} \tag{3.20}$$

So we can write:

$$L(\mathbf{w}) = -C(\mathbf{w}) = \sum_{i=1}^{N} \log(1 + \exp(-y_{\pm}^{i} f(x^i))) \tag{3.21}$$

The quantity

$$l(y^i, x^i) = \log(1 + \exp(-y_{\pm}^{i} f(x^i))) \tag{3.22}$$

is called the log-loss.

# Chapter 4

# Support Vector Machines

## 4.1 Large Margin classifiers

In linear classification, we are concerned with construction of separating hyperplane: $\mathbf{w}^T x = 0$.

For instance, logistic regression provides an estimate of:

$$P(y = 1|x; w) = g(\mathbf{w}^T x) = \frac{1}{1 + \exp(-\mathbf{w}^T x)}$$

and provide a decision rule using:

$$\hat{y}^i = \begin{cases} 1 & \text{if} P(y = 1|x; w) \geq .5 \\ 0 & \text{if} P(y = 1|x; w) < .5 \end{cases} \tag{4.1}$$

Intuitively, a 'confident' decision is made when $|\mathbf{w}^T x|$ is large. So we would like to have:

$$\begin{aligned} (\mathbf{w}^T x^i) &\gg 0 \quad \text{if} \quad y^i = 1 \\ (\mathbf{w}^T x^i) &\ll 0 \quad \text{if} \quad y^i = -1 \end{aligned}$$

or, combined:

$$y^i(\mathbf{w}^T x^i) \gg 0.$$

Given a training set, we want to find a decision boundary that allows us to make correct and confident predictions on all training examples.

For points near the decision boundary, even though we may be correct, small changes in the training set could cause different classifications; so our classifier will not be confident. If a point is far from the separating hyperplane then we are more confident in our predictions. In other words, we would like to have all points being far (and on the correct side of) the decision plane.

### 4.1.1 Notation

Instead of

$$h(x) = g(\mathbf{w}^T x) \tag{4.2}$$

we now consider classifiers of the form:

$$h(x) = g(\mathbf{w}^T x + b) \tag{4.3}$$

($b = w_0$), where $g(z) = 1$ if $z > 0$, and $g(z) = -1$, if $z < 0$. Unlike logistic regression, where $g$ gives a probability estimate, here the classifier directly outputs a decision.

### 4.1.2 Margins

The 'confidence scores' described above are called functional margins: scaling $\mathbf{w}, b$ by two doubles the margin, without essentially changing the classifier.

We therefore now turn to geometric margins, which are defined in terms of unit-norm vector $\frac{w}{|w|}$.

We express first any point as:

$$\mathbf{x} = \mathbf{x}_\perp + \gamma \frac{\mathbf{w}}{|\mathbf{w}|}$$

where $\mathbf{x}_\perp$ is the projection of $\mathbf{x}$ on the decision plane, $\mathbf{w}^T\mathbf{x}_\perp + b = 0$, $\frac{\mathbf{w}}{|\mathbf{w}|}$ is a unit-norm vector perpendicular to the decision plane and $\gamma$ is thus the distance of $\mathbf{x}$ from the decision plane. Solving for $\gamma$ we have:

$$\begin{aligned} \mathbf{w}^T\mathbf{x} &= \mathbf{w}^T\mathbf{x}_\perp + \mathbf{w}^T\gamma\frac{\mathbf{w}}{|\mathbf{w}|} \\ \mathbf{w}^T\mathbf{x} &= -b + \gamma|\mathbf{w}| \\ \gamma = \frac{\mathbf{w}^T\mathbf{x} + b}{|\mathbf{w}|} &= \frac{\mathbf{w}^T}{|\mathbf{w}|}\mathbf{x} + \frac{b}{|\mathbf{w}|} \end{aligned}$$

The above analysis works for the case where $y = 1$. If $y = -1$, we would like $\mathbf{w}^T\mathbf{x} + b$ to be negative, we therefore write:

$$\gamma^i = y^i \left( \frac{\mathbf{w}^T}{|\mathbf{w}|}\mathbf{x}^i + \frac{b}{|\mathbf{w}|} \right)$$

For a training set $x^i, y^i$ and a classifier $\mathbf{w}$ we define its geometric margin as:

$$\gamma = \min_i \gamma^i \tag{4.4}$$

which indicates the worst performance of the classifier on the training set.

### 4.1.3 Introducing margins in classification

Our task is to classify all points with maximal geometric margin. If we consider the case where the points are separable, we have the following optimization problem:

$$\begin{aligned} \max_{\gamma,\mathbf{w},b} \quad & \gamma \\ s.t. \quad & y^i(\mathbf{w}^T x^i + b) \geq \gamma, \quad i = 1 \dots M \\ & |\mathbf{w}| = 1 \end{aligned}$$

In words, maximize $\gamma$, where all points have at least $\gamma$ geometric margin. It is hard to incorporate the $|\mathbf{w}| = 1$ constraint (it is non-convex; for two dimensional inputs, $w$ should lie on a circle). We therefore divide both sides by $|\mathbf{w}|$:

$$\begin{aligned} \max_{\gamma,\mathbf{w},b} \quad & \frac{\gamma'}{|\mathbf{w}|} \\ s.t. \quad & y^i(\mathbf{w}^T x^i + b) \geq \gamma', \quad i = 1 \dots M \end{aligned}$$

From the constraint, since $\mathbf{w}$ is no longer constrained to have unit norm, we realize that $\gamma'$ is a functional margin. Consider we find a combination of $\mathbf{w}, b$ that solves the optimization problem. We get a one-dimensional family of solutions by scaling $\mathbf{w}, b, \gamma'$ by a common constant. We discard this ambiguity by consider only those values of $\mathbf{w}, b$ for which $\gamma' = 1$, which gives us the following problem:

$$\begin{aligned} \max_{\mathbf{w},b} \quad & \frac{1}{|\mathbf{w}|} \\ s.t. \quad & y^i(\mathbf{w}^T x^i + b) \geq 1, \quad i = 1 \dots M \end{aligned}$$

Equivalently:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}|\mathbf{w}|^2 \\ s.t. \quad & y^i(\mathbf{w}^T x^i + b) \geq 1, \quad i = 1 \dots M \end{aligned}$$

So we end up with a quadratic optimization problem, involving a Convex (quadratic) function of $\mathbf{w}$ and linear constraints on $\mathbf{w}$.

## 4.2 Duality

Consider a general constrained optimization problem:

$$\begin{aligned} \min_{w} \quad & f(w) \\ s.t. \quad & h_i(w) = 0, \quad i = 1 \dots l \\ & g_i(w) \leq 0, \quad i = 1 \dots m \end{aligned}$$

This is equivalent to unconstrained optimization:

$$\min_{w} \quad f_{uc}(w) = f(w) + \textstyle\sum_{i=1}^{l} I_0(h_i(w)) + \sum_{i=1}^{m} I_+(g_i(w)) \tag{4.5}$$

where $I_0, I_-$: hard constraints

$$I_0(x) = \begin{cases} 0, & x = 0 \\ \infty, & x \neq 0 \end{cases}, \qquad I_+(x) = \begin{cases} 0, & x \leq 0 \\ \infty, & x > 0 \end{cases}$$

We form a lower bound of this function, by replacing each hard constraint with a soft one:

$$I_0(x_i) \to \lambda_i x_i$$
$$I_+(x_i) \to \mu_i x_i, \mu > 0$$

$\lambda, \mu$: Penalty for violating constraints.

This gives us the Lagrangian:

$$L(w, \lambda, \mu) = f(w) + \sum_{i=1}^{l} \lambda_i h_i(w) + \sum_{i=1}^{m} \mu_i g_i(w), \quad \mu_i > 0 \forall i$$

We observe that

$$f_{uc}(w) = \max_{\lambda, \mu : \mu_i > 0} L(w, \lambda, \mu). \tag{4.6}$$

Therefore

$$f(w^*) = \min_{w} \max_{\lambda, \mu : \mu_i > 0} L(w, \lambda, \mu). \tag{4.7}$$

Now consider the Lagrange Dual Function:

$$\theta(\lambda, \mu) = \inf_{w} L(w, \lambda, \mu). \tag{4.8}$$

$\theta(\lambda, \mu)$ is always a lower bound on the optimal value $p^*$ of the original problem. If the original problem has a (feasible) solution $w^*$, then:

$$\begin{aligned} L(w^*, \lambda, \mu) \quad &= \quad f(w^*) + \sum_{i=1}^{l} \lambda_i h_i(w^*) + \sum_{i=1}^{m} \mu_i g_i(w^*) = \\ &\overset{w^*:feasible}{=} \quad f(w^*) + \sum_{i=1}^{l} \lambda_i 0 + \sum_{i=1}^{m} \mu_i \underbrace{g_i(w^*)}_{<0} = \\ &\overset{\mu_i > 0}{\leq} \quad f(w^*). \end{aligned}$$

So

$$\theta(\lambda, \mu) = \inf_{w} L(w, \lambda, \mu) \leq L(w^*, \lambda, \mu) \leq f(w^*). \tag{4.9}$$

So we have a function $\theta(\lambda, \mu)$ that lower bounds the minimum cost of our original problem. We want to make this bound as tight as possible, i.e. maximize it.

New task (dual problem):

$$\begin{aligned} \max_{\lambda,\mu} \quad & \theta(\lambda,\mu) \\ s.t. \quad & \mu_i > 0 \quad \forall i \end{aligned} \tag{4.10}$$

In general,

$$\begin{aligned} d^* &= \max_{\lambda,\mu} \theta(\lambda,\mu) \\ &= \max_{\lambda,\mu:\mu_i>0} \min_w L(w,\lambda,\mu) \\ &\leq \min_w \max_{\lambda,\mu:\mu_i>0} L(w,\lambda,\mu) \\ &= \min_w f_{uc}(w) = p^* \end{aligned}$$

If $f(w), h_i(w), f_i(w)$ are convex functions, and the primal and dual problems are feasible, then the bound is tight, i.e.

$$d^* = p^*,$$

which means that there exist $w^*, \lambda^*, \mu^*$ meeting all constraints while satisfying:

$$f(w^*) = \theta(\lambda^*, \mu^*) \tag{4.11}$$

At that point we have:

$$\begin{aligned} f(w^*) &= \theta(\lambda^*,\mu^*) \\ &= \inf_w f(w) + \sum_{i=1}^{l} \lambda_i^* h_i(w) + \sum_{i=1}^{m} \mu_i^* f_i(w) \\ &\leq f(w^*) + \sum_{i=1}^{M} \lambda_i^* h_i(w^*) + \sum_{i=1}^{m} \mu_i^* f_i(w^*) \\ &\leq f(w^*) \end{aligned}$$

Therefore: $\mu_i^* f_i(w^*) = 0, \quad \forall i$ ('Complementary Slackness').

### 4.2.1 Karush-Kuhn-Tucker Conditions

$L(w^*, \lambda^*, \mu^*)$ is a saddle point (minimum for $w$, maximum for $\lambda, \mu$). From the minimum w.r.t $w^*$ we have that

$$\nabla f(w^*) + \sum_{i=1}^{l} \lambda_i \nabla h_i(w^*) + \sum_{i=1}^{m} \nabla f_i(w^*) = 0$$

From the maximum w.r.t. $\lambda, \mu$ we have that the constraints are satisfied. Combining with the complementary slackness conditions we have the KKT conditions:

$$\begin{aligned} h_i(w^*) &= 0 \\ f_i(w^*) &\leq 0 \\ \mu_i f_i(w^*) &= 0 \\ \mu_i &\geq 0 \\ \nabla f(w^*) + \sum_{i=1}^{l} \lambda_i \nabla h_i(w^*) + \sum_{i=1}^{m} \nabla f_i(w^*) &= 0 \end{aligned}$$

### 4.2.2 Dual Problem for Large-Margin training

The primal problem is the objective we setup for max-margin training:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}|\mathbf{w}|^2 \\ s.t. \quad & -y^i(\mathbf{w}^T x^i + b) + 1 \leq 0, \quad i = 1 \dots M \end{aligned}$$

Its Lagrangian will be:

$$L(\mathbf{w}, b, \mu) = \frac{1}{2}|\mathbf{w}|^2 - \sum_{i=1}^{M} \mu_i [y^i(\mathbf{w}^T x^i + b) - 1] \tag{4.12}$$

At an optimum we have a minimum w.r.t. $\mathbf{w}$, which gives:

$$\begin{aligned} 0 &= \mathbf{w}^* - \sum_{i=1}^{M} \mu_i [y^i x^i] \\ \mathbf{w}^* &= \sum_{i=1}^{M} \mu_i y^i x^i, \end{aligned}$$

and a minimum w.r.t. $b$, which gives:

$$0 = \sum_{i=1}^{M} \mu_i y^i$$

Plugging in the optimal values into the Lagrangian we get:

$$\begin{aligned} \theta(\mu) &= L(\mathbf{w}^*, b^*, \mu) \\ &= \frac{1}{2}|\mathbf{w}^*|^2 - \sum_{i=1}^{M} \mu_i [y^i(\mathbf{w}^{*T} x^i + b) - 1] \\ &= \frac{1}{2}(\sum_{i=1}^{M} \mu_i y^i x^i)^T (\sum_{j=1}^{M} \mu_j y^j x^j) - \sum_{i=1}^{M} \mu_i [y^i((\sum_{j=1}^{M} \mu_j y^j x^j)^T x^i + b) - 1] \\ &= \sum_{i=1}^{M} \mu_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \mu_i \mu_j y^i y^j (x^i)^T (x^j) - b \sum_{i=1}^{M} \mu_i y^i \end{aligned}$$

We thereby eliminated $\mathbf{w}^*$ from the problem. For reasons that will become clear later, we write $(x^i)^T(x^j) =< x^i, x^j >$ (inner product). The new optimization problem becomes:

$$\begin{aligned} \max_{\mu} \quad & \theta(\mu) = \sum_{i=1}^{M} \mu_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \mu_i \mu_j y^i y^j < x^i, x^j > \\ s.t. \quad & \mu_i > 0, \quad \forall i \\ & \sum_{i=1}^{M} \mu_i y^i = 0 \end{aligned}$$

If we solve the problem above w.r.t. $\mu$, we can use $\mu$ to determine $\mathbf{w}^*$. One important note is that as

$$\mathbf{w}^* = \sum_{i=1}^{M} \mu_i y^i x^i, \tag{4.13}$$

the final decision boundary orientation is a linear combination of the training points.

Moreover, from complementary slackness we have:

$$\mu_i g_i(x) = 0, \tag{4.14}$$

where:

$$\begin{aligned} g_i(x) = 1 - y^i(\mathbf{w}^T x^i + b) \leq 0 \\ y^i(\mathbf{w}^T x^i + b) \geq 1 \end{aligned} \tag{4.15}$$

Therefore if $\mu_i \neq 0$ then $y^i(\mathbf{w}^T x^i + b) = 1$.

As $\mathbf{w}^*$ is formed from the set of points for which $\mu_i \neq 0$, this means that only the points that lie on the margin influence the solution. These points are called Support Vectors.

From the set $S$ of support vectors we can determine best $b^*$:

$$\begin{aligned} y^i(\mathbf{w}^T x^i + b) = 1, \quad \forall i \in S \\ b^* = \frac{1}{N_S} \sum_{i \in S} \left( y^i - \mathbf{w}^T x^i \right) \end{aligned} \tag{4.16}$$

### 4.2.3 Non-separable data

To deal with the case where the data cannot be separated, we introduce positive 'slack variables', $\xi^i, i = 1 \ldots M$ in the constraints:

$$\begin{aligned} \mathbf{w}^T x^i + b \geq \quad & 1 - \xi^i, \qquad y_i = 1 \\ \mathbf{w}^T x^i + b \leq \quad & -1 + \xi^i, \qquad y_i = -1 \end{aligned}$$

or equivalently:

$$y^i(\mathbf{w}^T x^i + b) \geq 1 - \xi^i$$

If $\xi^i > 1$ an error occurs, and $\sum_i \xi^i$ is an upper bound on the number of errors. We consider a new problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{M}\xi^i \\ s.t. \quad & y^i(\mathbf{w}^T x^i + b) \geq 1 - \xi^i \\ & \xi^i \geq 0 \end{aligned} \tag{4.17}$$

where $C$ determines the tradeoff between the large margin (low $\|\mathbf{w}\|$) and low error constraint.

The Lagrangian of this problem is:

$$L(\mathbf{w}, b, \xi, \mu, \nu) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{M}\xi^i - \sum_{i=1}^{M}\mu_i[y^i(\mathbf{w}^T x + b) - 1 + \xi] - \sum_{i=1}^{M}\nu_i\xi_i,$$

and its dual becomes:

$$\begin{aligned} \max_{\mu} \quad & \sum_{i=1}^{M}\mu_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y^i y^j \mu_i \mu_j < x^i, x^j > \\ s.t. \quad & 0 \leq \mu_i \leq C \\ & \sum_{i=1}^{M}\mu_i y^i = 0. \end{aligned}$$

The KKT conditions are similar; an interesting change is that we now have:

$$C - \mu^i - \nu^i = 0 \tag{4.18}$$
$$y^i(< \mathbf{x}^i, \mathbf{w} > +b) - 1 + \xi^i \geq 0 \tag{4.19}$$
$$\mu^i \left[y^i(< \mathbf{x}^i, \mathbf{w} > +b) - 1 + \xi^i\right] = 0 \tag{4.20}$$
$$\nu^i\xi^i = 0 \tag{4.21}$$
$$\xi^i \geq 0 \tag{4.22}$$
$$\mu^i \geq 0 \tag{4.23}$$
$$\nu^i \geq 0 \tag{4.24}$$

which gives

$$\begin{aligned} y^i(< \mathbf{x}^i, \mathbf{w} > +b) > 1 \quad & \overset{3:\xi^i \geq 0}{\rightarrow} \quad \mu_i = 0 \\ y^i(< \mathbf{x}^i, \mathbf{w} > +b) < 1 \quad & \overset{2:\xi^i > 0 \rightarrow 4:\nu^i = 0 \rightarrow 1:}{\rightarrow} \quad \mu_i = C \\ y^i(< \mathbf{x}^i, \mathbf{w} > +b) = 1 \quad & \overset{3:\mu^i\xi^i = 0 \rightarrow 1:}{\rightarrow} \quad \mu_i \in [0, C] \end{aligned}$$

$$\mu^i \neq 0 \overset{(3)}{\rightarrow} \xi^i = 1 - y^i(< \mathbf{x}^i, \mathbf{w} > +b) \tag{4.25}$$

$$\mu^i = 0 \overset{(1)}{\rightarrow} \nu^i = C \overset{(4)}{\rightarrow} \xi^i = 0 \tag{4.26}$$

$$\xi^i \geq 0 \tag{4.27}$$

$$\xi^i = \max(0, 1 - y^i(< \mathbf{x}^i, \mathbf{w} > +b)) \tag{4.28}$$

### 4.2.4 Kernels

Up to now, the decision was based on the sign of

$$y = \mathbf{w}^T x = \sum_{i=1}^{N} \mathbf{w}_i x_i \tag{4.29}$$

What if we used nonlinear transformations of the data?

$$y = \mathbf{w}^T \phi(x) = \sum_{i=1}^{K} \mathbf{w}_i \phi_k(x) \tag{4.30}$$

As a concrete example, consider

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix}$$

Using such features would allow us to go from linear to quadratic decision boundaries, defined by:

$$\sum_i \mathbf{w}_i \phi_i(x) = 0 \tag{4.31}$$

We observe that, in the linear case, for any new point $x$, the decision will have the form:

$$y = \text{sign}(< \mathbf{w}, \mathbf{x} > +b) \overset{\mu_i' = \mu_i y^i}{=} \text{sign}(\sum_{i=1}^{M} \mu_i' < \mathbf{x}^i, \mathbf{x} > +b) \tag{4.32}$$

So all we need to know in order to compute the response at a new $x$ is its inner product with the Support Vectors. So if we plug in our new features, we guess that we should get a classifier of the form:

$$y = \text{sign}(\sum_{i=1}^{M} \mu_i' < \phi(\mathbf{x}^i), \phi(\mathbf{x}) > +b) \tag{4.33}$$

Actually everything in the original algorithm was written in terms of inner products between input vectors. So we can write everything in terms of inner products of our new features $\phi$; in specific, the dual optimization problem becomes:

$$\begin{aligned} \max_{\mu} \quad & \sum_{i=1}^{M} \mu_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y^i y^j \mu_i \mu_j < \phi(x^i), \phi(x^j) > \\ s.t. \quad & 0 \leq \mu_i \leq C \\ & \sum_{i=1}^{M} \mu_i y^i = 0 \end{aligned}$$

If now we define a Kernel between two points $\mathbf{x}$ and $\mathbf{y}$ as:

$$K(\mathbf{x}, \mathbf{y}) =< \phi(\mathbf{x}), \phi(\mathbf{y}) >$$

and replace $< \mathbf{x}, \mathbf{y} >$ with $K(\mathbf{x}, \mathbf{y})$, we can write the same algorithm:

$$y = \text{sign}(\sum_{i=1}^{M} \mu_i y^i K(\mathbf{x}^i, \mathbf{x}) + b) \tag{4.34}$$

$$\begin{aligned} \max_{\mu} \quad & \sum_{i=1}^{M} \mu_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y^i y^j \mu_i \mu_j K(\mathbf{x}^i, \mathbf{x}^j) \\ s.t. \quad & 0 \leq \mu_i \leq C \\ & \sum_{i=1}^{M} \mu_i y^i = 0 \end{aligned}$$

As a concrete example, consider the mapping:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ \sqrt{2} x_1 x_2 \end{bmatrix}$$

Then

$$\begin{aligned} < \phi(x), \phi(y) > &= x_1^2 y_1^2 + 2 x_1 x_2 y_1 y_2 + x_2^2 y_2^2 \\ &= (x_1 y_1 + x_2 y_2)^2 \\ &= (x^T y)^2 \\ &= K(x, y) \end{aligned}$$

In the same way, we can use $K(x, y) = (x^T y + c)^d$ to implement a mapping into $\begin{pmatrix} N+d \\ d \end{pmatrix}$ space.

Another case:

$$K(x,y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2}) \tag{4.35}$$

It can be shown that this amounts to an inner product in an infinite-dimensional feature space.

K can de a distance-like measure among inputs, even though we may not have a straightforward feature-space interpretation; it allows us to apply SVMs to strings, graphs, documents etc. Most importantly, it allows us to work in very high dimensional spaces without necessarily finding and/or computing the mappings.

Under what conditions does a kernel function represent an inner product in some space? We want to have $K_{ij} = K(x^i, x^j) = K(x^j, x^i) = K_{ji}$. $K$: Gram matrix.

For $m$ points this should be valid $\forall i = 1 \dots m, j = 1 \dots m$.

Consider now the quantity:

$$\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi^T(x^i)\phi(x^j) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x^i)\phi_k(x^j) z_j \\
&= \sum_k \left[ \sum_i \sum_j z_i \phi_k(x^i)\phi_k(x^j) z_j \right] \\
&= \sum_k \left( \sum_i z_i \phi_k(x^i) \right)^2 \geq 0
\end{aligned}$$

$K$ is therefore positive semi-definite and symmetric.

'Mercer Kernel' consider a Kernel $K : R^N \times R^N \to R$; K represents an inner product if and only if K is symmetric positive semi-definite.

## 4.3 Hinge vs. Log-loss

Consider the SVM training criterion:

$$\begin{aligned}
L(w) &= \sum_{i=1}^{m} [\xi_i] + \frac{1}{2C} w^T w \\
&= \sum_{i=1}^{m} [1 - y^i (w^T x^i)]_+ + \lambda w^T w
\end{aligned} \tag{4.36}$$

$[1 - y^i(t^i)]_+$ : hinge loss

Compared to regularized Logistic regression

$$L(w) \quad = \quad \sum_{i=1}^{m} \log(1 + \exp(-y(w^T x^i)) + \lambda w^T w,$$

the hinge loss is more abrupt: as soon as as $y^i(w^T x^i) > 1$, the $i$-th point is declared 'well-classified' and no longer contributes to the solution (i.e. is not a support vector).

# Chapter 5

# Adaboost

If many cases it easy to find some rules of thumb that are often correct but hard to find a single highly accurate prediction rule

Boosting is a general method for converting a set of rules of thumb into highly accurate prediction rules; assuming that a given weak learning algorithm can consistently find classifiers (rules of thumb) at least slightly better than random, say, accuracy 51(in two-class setting), then given sufficient data a boosting algorithm can provably construct a single classifier with very high accuracy.

## 5.1 AdaBoost

Adaboost is a boosting algorithm that provides justifiable answers to the following questions:

- First, how to choose examples on each round? Adaboost concentrates on hardest examples, i.e. those most often misclassified by previous rules of thumb.
- Second how to combine rules of thumb into single prediction rule? Adaboost proposes to take a weighted majority vote of these rules of thumb.

In specific we consider that we are provided with a training set: $(x^i, y^i), x^i \in \mathcal{X}, y^i \in \{-1, 1\}, i = 1, \ldots, N$. Adaboost maintains a probability density on these data which is initially uniform, and on the way places larger weight on harder examples. At each round a weak classifier $h_t$ ('rule of thumb') is identified that has the best performance on the weighted data, and used to update (i) the classifier's expression (ii) the probability density on the data.

In specific Adaboost can be described in terms of the following algorithm:
Initially, set $D_1(i) = \frac{1}{N}, \quad \forall i$
For $t = 1 \ldots T$:

- Find weak classifier $h_t : \mathcal{X} \to \{-1, 1\}$ with smallest error

$$\epsilon_t = \frac{\sum_{i=1}^{N} D_t^i [y^i \neq h_t(x^i)]}{\sum_i D_t^i} \tag{5.1}$$

- Set $a_t = \frac{1}{2} \log \frac{(1-\epsilon_t)}{\epsilon_t}$
- Update distribution

$$\begin{aligned} D_{t+1}^i &= \frac{D_t^i}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if} y^i = h_t(x^i) \\ \exp(\alpha_t), & \text{if} y^i \neq h_t(x^i) \end{cases} \\ &= \frac{D_t^i}{Z_t} \exp(-\alpha_t y^i h_t(x^i)), \end{aligned}$$

  where to guarantee that $D_{t+1}$ remains a proper density we normalize by:

$$Z_t = \sum_i \frac{D_t^i}{\exp}(-\alpha_t y^i h_t(x^i))$$

- Combine classifier results (weighted vote)

$$f(x) = \sum_t a_t h_t(x)$$

- Output final classifier:

$$H(x) = \text{sign}\,(f(x)) \tag{5.2}$$

### 5.1.1 Training Error Analysis

We first introduce first the edge of the weak learner at the $t$-th round as:

$$\epsilon_t = \frac{1}{2} - \underbrace{\gamma_t}_{\text{Edge}}$$

We will prove that:

$$\begin{aligned} \underbrace{\sum_{i=1}^{N} \frac{1}{N} [y^i \neq H(x^i)]}_{\text{Final Training Error}} &\leq \prod_{t=1}^{T} \left[ 2\sqrt{\epsilon_t(1-\epsilon_t)} \right] \\ &= \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \\ &\leq \exp\left( -2 \sum_{t=1}^{T} \gamma_t^2 \right) \end{aligned}$$

The challenge lies in proving the first inequality; the second comes from the definition of the edge, while the third from the fact that $\sqrt{1-2x} \leq \exp(-x), x \in [0, \frac{1}{2}]$. If we

We have required that the weak learner is always correct more than half of the times on the training data, i.e. that $\gamma_t > 0 \forall t$, so if $\forall t\ \gamma_t > \gamma > 0$ we will then have:

$$\sum_{i=1}^{N} \frac{1}{N}[y^i \neq H(x^i)] =\leq \exp(-2T\gamma^2) \tag{5.3}$$

which means that the number of training errors will be exponentially decreasing in time.

First, by recursing, we have that:

$$\begin{aligned} D_{t+1}^i &= \frac{D_t^i}{Z_t} \exp(-\alpha_t y^i h_t(x^i)) \\ &= \frac{\frac{D_{t-1}^i}{Z_{t-1}} \exp(-\alpha_{t-1} y^i h_{t-1}(x^i))}{Z_t} \exp(-\alpha_t y^i h_T(x^i)) \\ &= \frac{D_{t-1}}{Z_t Z_{t-1}} \exp\left(-\left[\alpha_t y^i h_t(x^i) + \alpha_{t-1} y^i h_{t-1}(x^i)\right]\right) \\ &\dots \\ &= \frac{1}{N} \frac{\exp\left(-y^i \sum_{k=1}^{t} \alpha_k h_k(x^i)\right)}{\prod_{k=1}^{t} Z_k} \end{aligned} \tag{5.4}$$

or:

$$D_{T+1}^i = \frac{1}{N} \frac{\exp\left(-y^i f(x^i)\right)}{\prod_{k=1}^{T} Z_k} \tag{5.5}$$

which expresses the distribution on the data in terms of the classifier's score on them.

We use this result to obtain that:

$$\begin{aligned} \sum_{i=1}^{N} \frac{1}{N}[y^i \neq H(x^i)] &\overset{H(x)=\text{sign} f(x)}{=} \frac{1}{N} \sum_i \begin{cases} 1 & y^i f(x^i) \leq 0 \\ 0 & y_i f(x_i) > 0 \end{cases} \\ &= \sum_{i=1}^{N} \frac{1}{N} \exp(-y^i f(x^i)) \\ &= \sum_{i=1}^{N} D_{T+1}^i \prod_{k=1}^{T} Z_k \\ &= 1 \prod_{k=1}^{T} Z_k \end{aligned}$$

which relates the number of errors with the normalizing constants $Z_k$ at each round.

We set out to prove that:

$$\underbrace{\sum_{i=1}^{N} \frac{1}{N}[y^i \neq H(x^i)]}_{\text{Final Training Error}} \leq \prod_{t=1}^{T} \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] \tag{5.6}$$

so, given the last result, all we need to prove is that:

$$2\sqrt{\epsilon_t(1-\epsilon_t)} = Z_t$$

The key to this is the choice we made for $\alpha$:

$$\alpha = \log(\frac{\epsilon}{1-\epsilon}) \tag{5.7}$$

In specific, we have:

$$\begin{aligned}
Z_t &= \sum_{i=1}^{N} D_t^i \exp(-\alpha_t y^i h_t(x^i)) \\
&= \sum_{i:y^i \neq h_t(x^i)} D_t^i \exp(-\alpha_t y^i h_t(x^i)) + \sum_{i:y^i = h_t(x^i)} D_t^i \exp(-\alpha_t y^i h_t(x^i)) \\
&= \sum_{i:y^i \neq h_t(x^i)} D_t^i \exp(\alpha_t) + \sum_{i:y^i = h_t(x^i)} D_t^i \exp(-\alpha_t) \\
&= (\epsilon_t)\exp(\alpha_t) + (1-\epsilon_t)\exp(-\alpha_t) \\
\overset{\alpha=\frac{1}{2}\log(\frac{1-\epsilon}{\epsilon})}{=} & \epsilon_t \frac{\sqrt{1-\epsilon_t}}{\sqrt{\epsilon_t}} + (1-\epsilon_t)\frac{\sqrt{\epsilon_t}}{\sqrt{1-\epsilon_t}} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}$$

This concludes the proof.

### 5.1.2 Adaboost as coordinate descent

We demonstrated that boosting optimizes the following upper bound to the misclassification cost:

$$\begin{aligned}
\sum_{i=1}^{N} \frac{1}{N}[y^i \neq H(x^i)] &\leq \frac{1}{m}\exp(-y_i f(x^i)) &(5.8)\\
&= \prod_{t=1}^{T} Z_t \\
&= \prod_{t=1}^{T} [\epsilon_t \exp(\alpha_t) + (1-\epsilon_t)\exp(-\alpha_t)] &(5.9)
\end{aligned}$$

Actually the choice of $\alpha_t$ is such that $Z_t$ is minimized at each iteration:

$$\begin{aligned}
\frac{\partial \epsilon_t \exp(\alpha_t) + (1-\epsilon_t)\exp(-\alpha_t)}{\partial \alpha_t} &= 0 \\
\epsilon_t \exp(\alpha_t) - (1-\epsilon_t)\exp(-\alpha) &= 0 \\
\epsilon_t \exp(2\alpha_t) &= (1-\epsilon_t) \\
\alpha_t &= \frac{1}{2}\log\left(\frac{(1-\epsilon_t)}{\epsilon_t}\right)
\end{aligned}$$

which gives

$$\epsilon_t \exp(\alpha_t) + (1-\epsilon_t)\exp(-\alpha_t) = 2\sqrt{\epsilon_t(1-\epsilon_t)} \tag{5.10}$$

So choosing the best (in the sense of minimizing $Z_t$, and thence (5.8)) value of $\alpha$ tells us that $Z_t$ will equal $2\sqrt{\epsilon_t(1-\epsilon_t)}$, where $\epsilon_t < 1/2$ is the weighted error of the weak learner. Moreover $\sqrt{\epsilon_t(1-\epsilon_t)}$ is increasing in $\epsilon_t$ in $[0, 1/2]$, so finding the weak learner with the minimal $\epsilon_t$ guarantees that we most quickly decrease $(5.8)$.

We can conceptually describe what this amounts to as follows: we have at our disposal the responses of all weak learners (an infinite number of functions in general), and consider combining them with a weight vector, initially set to zero. In this setting, the Adaboost algorithm can be interpreted as *coordinate descent*:[1] we pick the 'dimension' (weak learner, $h_t$), and 'step' (voting strength, $\alpha_t$) that result in the sharpest decrease in the cost.

So we can see Adaboost as a greedy, but particularly effective optimization procedure.

### 5.1.3 Adaboost as building an additive model

A similar interpretation of Adaboost is in terms of fitting a linear model by finding at each step the optimal perturbation of a previously constructed decision rule. In specific consider that we have $f(x)$ at round $t$ and we want to find how to optimally perturb it by:

$$f'(x) = f(x) + \alpha h(x) \tag{5.11}$$

[1] Coordinate descent is an optimization technique that changes a single dimension of an optimized vector at each iteration

where $h(x)$ is a weak learner and $\alpha$ is the perturbation scale. To consider the impact that this perturbation will have on the cost we use a Taylor expansion as follows :

$$\begin{aligned}
\sum_{i=1}^{N} \exp(-y^i f'(x^i)) &= \sum_{i=1}^{M} \exp(-y^i[f(x^i) + \alpha h(x^i)]) \\
&\simeq \sum_{i=1}^{N} \exp(-y^i f(x^i))[1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2}(y^i)^2 h^2(x^i)] \\
&= \sum_{i=1}^{N} \exp(-y^i f(x^i))[1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2}] \\
&= Z \sum_{i=1}^{N} \underbrace{\frac{\exp(-y^i f(x^i))}{Z}}_{D^i} [1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2}]
\end{aligned}$$

where $Z = \sum_{i=1}^{N} \exp(-y^i f(x^i))$. So our goal is to minimize:

$$\sum_{i=1}^{N} D^i [1 - \alpha y^i h(x^i) + \frac{\alpha^2}{2}] \tag{5.12}$$

Finding the optimal $h$ amounts to solving:

$$h = \text{argmax} \sum_{i=1}^{N} D^i y^i h(x^i) = 1 - \epsilon \tag{5.13}$$

So picking the classifier with minimal $\epsilon$ amounts to finding the optimal $h$ And when optimizing wrt $\alpha$ we get:

$$\alpha = \text{argmax}_\alpha \sum_{i=1}^{N} D^i \exp(-\alpha y^i h(x^i)) = \epsilon \exp(\alpha) + (1 - \epsilon) \exp(-\alpha) \tag{5.14}$$

So, once more, we get the Adaboost algorithm.

### 5.1.4 Adaboost as a method for estimating the posterior

Consider the expectation of the exponential loss:

$$\begin{aligned}
E_{X,Y} \exp(-yf(x)) &= \int_x \sum_y \exp(-yf(x)) P(x, y) dx dy \\
&= \int_x \sum_y \exp(-yf(x)) P(y|x) dy P(x) dx \\
&= \int_x [[P(y = 1|x) \exp(-f(x)) + P(y = -1|x) \exp(f(x))] dy] \, dx
\end{aligned}$$

If a function $f(x)$ where to minimize this cost, at a point $x$ it should satisfy

$$f(x) = \text{argmax}_g[P(y=1|x)\exp(-g) + P(y=-1|x)\exp(g)$$

or

$$f(x) = \log\frac{P(y=1|x)}{P(y=-1|x)} P(y=1|x) = \frac{1}{1+\exp(-2yf(x))}$$

In Adaboost work with functions that are expressed as sums of weak learners. So the optimal $f(x)$ will not necessarily be given by the expression $\log\frac{P(y=1|x)}{P(y=-1|x)}$, but rather will be an approximation to it formed by a linear combination of weak learners.

## 5.2 Learning Theory

For pattern recognition:

- Small sample analysis.
- Bounds on worst performance.

Hoeffding inequality: Let $Z_1, \ldots, Z_m$ be iid, drawn from Bernoulli distribution: $P(Z_i) = \phi^{Z_i}(1-\phi)^{1-Z_i}$. Consider their mean: $\hat{\phi} = \frac{1}{m}\sum_{i=1}^m Z_i$. Then for any $\gamma > 0$,

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

On training set $\{x^i, y^i\}, i = 1 \ldots m$, consider training error for hypothesis $h$:

$$\hat{e}(h) = \frac{1}{m}\sum_{i=1}^m [h(x^i) \neq y^i]$$

Generalization error

$$e(h) = P_{(x,y)\propto D}([h(x) \neq y])$$

How different will $e$ be from $\hat{e}$ if training set is drawn from $D$?.

Consider an *Hypothesis Class*: the set of all classifiers considered. E.g. for linear classification, all linear classifiers.

Consider that we have a finite class:

$$\mathcal{H} = \{h_1, \ldots h_K\} \tag{5.15}$$

e.g. a small set of decision stumps for a few thresholds. No parameters to estimate.

Take any of these classifiers, $h_i$. Consider Bernoulli variable $Z$ generated by sampling $(x, y)$ $D$ and indicating whether $h_i(x) \neq y$. We can write the empirical error in terms of this Bernoulli variable:

$$\hat{e}(h_i) = \frac{1}{m}\sum_{j=1}^m Z_j$$

The mean of this variable is $e(h_i)$, i.e. the generalization error for the $i$-th classifier. We then have from the Hoeffding inequality:

$$P(|\hat{e}(h_i) - e(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

For large $m$, the training error will be close to the generalization error. Consider all possible classifiers:

$$\begin{aligned} P(\exists h \in \mathcal{H} : |\hat{e}(h) - e(h)| > \gamma) &\leq \sum_{i=1}^{K} P(|\hat{e}(h_i) - e(h)| > \gamma) \\ &\leq \sum_{k=1}^{K} 2\exp(-2\gamma^2 m) \\ &\leq K2\exp(-2\gamma^2 m) \end{aligned}$$

Union Bound:

$$P(A_1 \cup A_2 \ldots \cup A_k) \leq P(A_1) + \ldots P(A_k)$$

Consider we have $m$ training points, $k$ hypotheses and want to find a $\gamma$ such that $P(|\hat{e} - e| > \gamma) = \delta$. Plugging in the above equation we have $\gamma = \sqrt{\frac{1}{2m}\log\frac{2k}{\delta}}$. So with probability $1 - \delta$ we have for all $h \in \mathcal{H}$:

$$|\hat{e}(h) - e(h)| \leq \sqrt{\frac{1}{2m}\log\frac{2k}{\delta}} \tag{5.16}$$

What does this mean? Consider $h^*$ to be the best possible hypothesis, in terms of generalization error and $\hat{h}$ to be the best in terms of training error. We then have

$$\begin{aligned} e(\hat{h}) &\leq \hat{e}(\hat{h}) + \gamma \\ &\leq \hat{e}(h^*) + \gamma \\ &\leq e(h^*) + 2\gamma \end{aligned} \tag{5.17}$$

Generalization error of chosen hypothesis is not much larger than that of the optimal one.

Bias/Variance: If we have small hypothesis set, we may have large $e(\hat{h})$, but it will not differ much from $e(h^*)$. If we have large hypothesis set, $e(\hat{h})$ becomes smaller, but $\gamma$ increases.