Fondements de l'apprentissage machine

Automne 2014

Roland Memisevic

Leçon 2

Plan

- Régression linéaire
- Apprentissage par optimisation
- Fonctions de base non-linéaires
- Apprentissage par maximum de vraisemblance
- La décomposition biais–variance
- Un premier aperçu de la modelisation Bayésienne

Régression linéaire

- La régression linéaire est un des concepts les plus fondamentaux de l'apprentissage machine et des statistiques.
- Il s'agit d'un simple problème avec une solution simple.
- Pourtant, elle nous permet d'étudier une variété de concepts au centre de l'apprentissage machine, avec lesquels nous travaillerons durant ce cours.
- En raison de sa simplicité, la régression linéaire est également utilisée dans de très nombreuses tâches pratiques.

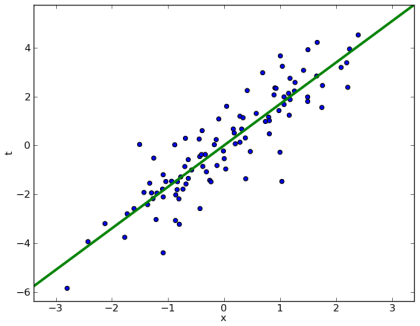
Régression linéaire

$$\mathbf{x} \rightarrow \mathbf{t}$$

- Soit un ensemble d'observations $\mathbf{x}$ et $\mathbf{t}$ à valeurs réelles.
- Problème : *Apprendre à prédire* $\mathbf{t}$ *à partir de* $\mathbf{x}$.
- Il s'agit d'un problème *d'apprentissage supervisé*.

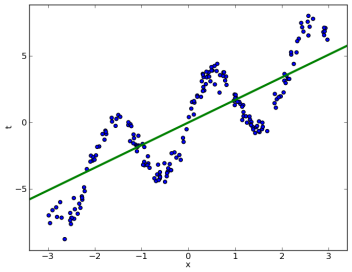
Régression linéaire 1-d

- Si les entrées et les sorties sont des scalaires (1-d), on peut les visualiser :



- La régression linéaire est basée sur l'hypothèse qu'il existe une relation linéaire entre $\mathbf{x}$ et $\mathbf{t}$.

Bruit vs. dépendances dont on ne se soucie pas



- Même si cette hypothèse n'est pas tout à fait correcte, on peut utiliser la régression linéaire pour capturer la *tendance linéaire* dans les données (en supposant que les dépendances non-linéaires sont du bruit)

Régression en 1-d

- Pour des entrées/sorties 1-d on a :

$$y(x) = w_0 + w_1 x$$

- Pour des entrées en D dimensions et des sorties 1-d on a :

$$y(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \big(= w_0 + \mathbf{w}^{\mathrm{T}} \mathbf{x} \big)$$

- Pour des sorties en K dimensions on a simplement un modèle pour chaque dimension de $\boldsymbol{y}$:

$$\begin{pmatrix} y_1(\mathbf{x}) \\ \vdots \\ y_K(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} y_1(\mathbf{x}) = w_{10} + w_{11} x_1 + w_{12} x_2 + \ldots + w_{1D} x_D \\ \vdots \\ y_K(\mathbf{x}) = w_{K0} + w_{K1} x_1 + w_{K2} x_2 + \ldots + w_{KD} x_D \end{pmatrix}$$

Le biais

- w_0 s'appelle biais («bias» en anglais). Il nous permet de deplacer le modèle linéaire à travers l'axe des $\boldsymbol{y}$.
- On peut toujours éliminer le biais (et c'est courant) en remplaçant :

$$\begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix}$$

Maintenant le modèle linéaire contient implicitement le biais (ce qui nous permet d'écrire $\mathbf{w}^{\mathrm{T}} \mathbf{x}$).

Méthode des moindres carrés

- Estimation des paramètres ("poids"). (sorties 1-d pour le moment)
- Pour estimer les paramètres, il faut avoir un ensemble d'entraînement

$$\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$$

dont on minimise la *somme des erreurs au carré* :

$$E(\mathbf{w}; \mathcal{D}) = \frac{1}{2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^{\mathrm{T}} \mathbf{x}_n\right)^2$$

par rapport à $\mathbf{w}$.

- D'autres fonctions de perte peuvent être utilisées, mais celle-ci rend l'optimisation plus facile.

Méthode des moindres carrés

- Pour l'optimiser par rapport à $\mathbf{w}$ on dérive

$$\frac{\partial E}{\partial \mathbf{w}} = -\sum_{n=1}^{N} \left(t_n - \mathbf{w}^{\mathrm{T}} \mathbf{x}_n\right) \mathbf{x}_n$$

- Définissant la dérivée à zéro

$$-\sum_{n=1}^{N} t_n \mathbf{x}_n + \left(\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}\right) \mathbf{w} = 0$$

on obtient :

$$\mathbf{w} = \left(\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}\right)^{-1} \left(\sum_{n=1}^{N} t_n \mathbf{x}_n\right)$$

- (Il est plus simple et instructif de le faire pour les entrées 1-d.)

Méthode des moindres carrés

- Pour formuler de manière plus compacte, on définit $\mathbf{t}$ le vecteur de sorties

$$\mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix}$$

ainsi que $\mathbf{X}$ une matrice d'entrées (une par rangée) :

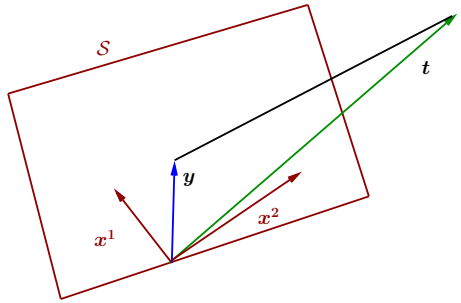
$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix}$$

Cela nous permet d'écrire la solution de manière plus compacte :

Les équations normales

$$\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}} \mathbf{X}\right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{t}$$

Une interprétation géométrique



- Pensez à l'erreur comme la norme au carré de la différence entre les vecteurs $\mathbf{t}$ et $\mathbf{y}$, contenant toutes les sorties et toutes les prédictions, respectivement.
- Le vecteur $\mathbf{y}$ $\left(= \mathbf{X}\mathbf{w}\right)$ réside dans le sous-espace $\mathcal{S}$ engendré par les colonnes $\mathbf{x}^i$ de $\mathbf{X}$.
- Il s'agit de la *projection orthogonale* de $\mathbf{t}$ sur $\mathcal{S}$.

Des sorties multidimensionelles

- La régression avec des sorties multidimensionnelles est similaire au cas 1-d.
- Imaginez un modèle séparé pour chaque sortie.
- Nous avons une *matrice* $\mathbf{W}$ de paramètres et minimisons

$$E(\mathbf{W}; \mathcal{D}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{W}^{\mathrm{T}} \mathbf{x}_n\|^2$$

Des sorties multidimensionelles

Les équations normales (sorties multidimensionnelles)

$$\mathbf{W} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{T}$$

où chaque colonne de $\mathbf{T}$ contient un vecteur, $\mathbf{t}_k$, de N sorties.

- La régression multidimensionnelle est comme K régressions 1-d indépendantes.

Descente de gradient stochastique Stochastic Gradient Descent (SGD)

- Comme l'a montré la régression linéaire, l'apprentissage peut être défini comme l'optimisation d'une fonction de perte.
- Souvent, on n'a pas de solution de forme fermée.
- Également, les données arrivent souvent un point à la fois et nous voudrions faire des prédictions au fur et à mesure qu'elles arrivent.
- Une solution commune : l'apprentissage en ligne.
- Une approche simple et commune est Descente de Gradient Stochastique ("Stochastic Gradient Descent/SGD").
- SGD est basée sur le fait que le gradient est dans la direction de descente maximale.

Descente de gradient stochastique Stochastic Gradient Descent (SGD)

- Elle s'applique lorsque la fonction de perte se décompose en une *somme sur les exemples d'entraînement* :

$$E = \sum_n E_n$$

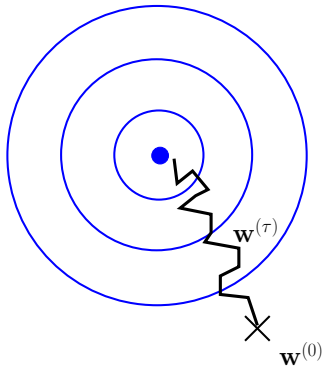
(où E_n ne dépend que d'un exemple d'entraînement n).

Descente de gradient stochastique

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \frac{\partial E_n}{\partial \mathbf{w}}$$

- τ est l'itération
- η est le taux d'apprentissage (*learning rate*) (normalement une petite valeur réelle (par exemple $\eta = 0.001$). On peut le réduire pendant l'apprentissage.

Descente de gradient stochastique Stochastic Gradient Descent (SGD)



- À chaque itération, l'algorithme voit un exemple d'entraînement.
- Il vacille autour d'une trajectoire moyenne idéalisée vers l'optimum.

Descente de gradient stochastique Stochastic Gradient Descent (SGD)

- Gradient Descent s'applique aussi á l'optimisation "batch" (non-en ligne) mais il n'est pas courant dans ce cas.
- Pour la régression linéaire SGD correspond à des mise à jour :

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta\big(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\mathbf{x}_n\big)\mathbf{x}_n$$

Fonctions de base non-linéaires

- Il existe un moyen très simple d'étendre la régression linéaire à la régression non-linéaire :
- Prétraiter les entrées en utilisant un ensemble de fonctions non-linéaires (qui restent fixes).
- Remplacer

$$\mathbf{x} \to \mathbf{\Phi}(\mathbf{x})$$

où $\mathbf{\Phi}$ est une fonction non-linéaire (fixe !).

- Le modèle est toujours linéaire par rapport aux paramètres du modèle.
- Cette approche s'appelle aussi « l'expansion de base ».

Examples de fonctions de base

- Modèle polynomial :

$$\phi_j(x) = x^j \quad j = 1, \ldots, M$$

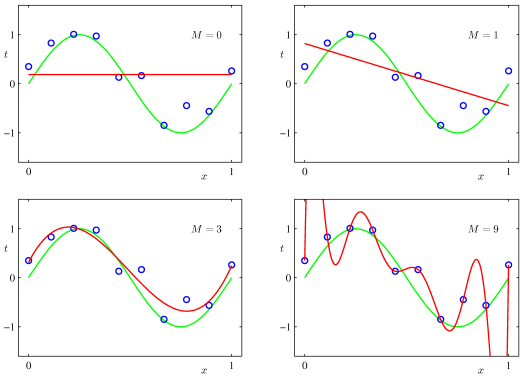
- Modèle polynomial (multidimensionnel, ordre 2) :

$$\phi_{ik}(\mathbf{x}) = x_i x_k \quad \phi_i(\mathbf{x}) = x_i$$

- Fonctions de base « radiales » ou « Gaussiennes » :

$$\phi_j(x) = \exp\Big(-\big(\frac{x-\mu_j}{2s^2}\big)^2\Big)$$

Régression polynomiale



- Modèle polynomial de la leçon 1.
- Notez que dans chaque exemple, l'entrée de la régression est de dimension différente même si l'entrée originale ($\mathbf{x}$) est toujours de dimension 1.

Bruit gaussien et maximum de vraisemblance

- Jusqu'à présent, nous avons traité l'apprentissage comme un problème d'optimisation.
- Nous obtenons une interprétation probabiliste si nous faisons l'hypothèse suivante :

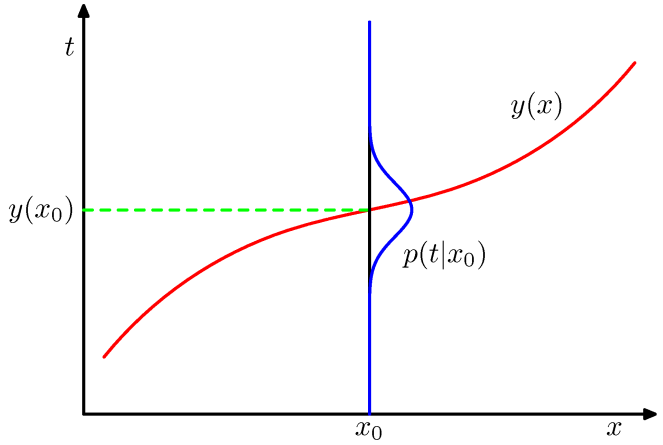
$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

où ϵ est une variable aléatoire gaussienne.

- Ainsi

$$p(t|\mathbf{x}; \mathbf{w}) = \mathcal{N}\big(t|y(\mathbf{x}, \mathbf{w}), \sigma^2\big)$$

Bruit gaussien et maximum de vraisemblance



Bruit gaussien et maximum de vraisemblance

- Pour estimer les paramètres de la gaussienne conditionnelle en utilisant des données d'entraînement $\mathcal{D} = \big\{(\mathbf{x}_n, t_n)\big\}_{n=1}^N$ IID (= « indépendantes et identiquement distribuées ») maximisez

$$p(\mathcal{D}) = \prod_n \mathcal{N}(t_n|y(\mathbf{x}_n, \mathbf{w}), \sigma^2)$$

- Équivalemment, nous pouvons maximiser la **log vraisemblance** (car le log est monotone) :

$$\text{maximisez} \quad -\sum_{n=1}^{N} \big(t_n - \mathbf{w}^{\mathrm{T}}\mathbf{x}_n\big)^2 + \text{const.}$$

ce qui est équivalent à minimiser la *somme des erreurs au carré* $E(\mathbf{w}; \mathcal{D})$ d'auparavant.

Bruit gaussien et maximum de vraisemblance

- Le maximum de vraisemblance est un principe (ou « recette ») pour mettre à jour les paramètres d'une distribution.
- Il s'applique à des distributions conditionnelles ou inconditionnelles.
- Il s'applique à des distributions discrètes ou continues.
- Le maximum de vraisemblance (et la méthode des moindres carrés en général) peut souffrir de sur-apprentissage.

La méthode des moindres carrés régularisée

- Pour prévenir le sur-apprentissage, on peut pénaliser la fonction de perte :

$$E_\lambda(\mathbf{w}, \mathcal{D}) = E(\mathbf{w}, \mathcal{D}) + \lambda E_W(\mathbf{w})$$

  pour réduire la taille de l'espace d'hypothèses.
- La pénalité la plus commune supprime les grandes entrées de $\mathbf{w}$:

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

  « **Weight decay** » ou **régression de ridge** (« **Ridge Regression** »)

La méthode des moindres carrés régularisée

- Il est facile de montrer que la solution devient :

Les équations normales pour la régression de ridge

$$\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{t}$$

- Notez que, dans la régression polynomiale, la pénalité décourage les polynômes « ondulés » (qui sont due à de grands coefficients).
- Il faut choisir λ ! (par exemple par validation croisée)

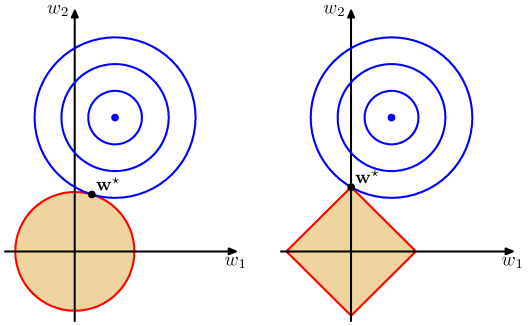
Régression Lasso

- Une alternative commune est de pénaliser la norme L1 des poids :

$$E_W(\mathbf{w}) = \lambda \sum_j |w_j|$$

- Cela s'appelle régression **Lasso** (« least absolute shrinkage and selection operator », Tibshirani 1996).
- Avantage : Il conduit à des solutions *éparses*, ou *clairsemées* (« sparse ») : de nombreux poids w_i ont exactement la valeur zéro.
- Désavantage : Il n'existe pas de solution de forme fermée.

Lasso : intuition



(weight decay) (lasso)

- La minimisation de

  "fonction de perte + terme de pénalité"

  est comme optimiser la fonction de perte *avec une contrainte* sur les paramètres.

Biais-variance

- Supposons que les données sont tirées d'une distribution jointe $p(\mathbf{x}, t)$.
- La meilleure prédiction possible sur un exemple de test $\mathbf{x}_n$ en termes d'erreur quadratique est l'espérance conditionnelle

$$\mathbb{E}[t|\mathbf{x}_n] = \int t p(t|\mathbf{x}_n)\, \mathrm{d}t$$

- En pratique, nous ne pouvons jamais produire cette prédiction optimale, car la quantité de données d'entraînement est toujours limitée.

Biais-variance

- Imaginez que vous estimez les paramètres $\mathbf{w}$ plusieurs fois, chaque fois en utilisant un ensemble d'entraînement $\mathcal{D}$ différent.
- Pour un exemple de test $\mathbf{x}_n$, la moyenne de l'erreur (par rapport au choix d'ensemble de données) est :

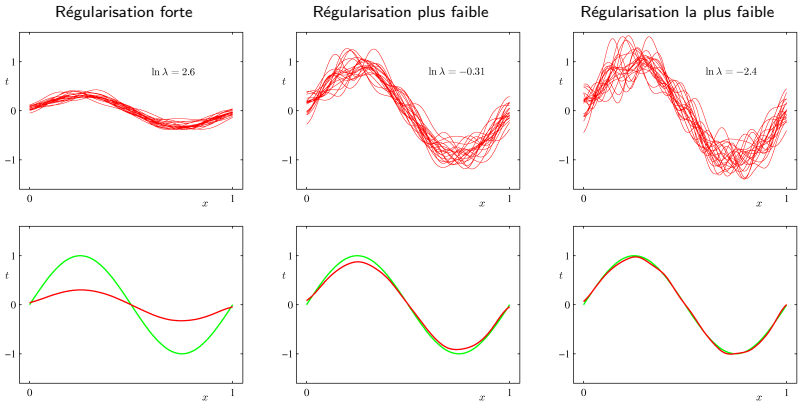
$$\mathbb{E}_{\mathcal{D}}[\big(y(\mathbf{x}_n;\mathcal{D}) - \mathbb{E}[t|\mathbf{x}_n]\big)^2]$$

Cela est égal à (dérivation, Bishop page 149) :

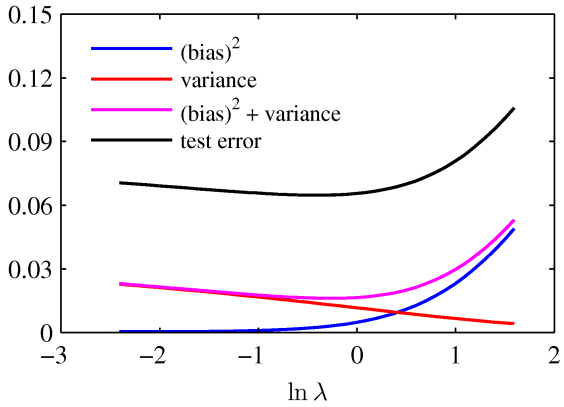
$$\big(\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}_n;\mathcal{D})] - \mathbb{E}[t|\mathbf{x}_n]\big)^2 \;+\; \mathbb{E}_{\mathcal{D}}[\big(y(\mathbf{x}_n;\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}_n;\mathcal{D})]\big)^2]$$

$$=: \big(\text{biais}\big)^2 + \text{variance}$$

Biais-variance



Biais-variance



Un aperçu de l'apprentissage bayésien

- Le maximum de vraisemblance est une méthode pour mettre à jour des paramètres afin que la probabilité des données d'entraînement devienne maximale.
- La régularisation est nécessaire pour prévenir le sur-apprentissage.
- On peut éliminer le sur-apprentissage en remplaçant le réglage de paramètres par un raisonnement probabiliste.
- Cela nécessite de traiter les paramètres eux-mêmes comme des variables aléatoires.

Un aperçu de l'apprentissage bayésien

- Écrivez la vraisemblance comme une *distribution conditionnelle* $p(\mathcal{D}|\mathbf{w})$ (au lieu d'une distribution qui est *paramétrisée* par $\mathbf{w}$).
- En utilisant la règle de Bayes :

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

- $p(\mathbf{w}|\mathcal{D})$ est appelé distribution postérieure de $\mathbf{w}$.
- Pour être en mesure de la calculer, il faut d'abord définir une distribution $p(\mathbf{w})$, connue comme la distribution à priori de $\mathbf{w}$.

Un aperçu de l'apprentissage bayésien

- Pensez à la modélisation bayésienne comme un moyen de mettre à jour nos suppositions $p(\mathbf{w})$ concernant les paramètres en présence de données $\mathcal{D}$.
- Malheureusement, cela est souvent très difficile en pratique, nécessitant fréquemment des approximations.
- Par exemple, le calcul de la constante de normalisation $p(\mathcal{D})$ est souvent coûteux.

Exemple de la régression bayésienne en utilisant une distribution à priori gaussienne pour $\mathbf{w}$