

BAB III

LANDASAN TEORI

3.1 Definisi Peramalan

Definisi Peramalan adalah memperkiraan atau memproyeksikan sesuatu yang akan terjadi dimasa sekarang maupun yang akan datang. dikarena masa yang akan datang tidak bisa dipastikan, maka dibutuhkan beberapa prediksi baik eksplisit maupun implisit. Peramalan bertujuan member informasi yang ada pada masa sekarang ini sebagai bahan arahan aktivitas di masa yang akan datang untuk mencapai suatu tujuan (Makridatis, 1999). Dalam peramalan pergerakan harga saham, harga penutupan dari indeks harga saham gabungan dapat dipengaruhi oleh faktor harga lainnya, yaitu harga minyak dan harga emas(Muharam et al, 2013).

3.2 Definisi Least Squares Support Vector Machines (LSSVM)

LSSVM (Least Squares Support Vector Machines) merupakan sebuah metode yang merupakan hasil modifikasi dari metode Support Vector Machines (SVM) yang dilakukan oleh Suykens et al. (2002). Dalam Penelitian Suykens et al. ini bertujuan untuk memecahkan persamaan – persamaan linier dengan cara yang lebih sederhana dibandingkan metode sebelumnya, yaitu metode SVM (Suykens et al, 2002). Selain memiliki kompleksitas kalkulasi yang lebih rendah,

proses training LSSVM dalam permasalahan berskala besar juga lebih cepat, karena menggunakan resource komputasi yang lebih sedikit dibanding metode SVM (Mustaffa et al, 2011). Dalam penggunaan kernel RBF, jumlah parameter yang harus dioptimalkan pada LSSVM juga lebih sedikit dibanding SVM, karena pada LSSVM hanya ada dua variable yang perlu ditentukan, yaitu (γ, σ^2) , sedangkan pada SVM terdapat tiga parameter, yaitu (C, σ^2, ϵ) (Ou, & Wang, 2009).

Seperti pada SVM, LSSVM juga dapat digunakan untuk permasalahan klasifikasi dan regresi baik pada kasus linear maupun non linear. Untuk kasus nonlinear, teknik kernel juga dapat diaplikasikan di dalam LSSVM. Pilihan kernel yang dapat digunakan sama seperti SVM, yaitu linear, polinomial, RBF, dan MLP.

3.2.1 LSSVM untuk klasifikasi

Pemandu Untuk klasifikasi, dalam model primal space, decision function dirumuskan sebagai berikut :

$$y(x) = \text{sign}[w \cdot \phi(x) + b] \dots\dots\dots(1)$$

dimana $\phi(.) : R^n \rightarrow R^{nh}$ adalah pemetaan ke ruang fitur high dimensional seperti pada SVM standar. Berikut adalah rumusan problema optimasi dalam LSSVM yang diperkenalkan oleh Suykens et al, (2002) :

$$\min_{w,b,\epsilon} J_p(w, \epsilon) = \frac{1}{2} w \cdot w + \gamma \frac{1}{2} \sum_{i=1}^N e_i^2 \dots\dots\dots(2)$$

Sedemikian hingga :

$$y_i[w \cdot \varphi(x_i) + b] = 1 - e_i, \quad i = 1, \dots, N \quad \dots\dots\dots(3)$$

Terlihat dua hal yang membedakan hasil modifikasi ini terhadap formulasi asli oleh vapnik. Yang pertama adalah pada penggunaan fungsi persamaan yang menggantikan pertidaksamaan dimana nilai 1 di bagian kanan lebih merupakan suatu nilai target alih – alih nilai threshold. Untuk mentolerir kesalahan, persamaan dilengkapi dengan suatu variable kesalahan e_i yang memiliki peran sama seperti ξ_i dalam rumus SVM. Perbedaan kedua yaitu pada penggunaan fungsi kuadrat untuk variable kesalahan tersebut.

Untuk kasus nonlinier, diturunkan rumus Lagrangian berikut :

$$L(w, b, e; \alpha) = J_p(w, e) - \sum_{i=1}^N \alpha_i \{y_i[w \cdot \varphi(x_i) + b] - 1 + e_i\} \quad \dots\dots\dots(4)$$

Dimana nilai α_i adalah Lagrange multipliers, yang bisa bernilai positif maupun negative. Untuk mencapai nilai optimal, kondisi berikut harus terpenuhi :

$$\begin{cases} \frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i=1}^N \alpha_i y_i \varphi(x_i) \\ \frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i y_i = 0 \\ \frac{\partial L}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma_{e_i}, \quad i = 1, \dots, N \\ \frac{\partial L}{\partial \alpha_i} = 0 \rightarrow y_i[w \cdot \varphi(x_i) + b] - 1 + e_i = 0, \quad i = 1, \dots, N \end{cases} \quad \dots\dots\dots(5)$$

Dengan mengeliminasi w dan e, diperoleh system linear :

$$\begin{bmatrix} 0 & y^T \\ y & \Omega + I/\gamma \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_v \end{bmatrix},_3$$

$$\dots\dots\dots(6)$$

Dimana :

$$\begin{aligned} Z^T &= [\varphi(x_1) \cdot y_1; \dots; \varphi(x_N) \cdot y_N] \\ y &= [y_1; \dots; y_N] \\ \mathbf{1}_v &= [1; \dots; 1] \\ e &= [e_1; \dots; e_N] \\ \alpha &= [\alpha_1; \dots; \alpha_N] \end{aligned} \dots\dots\dots(7)$$

Dan $\Omega = Z^T Z$. Trik kernel dapat diaplikasikan di dalam matriks Ω :

$$\begin{aligned} \Omega_{ij} &= y_i y_j \varphi(x_i) \cdot \varphi(x_k) \\ &= y_i y_j K(x_i, x_j) \end{aligned} \dots\dots\dots(8)$$

Untuk $i, j = 1, \dots, N$.

Dengan demikian, model LSSVM yang dihasilkan dalam dual space adalah sebagai berikut :

$$y(x) = \text{sign} \left[\sum_{i=1}^N \alpha_i y_i K(x, x_i) + b \right] \dots\dots\dots(9)$$

3.2.2 LSSVM untuk estimasi fungsi

Penurunan rumus pada LSSVM untuk estimasi fungsi pada dasarnya sama seperti pada LSSVM untuk klasifikasi karena LSSVM klasifikasi sebenarnya dapat dipahami sebagai kasus regresi dengan target +1 dan -1.

Misalkan decision function dari model LSSVM dalam primal space dirumuskan sebagai berikut (Suykens et al, 2002) :

$$y(x) = w \cdot \varphi(x) + b \quad \dots\dots\dots(10)$$

Dimana $\varphi(.) : R^n \rightarrow R^{n_h}$ adalah pemetaan ke ruang fitur high dimensional seperti pada SVM standar. Maka, dirumuskan problema optimasi dalam LSSVM regresi, yaitu :

$$\min_{w,b,e} J_p(w,e) = \frac{1}{2} w \cdot w + \gamma \frac{1}{2} \sum_{i=1}^N e_i^2 \quad \dots\dots\dots(11)$$

Sedemikian sehingga

$$y_i = w \cdot \varphi(x_i) + b + e_i, \quad i = 1, \dots, N \quad \dots\dots\dots(12)$$

Rumus Lagrangian yang dibangun untuk kasus nonlinier sebagai berikut :

$$L(w,b,e;\alpha) = J_p(w,e) - \sum_{i=1}^N \alpha_i \{w \cdot \varphi(x_i) + b + e_i - y_i\} \quad \dots\dots\dots(13)$$

Dimana nilai α_i adalah Lagrange multipliers. Kondisi optimal yang harus dipenuhi antara lain :

$$\begin{cases} \frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i=1}^N \alpha_i \varphi(x_i) \\ \frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i = 0 \\ \frac{\partial L}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i, \quad i = 1, \dots, N \\ \frac{\partial L}{\partial \alpha_i} = 0 \rightarrow w \cdot \varphi(x_i) + b + e_i - y_i = 0, \quad i = 1, \dots, N \end{cases} \quad \dots\dots\dots(14)$$

Setelah mengeliminasi w dan e , diperoleh :

$$\dots \begin{bmatrix} 0 & 1_v^T \\ y & \Omega + I/\gamma \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \dots\dots\dots(15)$$

dimana :

$$\begin{aligned} y &= [y_1; \dots; y_N] \\ 1_v &= [1; \dots; 1] \\ \alpha &= [\alpha_1; \dots; \alpha_N] \end{aligned} \dots\dots\dots(16)$$

Trik kernel dapat diaplikasikan sebagai berikut :

$$\begin{aligned} \Omega_{ij} &= \varphi(x_i) \cdot \varphi(x_k) \\ &= K(x_i, x_j) \end{aligned} \dots\dots\dots(17)$$

Untuk $i, j = 1, \dots, N$.

Dengan demikian, fungsi estimasi yang dihasilkan adalah :

$$y(x) = \sum_{i=1}^N \alpha_i K(x, x_i) + b \dots\dots\dots(18)$$

3.3 Definisi Backpropagation

Backpropagation merupakan suatu algoritma pembelajaran terawasi yang menggunakan perceptron dengan lapisan yang banyak untuk mengubah bobot yang terhubung pada neuron yang berada pada lapisan yang tersembunyi. *Error Output* digunakan untuk mengubah perubahn nilai bobotnya dalam backward (arah mundur) dan untuk mendapatkan nilai error dilakukan pekerjaan tahapan

forward propagation atau perambatan maju terlebih dahulu. Dalam perambatan maju diaktifkan neuron pada fungsi aktivasi yang dapat dideferensiasikan (Kusumadewi, 2010) :

$$y = F(X) = \frac{1}{1 + e^{-\sigma X}}$$

$$f'(X) = \sigma f(X)[1 - f(X)] \dots\dots\dots(20)$$

Atau seperti tangent sigmoid :

$$y = F(X) = \frac{e^X - e^{-X}}{e^X + e^{-X}} \dots\dots\dots(21)$$

$$f'(X) = [1 + f(X)][1 - f(X)]$$

Pelatihan Backpropagation dilakukan melalui langkah-langkah berikut ini :

Langkah ke-0 : Inisialisasi bobot;

Langkah ke-1 : Selama kondisi berhenti bernilai salah, kerjakan langkah 2-9;

Langkah ke-2 : Untuk setiap data *training*, lakukan langkah 3-8.

Umpan Maju (*Feedforward*)

Langkah ke-3 : Setiap unit input ($X_i, i = 1, \dots, n$) menerima sinyal input

X_i dan menyebarkan sinyal tersebut ke seluruh unit tersembunyi.

Langkah ke-4 : Pada unit tersembunyi ($Z_j, j = 1, \dots, p$) menunjukan

sinyal-sinyal input yang sudah berbobot (termasuk biasanya)

$$Z_{in_j} = v_{0j} + \sum_{i=1}^n X_i V_{ij} \dots\dots\dots(22)$$

Lalu menghitung sinyal output dari unit tersembunyi dengan menggunakan fungsi fungsi aktivasi yang telah ditentukan :

$$Z_j = f(Z_in_j) \dots\dots\dots(23)$$

Sinyal output ini selanjutnya dikirim ke seluruh unit pada unit atas (unit output)

Langkah ke-5 : Tiap-tiap output ($Y_k, k = 1, \dots, m$) menjumlahkan bobot sinyal input :

$$y_in_k = w_{0k} + \sum_{i=1}^n Z_i w_{jk} \dots\dots\dots(24)$$

Lalu menghitung sinyal output dari unit output bersangkutan dengan menggunakan fungsi aktivasi yang telah ditentukan.

$$y_k = f(y_in_k) \dots\dots\dots(25)$$

Sinyal output ini selanjutnya dikirim keseluruhan unit pada output.

Umpan Mundur/ Propagasi Error (*Backpropagation of Error*)

Langkah ke-6 : Setiap unit output ($Y_k, k = 1, \dots, m$) menerima suatu pola target yang sesuai dengan pola input pelatihan, untuk menghitung kesalahan (error) antara target dengan output yang dihasilkan jaringan

$$\delta_k = (t_k - y_k) f'(y_in_k) \dots\dots\dots(26)$$

Factor δ_k digunakan untuk menghitung koreksi

error (ΔW_{jk}) yang nantinya akan dipakai untuk

memperbaiki W_{jk} , dimana $\Delta W_{jk} = \alpha \delta_k Z_j$

Selain itu juga dihitung koreksi bias ΔW_{0k} yang nantinya

akan dipakai untuk memperbaiki W_{0k} , dimana

$\Delta W_{0k} = \alpha \delta_k$ Faktor δ_k kemudian dikirim pada lapisan

yang berada pada langkah ke-7.

Langkah ke-7 : Setiap unit tersembunyi $(Z_j, j = 1, \dots, p)$ menerima input delta (dari langkah ke-6) yang sudah berbobot

$$\delta_{in_j} = w_{0k} + \sum_{k=1}^m \delta_k w_{jk} \dots\dots\dots(27)$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghitung informasi kesalahan error δ_j , dimana :

$$\delta_j = \delta_{in_j} f'(Z_{in_j}) \dots\dots\dots(28)$$

Kemudian hitunglah koreksi bobot (untuk memperbaiki V_{ij})

$$\Delta v_{ij} = \alpha \delta_j x_i \dots\dots\dots(29)$$

Setelah itu hitung koreksi bias (digunakan untuk memperbaiki V_{0j})

$$\Delta v_{0j} = \alpha \delta_j \dots\dots\dots(30)$$

Update Bobot dan Bias (Adjustment)

Langkah ke-8 : Setiap unit output ($Y_k, k = 1, \dots, m$) memperbaiki bobot

dan bias dari setiap unit tersembunyi ($j = 0, \dots, p$)

$$W_{jk}(\text{baru}) = W_{jk}(\text{lama}) + \Delta W_{jk} \dots\dots\dots(31)$$

Demikian pula untuk setiap unit tersembunyi

($Z_j, j = 1, \dots, p$) akan memperbaharui bobot dan bias dari setiap unit input ($i = 0, \dots, n$)

$$v_{ij}(\text{baru}) = v_{ij}(\text{lama}) + \Delta v_{ij} \dots\dots\dots(32)$$

Langkah ke-9 : Test kondisi berhenti apabila error ditemukan Jika kondisi

STOP telah terpenuhi, maka pelatihan jaringan dapat dihentikan. Untuk memeriksa kondisi STOP, biasanya digunakan kriteria MSE (Mean Square Error) berikut ini :

MSE

$$= 0.5$$

$$X \{(t_{k1} - y_{k1})^2 + (t_{k2} - y_{k2})^2 + \dots \dots\dots(33) \\ + (t_{km} - y_{km})^2\}$$

3.4 Sampel data dari IHSG, harga emas, dan harga minyak

Tanggal	Open	High	Low	Close	Gold	Oil
1 Apr 2010	25339480	25760560	25328960	25754130	5155826	6443140
2 Apr 2010	25756169	26060690	25756170	26052770	5215554	6518173
...	...	...	...	...	...	...
...	...	...	...	...	...	...
30 Jul 2015	47402500	47547210	47115630	47124920	9429984	11889803
31 Jul 2015	47216600	48025290	47214670	48025290	9610058	12009323